A Toolkit for Managing Enterprise Privacy Policies

Michael Backes, Birgit Pfitzmann, and Matthias Schunter

IBM Zurich Research Laboratory, Rüschlikon, Switzerland
{mbc,bpf,mts}@zurich.ibm.com

Abstract. Enterprise privacy enforcement allows enterprises to internally enforce a privacy policy that the enterprise has decided to comply to. An enterprise privacy policy often reflects different legal regulations, promises made to customers, as well as more restrictive internal practices of the enterprise. Further, it may allow customer preferences. Hence it may be authored, maintained, and audited in a distributed fashion.
Our goal is to provide the tools for such management of enterprise privacy policies. The syntax and semantics is a superset of the Enterprise Privacy Authorization Language (EPAL) recently proposed by IBM. The basic definition is refinement, i.e., the question whether fulfilling one policy automatically fulfills another one. This underlies auditing of a policy against an old or new regulation or promise and transferring data into a realm with a different policy. It is also the semantic basis for composition operators. We further define such composition operators for different purposes. Our main focus it to combine usability for enterprises, e.g., by treating multiple terminologies, incomplete data, and different types of errors and defaults, with the formal rigor needed to make privacy compliance meaningful and predictable.

1 Introduction

An increasing number of enterprises make privacy promises to customers or, at least in the US and Canada, fall under new privacy regulations. To ensure adherence to these promises and regulations, enterprise privacy technologies are emerging [8]. An important tool for enterprise privacy enforcement is formalized enterprise privacy policies [10, 17, 16]. Compared with the well-known language P3P [19] intended for privacy promises to customers, languages for the internal privacy practices of enterprises and for technical privacy enforcement must offer more possibilities for fine-grained distinction of data users, purposes, etc., as well as a clearer semantics.

Although the primary purpose of enterprise privacy policies is enterprise-internal use, many factors speak for standardization of such policies: First, it would allow certain technical parts of regulations to be encoded into such a standardized language once and for all. Secondly, a large enterprise with heterogeneous repositories of personal data could then hope that enforcement tools for all these repositories become available that allow the enterprise to consistently enforce at least the internal privacy practices chosen by the CPO (chief privacy officer). Thirdly, with increasingly dynamic e-business, data will be exchanged between enterprises, and enterprise boundaries change due to mergers, acquisitions, or virtual enterprises. Then the sticky-policy paradigm stressed in papers like [17] must be enforced. It states that the policy under which data have been collected has to govern the use of these data at all times. This also requires compatible enterprise privacy enforcement mechanisms. For these reasons, IBM has recently proposed an Enterprise Privacy Authorization Language (EPAL) [1] as an XML specification for public comments and possible subsequent input to standardization.

An enterprise privacy policy often reflects different legal regulations, promises made to customers, as well as more restrictive internal practices of the enterprise. Further, it may allow customer preferences. Hence it may be authored, maintained, replaced, and audited in a distributed fashion. In other words, one will need a life-cycle management system for the collection of enterprise privacy policies. While such thoughts occur as motivation in most prior work on enterprise privacy policies, no actual definitions and algorithms needed for these management tools have been proposed.

The overall goal of this article is therefore to provide a comprehensive range of tools for designing and managing privacy policies in an enterprise. We do this concretely for the IBM EPAL proposal. However, for a scientific paper we cannot use the lengthy XML syntax, but have to use a corresponding abstract syntax presented in [2] (which, like EPAL, is based on [17]). Our paper reflects recent updates made between the earlier abstract [2] and the published specification and XML Schema [1], so that it is currently as close as possible to EPAL. Further, we do not abstract from conditions in contrast to [2] so that we can define a semantics for incomplete context data, which is useful both in general practice and specifically for refinements and composition of policies from different realms. In spite of the current closeness to EPAL, we continue to call the abstract language E-P3P as in [2] to avoid confusion with possible changes to EPAL.

The first tool we define is policy refinement. Intuitively, one policy refines another if using the first policy automatically also fulfills the second policy. It is thus the fundamental notion for many situations in policy management. For instance, it enables verification that an enterprise policy fulfills regulations or adheres to standards set by consumer organizations or a self-regulatory body, assuming only that these coarser requirements are once and for all also formalized as a privacy policy. Similarly, it enables verification that a detailed policy for a part of the enterprise (defined by responsibility or by technology) refines the overall privacy policy set by the company's CPO. The verification can be done in the enterprise or by external auditors, such as [21].

When a policy is first designed, refinement may be achieved in a constructive way, e.g., by starting with the coarse policy and only adding details by certain provably refining syntactic means. However, if a regulation changes or the enterprise extends its operation to new sectors or countries, the enterprise has to verify that its existing policy still complies with the new or additional regulations. Hence a definition of refinement between two arbitrary policies is needed. Sticky policies are another application of general refinement: Here data are transferred from the realm of one policy into another (where the transfer must of course be permitted by the first policy), and the second realm must enforce the first policy. However, the enforcement mechanisms (both organizational and technical) in the second realm will often not be able to deal with arbitrary policies for each obtained set of data. In this case, one realm must perform a refinement test before the data are transferred, i.e., one has to verify that the policy of the second realm refines the policy of the first, at least for the restriction of the first policy to the data types being transferred.

Composition is the notion of constructively combining two or more policies; typically the goal is that the resulting policy refines them all. For instance, an enterprise might first take all applicable regulations and combine them into a minimum policy. A general promise made to customers, e.g., an existing P3P translated into the more general language, may be a further input. In enterprise parts that support detailed preferences of individuals, such preferences may be yet another policy to be composed with the others, yielding one final policy per individual. (In contrast, simple preferences may be represented as a set of Boolean opt-in or opt-out choices, and treated as context data by conditions within a single policy.) Typical applications where detailed preferences are needed are wallet-style collections of user data for the purpose of transfer to other enterprises, and collaborative tools such as team-rooms.

Composition is not a simple logical AND for powerful enterprise privacy policies as in EPAL, e.g., because of the treatment of obligations, different policy scopes, and default values. Moreover, refinement and composition turn up two basic questions about the meaning of a privacy policy, which are not answered by the abstract semantics of an individual policy. The first question is the meaning of a positive ruling in privacy policies. Intuitively, negative rulings are understood to be definite; e.g., if a policy states that certain data are not used for email marketing, then no such email marketing should happen. The intuition is different for most positive rulings: If a policy allows third-party email marketing, it is typically not seen as a promise to actually do marketing, neither to the owners of the email addresses nor to the third parties. However, if one decides to represent access rights for data subjects to their data, such as the right to see all their data or to correct mistakes, with the normal policy mechanisms, then these positive rulings must be mandatory. The second question is related: If a privacy policy, like EPAL, is formulated with precedences to enable easy formulations of positive and negative exceptions, then within a policy, neither negative nor positive rules are "final", i.e., can be considered isolated from the policy. In contrast, in compositions, one may want to retain an entire original policy as final. We solve both these problems by allowing mandatory sub-policies. This allows us to distinguish final decisions from decisions that may be overturned by other rules, and thus to represent all the cases just discussed. We extend the notion of composition and refinement to these two-part policies.

Further Related Literature. The core contribution of new privacy-policy languages [10, 17, 16], compared with other access-control languages, is the notion of purpose and purpose-bound collection of data, which is essential to privacy legislation. Other necessary features that prevent enterprises from simply using their existing access-control systems are obligations and conditions on context information. Individually, these features were also considered in recent literature on access control, e.g., purpose hierarchies in [5], obligations in [4, 14, 20], and conditions on context information in [22]. However, we need them all in one language, and even for the individual features the detailed semantics needed in practice, such as with multiple terminologies, typically does not exist yet, and thus nor does a comparable toolkit. Policy composition has been treated before, in particular for access control [6, 7, 9, 13, 15, 22], systems management [18], or IPSEC [11]; however none of these papers does it for the general policies we need and several do not have a clear underlying semantics. The publications closest to our treatment of incomplete data are those on information-disclosure-minimal negotiation of access-control policies, e.g., [3, 12].

2 Syntax and Semantics of E-P3P Enterprise Privacy Policies

Privacy policies define the purposes for which collected data can be used, model the consent a data subject can give, and may impose obligations onto the enterprise. They can formalize privacy statements like “we use data of a minor for marketing purposes only if the parent has given consent” or “medical data can only be read by the patient’s primary care physician”. In this section, we present the abstract syntax and semantics E-P3P of IBM’s EPAL privacy policy language [1]. Compared with [2], we abstract less from conditions and obligations, so that we can present a more detailed semantics.

2.1 Hierarchies, Obligations, and Conditions

We start by defining the models of hierarchies, obligations, and conditions used in E-P3P, and operations on them as needed in later refinements and compositions.

For conveniently specifying rules, the data, users, etc. are categorized in E-P3P as in many access-control languages. This also applies to the purposes. In order to allow structured rules with exceptions, categories are ordered in hierarchies; mathematically they are forests, i.e., multiple trees. For instance a user “company” may group several “departments”, each containing several “employees”. The enterprise can then write rules for the whole “company” with exceptions for some “departments”.

Definition 1 (Hierarchy). *A* hierarchy *is pair $(H, >_H)$ of a finite set H and a transitive, non-reflexive relation $>_H \subseteq H \times H$, where every $h \in H$ has at most one immediate predecessor (parent). As usual we write $\geq_H$ for the reflexive closure.*

For two hierarchies $(H, >_H)$ and $(G, >_G)$, we define

$$(H, >_H) \subseteq (G, >_G) :\Leftrightarrow (H \subseteq G) \wedge (>_H \subseteq >_G);$$
$$(H, >_H) \cup (G, >_G) := (H \cup G, (>_H \cup >_G)^*);$$

where * denotes the transitive closure. Note that a hierarchy union is not always a hierarchy again. ◇

E-P3P policies can impose obligations, i.e., duties for the enterprise. Examples are to send a notification to the data subject after each emergency access to medical data, or to delete data after a given time. Obligations are not structured in hierarchies, but by an implication relation. For instance, an obligation to delete data within 30 days implies that the data are deleted within 60 days. The overall obligations for a rule in E-P3P are written as sets of individual obligations, which must have an interpretation in the application domain. As multiple obligations may imply more than each one individually, we define the implication (which must also be realized in the application domain) on these sets. We also define how this relation interacts with vocabulary extensions.

Definition 2 (Obligation Model). *An* obligation model *is a pair $(O, \rightarrow_O)$ of a set O and a relation $\rightarrow_O \subseteq \mathfrak{P}(O) \times \mathfrak{P}(O)$, spoken* implies*, on the powerset of O, where $\bar{o}_1 \rightarrow_O \bar{o}_2$ for all $\bar{o}_2 \subseteq \bar{o}_1$, i.e., fulfilling a set of obligations implies fulfilling all subsets.*

For $O' \supset \mathfrak{P}(O)$, we extend the implication to $O' \times \mathfrak{P}(O)$ by $((\bar{o}_1 \rightarrow_O \bar{o}_2) :\Leftrightarrow (\bar{o}_1 \cap \mathfrak{P}(O) \rightarrow_O \bar{o}_2))$. $\diamond$

The decision formalized by a privacy policy can depend on context data. Examples are a person's age or opt-in consent. In EPAL this is represented by conditions over data in so-called containers [1]. The XML representation of the formulas is taken from [22], which corresponds to a predicate logic without quantifiers. In the abstract syntax in [2], conditions are abstracted into propositional logic, but this is too coarse for our purposes. Hence we extend E-P3P to be closer to EPAL by formalizing the containers as a set of variables with domains, and the conditions as formulas over these variables.

Definition 3 (Condition Vocabulary). *A* condition vocabulary *is a pair $Var = (V, Scope)$ of a finite set V and a function assigning every $x \in V$, called a* variable*, a set $Scope(x)$, called its* scope.

Two condition vocabularies $Var_1 = (V_1, Scope_1)$, $Var_2 = (V_2, Scope_2)$ are compatible *if $Scope_1(x) = Scope_2(x)$ for all $x \in V_1 \cap V_2$. For that case, we define their* union *by $Var_1 \cup Var_2 := (V_1 \cup V_2, Scope_1 \cup Scope_2)$.* $\diamond$

In the future, one might extend this to a full signature in the sense of logic, i.e., including predicate and function symbols. In EPAL, this is hidden in user-defined functions that may occur in the XACML conditions. For the moment, we assume a given universe of predicates and functions with fixed domains and semantics.

Definition 4 (Condition Language). *Let a condition vocabulary $Var = (V, Scope)$ be given.*

- *The* condition language *$C(Var)$ is the set of correctly typed formulas over V using the assumed universe of predicates and functions, and in the given syntax of predicate logic without quantifiers.*
- *The free variables of a formula $c \in C(Var)$ are denoted by $free(c)$. Here these are all variables of c.*
- *A (partial)* assignment *of the variables is a (partial) function $\chi \colon V \rightarrow \bigcup_{x \in V} Scope(x)$ with $\chi(x) \in Scope(x)$ for all $x \in V$. The set of all assignments for the set Var is written $\mathfrak{Ass}(Var)$; that of all partial assignments $\mathfrak{Ass}_{\subseteq}(Var)$.*
- *For $\chi \in \mathfrak{Ass}(Var)$, let $eval_\chi \colon C(Var) \rightarrow \{true, false\}$ denote the evaluation function for conditions given this variable assignment. This is defined by the underlying logic and the assumption that all predicate and function symbols come with a fixed semantics.* $\diamond$

An important aspect of our semantics is the ability to deal meaningfully with underspecified requests. This means that a condition might not be evaluatable since only a subset of the variables used in the conditions has been assigned. This is important not only in federated scenarios as introduced in [9], but also for overall in-enterprise policies. For instance, some rules of a policy may need the age of a person or its employee role, while for many people no age or employee role is known in the enterprise. This typically does no harm because other rules apply to these persons. For such situations, we will need to know whether a condition can still become $true$ or $false$, respectively, when a partial assignment is extended. Hence we define extensions.

Definition 5 (Extension of partial assignments). *Let a condition vocabulary* $Var = (V, Scope)$ *be given. If* $\chi \in \mathfrak{Ass}_{\subseteq}(Var)$ *is defined on* $U \subseteq V$*, let*

$$Ext(\chi, Var) := \{\chi^* \in \mathfrak{Ass}(Var) \mid \forall u \in U\colon \chi^*(u) = \chi(u)\}$$

denote the set of extensions *of* χ. ◇

2.2 Syntax of E-P3P Policies

An E-P3P policy is a triple of a vocabulary, a set of authorization rules, and a default ruling. The vocabulary defines element hierarchies for data, purposes, users, and actions, as well as the obligation model and the condition vocabulary. Data, users and actions are as in most access-control policies (except that users are typically called "subjects" there, which in privacy would lead to confusion with data subjects), and purposes are an important additional hierarchy for the purpose binding of collected data.

Definition 6 (Vocabulary). *A* vocabulary *is a tuple* $Voc = (UH, DH, PH, AH, Var, OM)$ *where* UH*,* DH*,* PH*, and* AH *are hierarchies called user, data, purpose, and action hierarchy, respectively, and* Var *is a condition vocabulary and* OM *an obligation model.* ◇

As a naming convention, we assume that the components of a vocabulary called Voc are always called as in Definition 6 with $UH = (U, >_U)$, $DH = (D, >_D)$, $PH = (P, >_P)$, $AH = (A, >_A)$, $Var = (V, Scope)$, and $OM = (O, \rightarrow_O)$, except if explicitly stated otherwise. In a vocabulary called Voc_i all components also get a subscript i, and similarly for superscripts. A rule set contains authorization rules that allow or deny operations. A rule basically consists of one element from each vocabulary component. Additionally, it starts with an integer precedence, and ends with a ruling.

Definition 7 (Ruleset and Privacy Policy). *A* ruleset *for a vocabulary* Voc *is a subset of* $\mathbb{Z} \times U \times D \times P \times A \times C(Var) \times \mathfrak{P}(O) \times \{+, \circ, -\}$.

A privacy policy *or* E-P3P policy *is a triple* (Voc, R, dr) *of a vocabulary* Voc*, a rule-set* R *for* Voc*, and a* default ruling $dr \in \{+, \circ, -\}$*. The set of these policies is called* $EP3P$*, and the subset for a given vocabulary* $EP3P(Voc)$. ◇

In EPAL, precedences are only given implicitly by the textual order of the rules. Hence our explicit precedences, and the fact that several rules can have the same precedence, make E-P3P a superset of EPAL. The rulings $+$, $\circ$, and $-$ mean 'allow', 'don't care', and 'deny'. The ruling $\circ$ was not yet present in [2]. In EPAL, it is called 'obligate' because it enables rules that do not make a decision, but only impose additional obligations. An example is a global rule "Whenever someone tries to access my data, I want to receive a notification".

As a naming convention, we assume that the components of a privacy policy called Pol are always called as in Definition 7, and if Pol has a sub- or superscript, then so do the components.

2.3 Semantics of E-P3P Policies

An E-P3P request is a tuple (u, d, p, a) which should belong to the set $U \times D \times P \times A$ for the given vocabulary. Note that E-P3P and EPAL requests are not restricted to "ground terms" as in some other languages, i.e., minimal elements in the hierarchies. This is useful if one starts with coarse policies and refines them because elements that are initially minimal may later get children. For instance, the individual users in a "department" of an "enterprise" may not be mentioned in the CPO's privacy policy, but in the department privacy policy. For similar reasons, we also define the semantics for requests outside the given vocabulary. We assume a superset $\mathcal{S}$ in which all hierarchy sets are embedded; in practice it is typically a set of strings or valid XML expressions.

Definition 8 (Request). *For a vocabulary Voc, we define the set of* valid requests *as* $Req(Voc) := U \times D \times P \times A$. *Given a superset* $\mathcal{S}$ *of the sets* U, D, P, A *of all considered vocabularies, the set of* all requests *is* $Req := \mathcal{S}^4$. ◇

The semantics of a privacy policy Pol is a function $eval_{Pol}$ that processes a request based on a given, possibly partial, assignment.

The evaluation result is a pair $(r, \bar{o})$ of a ruling (decision) and associated obligations. Our semantics extends that of [2] in three ways. First, we have to deal with the new partial assignments in the conditions of rules. Secondly, the ruling $\circ$ that was added to the rule syntax gets a semantics; as explained above it is used to make obligations without enforcing a decision. Thirdly, the ruling r may not only be $+$, $\circ$, or $-$ as in a rule, but also $scope\_error$ or $conflict\_error$. This denotes that the request was out of scope of the policy or that there was a conflict among applicable rules. The reason for distinguishing these errors is that out-of-scope errors can be eliminated by enlarging the policy, in contrast to conflict errors. This will become important for policy composition.

The semantics is defined by a virtual pre-processing that unfolds the hierarchies and a request processing stage. Note that this is only a compact definition of the semantics and not an efficient real evaluation algorithm.

Definition 9 (Unfolded Rules). *For a privacy policy* $Pol = (Voc, R, dr)$, *the* unfolded rule set $UR(Pol)$ *is defined as follows:*

$$\begin{aligned} URdown(Pol) &:= \{(i, u', d', p', a', c, \bar{o}, r) \mid \exists (i, u, d, p, a, c, \bar{o}, r) \in R \\ &\qquad\qquad \text{with } u \geq_U u' \wedge d \geq_D d' \wedge p \geq_P p' \wedge a \geq_A a'\}; \\ UR(Pol) &:= URdown(Pol) \\ &\quad \cup \{(i, u', d', p', a', c, \bar{o}, -) \mid \exists (i, u, d, p, a, c, \bar{o}, -) \in URdown(Pol) \\ &\qquad\qquad \text{with } u' \geq_U u \wedge d' \geq_D d \wedge p' \geq_P p \wedge a' \geq_A a\}. \end{aligned}$$
◇

Note that 'deny'-rules are inherited both downwards and upwards along the four hierarchies while 'allow'-rules are inherited only downwards. The reason is that the hierarchies are considered groupings; if access is forbidden to an element of a group, it is also forbidden for the group as a whole.

Next we define which rules are applicable for a request given a partial assignment of the condition variables. These (unfolded) rules have the user, data, purpose, and action as in the request. Positive rules are only defined to be applicable if they evaluate to $true$ for all extensions of the partial assignment χ. Negative and don't-care rules are defined to be applicable whenever the conditions could still become true. For instance, if a rule forbids access to certain data for minors, a child should not be able to obtain access by omitting its age, and obligations from don't-care rules for children should apply.

Definition 10 (Applicable Rules). *Let a privacy policy $Pol = (Voc, R, dr)$, a request $q = (u, d, p, a) \in Req(Voc)$, and a partial assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var)$ be given. Then the set of* applicable rules *is*

$$\begin{aligned} &AR(Pol, q, \chi) := \\ &\{(i, u, d, p, a, c, \bar{o}, +) \in UR(Pol) \mid \forall \chi^* \in Ext(\chi, Var)\colon eval_{\chi^*}(c) = true\} \\ \cup\ &\{(i, u, d, p, a, c, \bar{o}, r) \in UR(Pol) \mid r \in \{-, \circ\} \wedge \\ &\quad \exists \chi^* \in Ext(\chi, Var)\colon eval_{\chi^*}(c) = true\}. \end{aligned}$$

$\diamond$

For formulating the semantics, we need the maximum and minimum precedence in a policy.

Definition 11 (Precedence Range). *For a privacy policy $Pol = (Voc, R, dr)$, let $max(Pol) := \max\{i \mid \exists(i, u, d, p, a, c, \bar{o}, r) \in R\}$, and similarly $min(Pol)$.* $\diamond$

We can now define the actual semantics, i.e., the result of a request given a partial assignment. Recall that rules with ruling $\circ$ are provided to allow obligations to accumulate before the final decision; this is done in a set $\bar{o}_{acc}$.

Definition 12 (Semantics). *Let a privacy policy $Pol = (Voc, R, dr)$, a request $q = (u, d, p, a) \in Req$, and a partial assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var)$ be given. Then the* evaluation result $(r, \bar{o}) := eval_{Pol}(q, \chi)$ *of policy Pol for q and χ is defined by the following algorithm, starting with $\bar{o}_{acc} := \emptyset$. Every "return" aborts the algorithm.*

– Out-of-scope testing. *If $q \notin Req(Voc)$, return $(r, \bar{o}) := (scope\_error, \emptyset)$.*
– Processing by precedence. *For each precedence level $i := max(Pol)$ down to $min(Pol)$:*
  - Accumulate obligations. *For each applicable rule $(i, u, d, p, a, c, \bar{o}', r) \in AR(Pol, q, \chi)$, set $\bar{o}_{acc} := \bar{o}_{acc} \cup \bar{o}'$.*
  - Conflict detection. *If two conflicting rules $(i, u, d, p, a, c_1, \bar{o}_1, +)$ and $(i, u, d, p, a, c_2, \bar{o}_2, -)$ exist in $AR(Pol, q, \chi)$, return $(conflict\_error, \emptyset)$.*
  - Normal ruling. *If at least one rule $(i, u, d, p, a, c, \bar{o}', r) \in AR(Pol, q, \chi)$ with $r \neq \circ$ exists, return $(r, \bar{o}_{acc})$.*
– Default ruling. *If this step is reached, return $(r, \bar{o}) := (dr, \bar{o}_{acc})$.*

We also say that policy Pol rules *$(r, \bar{o})$ for q and χ, omitting q and χ if they are clear from the context.* $\diamond$

3 Refinement of Privacy Policies

In this section, we define the notion of refinement for E-P3P policies. As explained in the introduction, refinement is the foundation of almost all operations on policies. We further define policy equivalence and show that it equals mutual refinement.

Refining a policy Pol_1 means adding more details, both rules and vocabulary, while retaining its meaning with respect to the original vocabulary. Our notion of refinement allows policy Pol_2 to define a ruling if Pol_1 does not care. Additionally, it is allowed to extend the scope of the original policy and to define arbitrary rules for the new elements. In all other cases, the rulings of both policies must be identical. This also comprises the ruling $\mathit{conflict\_error}$. For new elements however, we have to capture that if they are appended to the existing hierarchies, there could exist applicable rules for these elements if they were already present, and newly added rules for these elements could influence existing elements as well. As an example, a rule for a "department" may forbid its "employees" to access certain data for marketing purposes. Now if a new employee is added, this rule should as well be applicable; furthermore, defining a new rule for this case with higher precedence, e.g., granting the new employee an exception to the department's rule should obviously not yield a refinement any more. In our definition of refinement, we therefore do not evaluate each policy on its own vocabulary but on the joint vocabulary of both policies. Since joining two vocabularies, i.e., joining their respective hierarchies, might not yield another vocabulary, we introduce the notion of compatible vocabularies.

Definition 13 (Compatible Vocabulary). *Two vocabularies Voc_1 and Voc_2 are* compatible *if their condition vocabularies are compatible and all hierarchy unions $UH_1 \cup UH_2$, $DH_1 \cup DH_2$, $PH_1 \cup PH_2$, and $AH_1 \cup AH_2$ are hierarchies again.*

We define the union *of two compatible vocabularies as $Voc_1 \cup Voc_2 := (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_1 \cup Var_2, OM_1 \cup OM_2)$.* ◇

Dealing with the respective obligations is somewhat more difficult. Intuitively, one wants to express that a finer policy may also contain refined obligations. However, since a refined policy might contain additional obligations, whereas some others have been omitted, it is not possible to simply compare these obligations in the obligation model of the original policy. (Recall that we also use refinement to compare arbitrary policies; hence one cannot simply expect that all vocabulary parts of the refined policy are supersets of those of the coarser policy.)

As an example, let the obligation model of the coarser policy contain obligations $o =$ "delete in a week" and o_1 = "delete in a month" with the implication $o \rightarrow_{O_1} o_1$. The refined policy contains o_2 = "delete immediately" and o as above with $o_2 \rightarrow_{O_2} o$. Now o_2 should be a refinement of o_1, but this cannot be deduced in either of the obligation models. Hence both obligation models have to be used, i.e., one has $o_2 \rightarrow_{O_2} o \rightarrow_{O_1} o_1$. We define this as *obligation refinement*. In order to obtain a meaningful refinement from the point of view of Pol_1, the relation $\rightarrow_{O_2}$ has to be certified by a party trusted by the maintainer of Pol_1.

Definition 14 (Obligation Refinement). *Let two obligation models $(O_i, \rightarrow_{O_i})$ and $\bar{o}_i \subseteq O_i$ for $i = 1, 2$ be given. Then $\bar{o}_2$ is a* refinement *of $\bar{o}_1$, written $\bar{o}_2 \prec \bar{o}_1$, iff the following holds:*

$$\exists \bar{o} \subseteq O_1 \cap O_2 \colon \bar{o}_2 \rightarrow_{O_2} \bar{o} \rightarrow_{O_1} \bar{o}_1. \qquad \diamond$$

We are now ready to introduce our notion of policy refinement.

Definition 15 (Policy Refinement). *Let two privacy policies $Pol_i = (Voc_i, R_i, dr_i)$ for $i = 1, 2$ with compatible vocabularies be given, and set $Pol_i^* = (Voc_i^*, R_i, dr_i)$ for $i = 1, 2$, where $Voc_i^* = (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_i, OM_i)$. Then Pol_2 is a* refinement *of Pol_1, written $Pol_2 \prec Pol_1$, iff for every assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$ and every authorization request $q \in Req$ one of the following statements holds, where $(r_i, \bar{o}_i) := eval_{Pol_i^*}(q, \chi)$ for $i := 1, 2$:*

- $(r_1, \bar{o}_1) = (r_2, \bar{o}_2) = (\mathit{conflict\_error}, \emptyset)$.
- $(r_1, \bar{o}_1) = (\mathit{scope\_error}, \emptyset)$.
- $r_1 \in \{+, -\}$ *and* $r_2 = r_1$ *and* $\bar{o}_2 \prec \bar{o}_1$.
- $r_1 = \circ$ *and* $r_2 \in \{+, \circ, -\}$ *and* $\bar{o}_2 \prec \bar{o}_1$. $\diamond$

Besides this rather strict notion of refinement, we can also define a notion of *weak refinement*, denoted by $\tilde{\prec}$, where the refining policy may be less permissive than the original policy. The only difference to Definition 15 is that $+$ is treated like $\circ$ in the fourth statement instead of like $-$ in the third statement. Weak refinement corresponds to the intuition that a policy Pol_1 implements a privacy promise or requirement to use data at most for certain purposes, so that a refining policy Pol_2 can only restrict that usage. However, while weak definition prevents misuse, it does not preserve guaranteed access rights: For instance, Pol_1 may guarantee an individual the right to read her data while policy Pol_2 does not. Strong refinement therefore seems the more useful notion for E-P3P with its 3-valued logic where $\circ$, meaning 'don't-care', is also a valid ruling. In contrast one might choose weak refinement for a 2-valued policy language with only the rulings $+$ and $-$. We therefore concentrate on strong refinement.

After refinement, we now introduce a notion of equivalence of policies. Similar to policy refinement, we start with the equivalence of obligations.

Definition 16 (Obligation Equivalence). *Let two obligation models $(O_i, \rightarrow_{O_i})$ and $\bar{o}_i \subseteq O_i$ for $i = 1, 2$ be given. Then $\bar{o}_1$ and $\bar{o}_2$ are* equivalent*, written $\bar{o}_1 \equiv \bar{o}_2$, iff $\bar{o}_1 \prec \bar{o}_2$ and $\bar{o}_2 \prec \bar{o}_1$.* $\diamond$

The relation $\equiv$ is clearly symmetric.

Definition 17 (Policy Equivalence). *Two privacy policies Pol_1 and Pol_2 with compatible vocabularies are* equivalent*, written $Pol_1 \equiv Pol_2$, iff for every assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$ and every request $q \in Req$ we have*

$$r_1 = r_2 \text{ and } \bar{o}_1 \equiv \bar{o}_2$$

for the evaluation results $(r_1, \bar{o}_1) := eval_{Pol_1}(q, \chi)$ and $(r_2, \bar{o}_2) := eval_{Pol_2}(q, \chi)$. $\diamond$

Clearly, policy equivalence is a symmetric relation, since obligation equivalence is symmetric. We can now establish the following theorem.

Theorem 1. *Two privacy policies Pol_1, Pol_2 are equivalent if and only if they are mutual refinements. Formally,*

$$Pol_1 \equiv Pol_2 \Leftrightarrow Pol_1 \prec Pol_2 \wedge Pol_2 \prec Pol_1. \qquad \square$$

Proof. Let an assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$ and an authorization request $q \in Req$ be given. Note that $Pol_1 \equiv Pol_2$ implies $Req(Voc_1) = Req(Voc_2)$, as otherwise there exists $q \in Req(Voc_1) \setminus Req(Voc_2)$ without loss of generality such that $eval_{Pol_2}(q, \chi) = (scope\_error, \emptyset) \neq eval_{Pol_1}(q, \chi)$. Similarly, we can show that $Pol_1 \prec Pol_2 \wedge Pol_2 \prec Pol_1$ implies $Req(Voc_1) = Req(Voc_2)$, as for $q \in Req(Voc_1) \setminus Req(Voc_2)$, we have $eval_{Pol_2}(q, \chi) = (scope\_error, \emptyset) \neq eval_{Pol_1}(q, \chi)$, which contradicts $Pol_2 \prec Pol_1$. Therefore, we have $Pol_i = Pol_i^*$ for $i = 1, 2$, with Pol_i^* as in Definition 15, i.e., we can consider the evaluation of Pol_i instead of Pol_i^* to show refinement. Hence let $(r_i, \bar{o}_i) := eval_{Pol_i}(q, \chi) = eval_{Pol_i^*}(q, \chi)$ for $i = 1, 2$ be the corresponding rulings.

"$\Rightarrow$" Since policy equivalence is symmetric, it is sufficient to show that Pol_2 refines Pol_1. If $(r_1, \bar{o}_1) = (conflict\_error, \emptyset)$ then also $(r_2, \bar{o}_2) = (conflict\_error, \emptyset)$ because $Pol_1 \equiv Pol_2$. If $(r_1, \bar{o}_1) = (scope\_error, \emptyset)$, nothing has to be shown. Now let $r_1 \in \{+, \circ, -\}$. Policy equivalence implies $r_2 = r_1$ and $\bar{o}_2 \equiv \bar{o}_1$; this is sufficient for refinement.

"$\Leftarrow$" We distinguish the following cases:

$(r_1, \bar{o}_1) = (conflict\_error, \emptyset)$. Then we also have $(r_2, \bar{o}_2) = (conflict\_error, \emptyset)$ since Pol_2 refines Pol_1. This implies $\bar{o}_1 \equiv \bar{o}_2$.

$(r_1, \bar{o}_1) = (scope\_error, \emptyset)$. If $r_2 \neq r_1$ we immediately obtain that Pol_1 is not a refinement of Pol_2. Thus $r_2 = scope\_error$. This implies $\bar{o}_2 = \emptyset$ and thus $\bar{o}_2 \equiv \bar{o}_1$.

$r_1 \in \{+, -\}$. Then $r_2 = r_1$ since Pol_2 refines Pol_1. Further, since Pol_1 and Pol_2 are mutual refinements, we have $\bar{o}_1 \prec \bar{o}_2$ and $\bar{o}_2 \prec \bar{o}_1$ and thus $\bar{o}_1 \equiv \bar{o}_2$.

$r_1 = \circ$. Assume for contradiction that $r_2 \in \{+, -, scope\_error, conflict\_error\}$. In this case Pol_1 is no refinement of Pol_2 any longer. Further, as in the previous case, we have $\bar{o}_1 \prec \bar{o}_2$ and $\bar{o}_2 \prec \bar{o}_1$ and thus $\bar{o}_1 \equiv \bar{o}_2$. ■

4 Composition of Privacy Policies

In this section, we introduce two notions of composition of E-P3P policies, i.e., the merging of two somehow compatible policies.

In an enterprise, policies may be defined on multiple levels in a management hierarchy. A chief privacy officer (CPO) may define enterprise-wide mandatory policy rules that implement the applicable privacy laws. In addition, the CPO can define defaults that apply if a department does not define its own rules. A department can then define its own privacy policy rules. These rules override the default rules but are overruled by the mandatory rules of the CPO. In order to allow such distributed authoring and maintenance of privacy policies, we now introduce a notion of policy composition. If two policies are composed, both rule-sets are enforced. By defining that one policy has a higher precedence than the other, one can define one way to resolve conflicts. For such precedence shifts and for dealing with default values, we start with the notion of the *normalization* of a policy.

4.1 Policy Normalization

Recall that the default ruling of a policy determines the result if no rule applies for a given request although the request is in the scope of the policy. When composing policies, different default rulings must be resolved first. This is simple if the scope is the same or the default ruling is the same. To resolve the more challenging cases, we first convert the default ruling of a policy into a set of normal rules. These new rules have the default ruling as their ruling, lowest precedence, no obligations and conditions, and they cover the root elements of all hierarchies.

Definition 18 (Policy with Removed Default Ruling). *Let* $Pol = (Voc, R, dr)$ *be a privacy policy and* $\underline{i} \in \mathbb{Z}$*. Then the* policy with removed default ruling *for* Pol *wrt.* $\underline{i}$ *is the following policy* $rmDR(Pol, \underline{i})$*:*

If $dr = \circ$*, then* $rmDR(Pol, \underline{i}) := Pol$*.*

Else for every hierarchy $XH = (H, >_H)$*, let* $roots(XH) := \{x \in H \mid \neg\exists x' \in H : x' >_H x\}$*. Then* $rmDR(Pol, \underline{i}) := (Voc, R', \circ)$ *with* $R' := R \cup DR$ *and*

$$DR := \{(\underline{i}, u, d, p, a, \emptyset, \emptyset, dr) \mid u \in roots(UH) \wedge d \in roots(DH) \\ \wedge p \in roots(PH) \wedge a \in roots(AH)\}.$$

We abbreviate $rmDR(Pol) := rmDR(Pol, min(Pol) - 1)$*.* $\diamond$

We now show that a policy with removed default ruling is equivalent to the original policy if $\underline{i}$ is smaller than all the precedences in the original policy.

Lemma 1. *Let* $Pol = (Voc, R, dr)$ *be a privacy policy and* $\underline{i} \in \mathbb{Z}$ *with* $\underline{i} < min(Pol)$*. Then* $rmDR(Pol, \underline{i}) \equiv Pol$*. In particular, this implies* $rmDR(Pol) \equiv Pol$*.* $\square$

Proof. Let a request $q \in Req$ and an assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var)$ be given. Since Pol and $rmDR(Pol)$ have the same vocabulary, we can show refinement using the evaluation of Pol and $rmDR(Pol)$ instead of Pol^* and $rmDR(Pol)^*$ as defined in Definition 15. Equal vocabularies also imply that either both policies rule $(scope\_error, \emptyset)$ or none of them.

If Pol rules $(conflict\_error, \emptyset)$ then so does $rmDR(Pol, \underline{i})$, since the rules in DR have lower precedence than all rules in R. Furthermore, all rules in DR have identical ruling; hence they cannot induce a conflict error. Thus either both policies rule $(conflict\_error, \emptyset)$ or none of them.

If a rule ρ from R applies to this request, it applies for both policies, since every rule of Pol is also contained in the ruleset of $rmDR(Pol, \underline{i})$. Moreover, every rule in DR has lower priority than ρ by construction. Hence neither a rule in DR nor the default ruling applies. Thus Pol and $rmDR(Pol, \underline{i})$ output the same pair $(r, \bar{o})$. Conversely, if a rule $\rho \in DR$ applies, this means that no rule of R applies, but the request is in scope of the policy. In this case Pol outputs $(dr, \bar{o}_{acc})$, where $\bar{o}_{acc}$ is the set of obligations accumulated while processing R. The policy $rmDR(Pol, \underline{i})$ applies the rule ρ, which also yields $(dr, \bar{o}_{acc})$ since no obligation is added by any rule in DR. ■

Next, we introduce an operation for changing the precedence of the rules of a policy, e.g., to overcome possible conflicts when merging the policy with another one. As a collective change for all rules seems useful, we define a precedence shift, which adds a fixed number to the precedence of all rules in a policy. This is particularly useful for the example at the beginning of this section, where the department policy can be shifted downwards to have lower precedences than the policy of the CPO.

Definition 19 (Precedence Shift). *Let $Pol = (Voc, R, dr)$ be a privacy policy and $j \in \mathbb{Z}$. Then $Pol + j := (Voc, R + j, dr)$ with $R + j := \{(i + j, u, d, p, a, c, \bar{o}, r) \mid (i, u, d, p, a, c, \bar{o}, r) \in R\}$ is called the* precedence shift *of Pol by j. We define $Pol - j := Pol + (-j)$.* ◇

Lemma 2. *A privacy policy Pol is equivalent to $Pol + j$ for all $j \in \mathbb{Z}$.* □

Proof. By Theorem 1, it is sufficient to show that Pol refines $Pol + j$ for all $j \in \mathbb{Z}$.

Let j, a request $q \in Req$, and an assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var)$ be given. By construction, the rules that apply to this request in Pol and $Pol + j$ can only differ in their precedence. Furthermore, if a rule applies in Pol, it also applies after the precedence shift, since the precedence of *all* rules in $Pol + j$ has been shifted similarly. Applying the same rule in both policies in particular yields the same set of obligations. Hence Pol refines $Pol + j$. ■

We now define the normalization of a policy. This corresponds to a precedence shift yielding a default ruling of precedence 0. Normalized policies are used in the definition of the ordered composition of two policies defined below.

Definition 20 (Normalized Policy). *Let Pol be a privacy policy. Then* $\mathrm{norm}_0(Pol) := rmDR(Pol - min(Pol) + 1, 0)$ *is called the* normalized policy *for Pol.* ◇

Lemma 3. *For every privacy policy Pol, we have* $\mathrm{norm}_0(Pol) \equiv Pol$. □

Proof. By definition, the precedence of all rules in $\mathrm{norm}_0(Pol)$ is greater than 0. Hence the claim follows from Lemmas 1 and 2. ■

4.2 Definition of Composition

We now have the tools ready to turn our attention to policy composition. The simplest case of merging two policies with compatible vocabularies is to compute the union of the two rule sets. This direct composition assumes that the precedences used in both policies have a common meaning. However, translating the default ruling of a policy is tricky: If elements are in the scope of only one policy, the default ruling of this policy should apply. If elements are covered by both policies, a conflict between the two default rulings needs to be resolved.

Definition 21 (Direct Composition). *Let Pol_1 and Pol_2 be two privacy policies with compatible vocabularies. Let $m := \min\{min(Pol_1), min(Pol_2)\}$ and $Pol'_i := (Voc_i, R'_i, \circ) := rmDR(Pol_i, m)$ for $i = 1, 2$. Then*

$$Pol_1 \bigcup Pol_2 := (Voc_1 \cup Voc_2, R'_1 \cup R'_2, \circ)$$

is called the direct composition *of Pol_1 and Pol_2.* ◇

In our second type of composition, the rules of one policy, here Pol_2 should be applied preferably. Hence the rules of the other policy are downgraded using a precedence shift. This also applies to the default ruling of the preferred policy, i.e., the downgraded policy is only used where it extends the scope of the preferred policy, or if no rule of the preferred policy applies and the default ruling is $\circ$.

Definition 22 (Ordered Composition). *Let Pol_1 and Pol_2 be two privacy policies with compatible vocabularies. Let $(Voc_1, R''_1, \circ) := rmDR(Pol_1 - max(Pol_1) - 1)$, and $(Voc_2, R'_2, \circ) := \mathrm{norm}_0(Pol_2)$. Then*

$$Pol_1 \overset{<}{\bigcup} Pol_2 := (Voc_1 \cup Voc_2, R''_1 \cup R'_2, \circ)$$

is called the ordered composition *of Pol_1 under Pol_2.* ◇

By the intuitive introduction to ordered compositions, an ordered composition should serve as a refinement of Pol_2. This is captured in the following lemma.

Lemma 4. *For all privacy policies Pol_1 and Pol_2 with compatible vocabularies, we have $Pol_1 \overset{<}{\cup} Pol_2 \prec Pol_2$.* □

Proof. Let $Pol_i =: (Voc_i, R_i, dr_i)$ for $i := 1, 2$, and we use all notation from Definition 22. Let a request $q = (u, d, p, a)$ and an assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$ be given. Let Pol_2^* and $(Pol_1 \overset{<}{\cup} Pol_2)^*$ be defined according to Definition 15. Note further that $(Pol_1 \overset{<}{\cup} Pol_2)^* = Pol_1 \overset{<}{\cup} Pol_2$ since their vocabularies are equal. We distinguish four cases:

1. Pol_2^* rules $(conflict\_error, \emptyset)$: In this case, two rules of the same precedence i collided in Pol_2^*. Since i is larger than the precedence of any rule of R''_1 by the shifts, we also get a conflict in $Pol_1 \overset{<}{\cup} Pol_2$, and an output $(conflict\_error, \emptyset)$.
2. Pol_2^* rules $(scope\_error, \emptyset)$: Nothing has to be shown for this case.
3. Pol_2^* rules $(r_2, \bar{o}_2)$ with $r_2 \neq \circ$: This means that a rule $(i_2, u, d, p, a, c_2, \bar{o}_2, r_2)$ from R'_2 is applicable. Because of $r_2 \neq \circ$ the evaluation function will stop at the precedence level i_2. Since the precedences of R''_1 are always less then zero by construction whereas $i_2 \geq 0$ by normalization, the same rules applies in $Pol_1 \overset{<}{\cup} Pol_2$ and we obtain identical outputs, i.e., $Pol_1 \overset{<}{\cup} Pol_2$ also rules $(r_2, \bar{o}_2)$. Note that several rules might have already occurred that added obligations only. However, they occur in both Pol_2^* and $Pol_1 \overset{<}{\cup} Pol_2$ and hence do not cause any harm.
4. No rule of R'_2 fits the current request, but the request is in the scope of Pol_2^*. In this case, the (expanded) default ruling of Pol_2 applies for both $Pol_1 \overset{<}{\cup} Pol_2$ and Pol_2^*, since this ruling has higher precedence than any rule in R''_1. If $dr_2 \neq \circ$,

both policies rule $(\circ, \bar{o}_2)$ for some obligation set $\bar{o}_2$, which is again equal for both policies since they processed the same set of rules so far. If $dr_2 = \circ$ then the evaluation function starts searching for matching rules in R_1''. Here Pol_2^* outputs $(\circ, \bar{o}_2)$, whereas $Pol_1 \biguplus Pol_2$ outputs $(r, \bar{o}_2 \cup \bar{o}_1)$ for some r and $\bar{o}_1 \subseteq O_1$. In order to prove refinement, we obtain $\bar{o}_1 \cup \bar{o}_2 \rightarrow_{O_2} \bar{o}_2$ because of $\bar{o}_2 \subseteq \bar{o}_1 \cup \bar{o}_2$. This further implies $\bar{o}_1 \cup \bar{o}_2 \rightarrow_{O_1 \cup O_2} \bar{o}_2$. Because of $\bar{o}_2 \rightarrow_{O_2} \bar{o}_2$ we obtain $\bar{o}_1 \cup \bar{o}_2 \rightarrow_{O_1 \cup O_2} \bar{o}_2 \rightarrow_{O_2} \bar{o}_2$, which proves refinement since $\bar{o}_2 \in O_2 = (O_1 \cup O_2) \cap O_2$. ■

The composition operators fulfill certain laws:

Lemma 5 (Laws for Policy Composition). *Direct composition is commutative and ordered composition is associative, i.e., for all privacy policies* Pol_i *for* $i = 1, 2, 3$ *with pairwise compatible vocabularies, we have*

$$Pol_1 \bigcup Pol_2 \equiv Pol_2 \bigcup Pol_1;$$

$$\left(Pol_1 \overset{<}{\bigcup} Pol_2\right) \overset{<}{\bigcup} Pol_3 \equiv Pol_1 \overset{<}{\bigcup} \left(Pol_2 \overset{<}{\bigcup} Pol_3\right). \qquad \square$$

Proof. (Sketch) The commutativity part is obvious. We now show the associativity part. The composition of vocabularies is associative. For the rulesets, $(Pol_1 \overset{<}{\bigcup} Pol_2) \overset{<}{\bigcup} Pol_3$ yields a ruleset where 0 is the lowest precedence of Pol_3 and higher than the highest precedence of Pol_2, while $Pol_1 \overset{<}{\bigcup} (Pol_2 \overset{<}{\bigcup} Pol_3)$ yields a ruleset where 0 is the lowest precedence of Pol_2 and higher than the highest precedence of Pol_1. By shifting the second ruleset by $max(Pol_2) - min(Pol_2) + 1$ one obtains a ruleset that is identical, and thus clearly equivalent, to the first set. Since this precedence shift retains equivalence, the second ruleset is equivalent to the first ruleset. ■

5 Two-Layered Privacy Policies

An enterprise must abide by the law. In addition, the consent that an individual has granted when submitting data to an enterprise is mandatory and should not be changed. In contrast, unregulated and non-promised issues can be freely decided by the enterprise, i.e., enterprise privacy practices can be changed by the CPO or the administrators of the enterprise. In order to reflect this requirement, we now introduce a distinction between mandatory and discretionary parts of a policy. This represents a modal view of a policy semantics [18]: The mandatory part *must* be adhered to under any circumstances, the remaining part *may* be adhered to. We capture this view by introducing *two-layered policies*.

The real value of this notion lies in new possibilities for composition that cannot be captured by just taking the ordered composition of the discretionary part under the mandatory part.

5.1 Syntax and Semantics of Two-Layered Privacy Policies

Syntactically, a two-layered policy is simply a pair of (usual) privacy policies. The first element is the mandatory part and the second element the discretionary part.

Definition 23. *A pair $Pol = (Pol_1, Pol_2)$ of privacy policies with compatible vocabularies is called a* two-layered policy. *The policy Pol_1 is called the* mandatory part *of Pol, and Pol_2 the* discretionary part. $\diamond$

The semantics of such a two-layered policy is described as an algorithm, given an authorization request $q \in Req$ and an assignment $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$:

– *Evaluate mandatory policy.* Evaluate the request q under Pol_1, yielding $(r_1, \bar{o}_1)$.
– *First policy dominates.* If $r_1 \notin \{\circ, scope\_error\}$, output $(r_1, \bar{o}_1)$.
– *Evaluate second policy.* If $r_1 \in \{\circ, scope\_error\}$, evaluate the request q under Pol_2, yielding $(r_2, \bar{o}_2)$. If $r_2 = scope\_error$ and $r_1 = \circ$, output $(r_1, \bar{o}_1)$, else output $(r_2, \bar{o}_1 \cup \bar{o}_2)$.

This captures the intuition that Pol_1 is mandatory: Only if Pol_1 does not care about a request, or if it does not capture the request, the discretionary part Pol_2 is executed. Note that the mandatory part can still be used to accumulate obligations, e.g., for strictly requiring that every employee has to send a notification to his or her manager before a specific action.

We now show that the resulting semantics is the same as that of ordered composition, as one would expect. Recall, however, that the composition operators make use of the fact that a two-layered policy retains the information which parts were mandatory.

Lemma 6. *For $Pol := (Pol_1, Pol_2)$, the two-layered semantics is equivalent to the ordinary semantics of an ordered composition: $Pol \equiv Pol_2 \uplus Pol_1$.* $\square$

Proof. We have to show that evaluation of Pol and $PolC := Pol_2 \uplus Pol_1$ always return the same ruling and equivalent obligations. . Let us first assume that the evaluation of Pol outputs $(r_1, \bar{o}_1)$ with $r_1 \in \{+, -\}$ (first policy dominates), i.e., the request was in the scope of Pol_1 and produced a ruling $+$ or $-$. Hence when evaluating $PolC$, only higher precedence rules of Pol_1 are used and $PolC$ also outputs $(r_1, \bar{o}_1)$.

Let us now assume that $r_1 = scope\_error$ in Pol. Then the result is determined by Pol_2. The same holds for $PolC$.

Let us finally assume that $(r_1, \bar{o}_1)$ was output by Pol_1 with $r_1 = \circ$. If the request is in the scope of Pol_2, then Pol and $PolC$ will output the ruling of Pol_2 with a union of both obligations. If the request is out of the scope of Pol_2, the pair $(r_1, \bar{o}_1)$ is output in both cases. $\blacksquare$

This lemma and Lemma 4 imply that $(Pol_1, Pol_2) \prec Pol_1$, i.e., that every two-layered policy refines its mandatory part.

5.2 Refinement and Composition of Two-Layered Policies

For two-layered privacy policies, we define the following notion of refinement.

Definition 24 (Two-Layered Refinement). *Let two-layered policies $Pol = (Pol_1, Pol_2)$ and $Q = (Q_1, Q_2)$ be given where Pol_1 and Q_1 as well as Pol_2 and Q_2 have compatible vocabularies. Then Pol is a* refinement *of Q, written $Pol \prec Q$, iff $Pol_1 \prec Q_1$ and $Pol_2 \tilde{\prec} Q_2$.* $\diamond$

The distinction between mandatory and discretionary parts enabled us to use the notion of weak refinement: We require that the mandatory part is a normal refinement. This reflects that access rights as well as denials must be preserved. The discretionary part may be weakly refined. This reflects the fact that this part can be modified at the discretion of the enterprise.

Coming up with a meaningful definition of composing two-layered policies is more difficult. The main goal is to never violate any mandatory part. Composing the mandatory parts either according to Definition 21 or 22 would typically overrule some mandatory rules, which would defeat this goal. Therefore, we only allow to compose two-layered policies if the rulesets of their respective mandatory parts are *collision-free*, which means that for all requests and all assignments, it is never the case that one mandatory part accepts the request (i.e., it rules $+$), whereas the other one denies the request (i.e., it rules $-$).

Definition 25 (Collision-Free). *Two privacy policies Pol_1 and Pol_2 with compatible condition vocabularies are called* collision-free *if for all requests $q \in Req$ and all assignments $\chi \in \mathfrak{Ass}_{\subseteq}(Var_1 \cup Var_2)$ the following holds: If Pol_i rules $(r_i, \bar{o}_i)$ for $i = 1, 2$, then $\{r_1, r_2\} \neq \{+, -\}$.* $\diamond$

Two-layered policies with conflicting mandatory parts cannot be composed. For two-layered policies with collision-free mandatory parts, we define composition as follows:

Definition 26 (Two-Layered Composition). *Let two-layered policies $Pol = (Pol_1, Pol_2)$ and $Q = (Q_1, Q_2)$ be given where Pol_1 and Q_1 as well as Pol_2 and Q_2 have compatible vocabularies, and Pol_1 and Q_1 are collision-free. Then the* composition *of Pol and Q is defined as $Pol \bigcup Q := (Pol_1 \bigcup Q_1, Pol_2 \bigcup Q_2)$. Similarly, the* ordered composition *of Pol under Q is defined as $Pol \bigcup\!\!\!\!\!\lhd\; Q := (Pol_1 \bigcup\!\!\!\!\!\lhd\; Q_1, Pol_2 \bigcup\!\!\!\!\!\lhd\; Q_2)$.* $\diamond$

The following lemma shows the main property that this composition wants to preserve, i.e., that the resulting composed policy refines both mandatory parts.

Lemma 7. *Let two-layered policies $Pol = (Pol_1, Pol_2)$ and $Q = (Q_1, Q_2)$ be given such that Pol_1 and Q_1 as well as Pol_2 and Q_2 have compatible vocabularies. Moreover, Pol_1 and Q_1 are collision-free. Then $Pol \bigcup\!\!\!\!\!\lhd\; Q$ is a refinement of Q. Furthermore, $PolC := Pol \bigcup\!\!\!\!\!\lhd\; Q$ is a refinement of Pol_1 and Q_1.* $\square$

Proof. The first claim follows directly from Lemma 4. Lemma 6 implies that $PolC \equiv (Pol_2 \bigcup\!\!\!\!\!\lhd\; Q_2) \bigcup\!\!\!\!\!\lhd\; (Pol_1 \bigcup\!\!\!\!\!\lhd\; Q_1)$. The fact that $PolC$ is a refinement of Q_1 follows from Lemma 4.

Let us now assume that $PolC$ does not refine Pol_1. Since $PolC$ refines $(Pol_1 \bigcup\!\!\!\!\!\lhd\; Q_1)$, this implies that $(Pol_1 \bigcup\!\!\!\!\!\lhd\; Q_1)$ does not refine Pol_1. This implies that there exists a request within the scope of Pol_1 and Q_1 such that Q_1 does not rule $\circ$ since otherwise the ruling of Pol_1 would be output. For this request, the rulings of Pol_1 and Q_1 differ, which contradicts our assumption that Pol_1 and Q_1 are collision-free. $\blacksquare$

6 Conclusion

Privacy policies are a core component for enterprise privacy technologies. The current proposals for privacy policies required policy authors to create one single overall policy for the complete enterprise(s) that are covered. We therefore described a toolkit to handle multiple policies. This includes refinement for auditing and policy validation and composition for multi-domain or delegated-authorship policies. In addition, we introduced a new notion of two-layered policies to track mandatory and discretionary parts. This enables privacy administrators to detect and resolve conflicts between mandatory policies, e.g., if a customer promise or another contract would violate a law. These tools enable the privacy officers of an enterprise to create and manage the complex privacy policy of an enterprise more efficiently while retaining the semantic rigor that is required for trustworthy privacy management.

Acknowledgments

We thank Günter Karjoth, Calvin Powers, and Michael Waidner for valuable discussions.

References

1. P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Authorization Language (EPAL). Research Report 3485, IBM Research, 2003. `http://www.zurich.ibm.com/security/enterprise-privacy/epal/specification`.
2. P. Ashley, S. Hada, G. Karjoth, and M. Schunter. E-P3P privacy policies and privacy authorization. In *Proc. 1st ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–109, 2002.
3. A. Belokosztolszki and K. Moody. Meta-policies for distributed role-based access control systems. In *Proc. 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 106–115, 2002.
4. C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekerat. Obligation monitoring in policy management. In *Proc. 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 2–12, 2002.
5. P. A. Bonatti, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. A component-based architecture for secure data publication. In *Proc. 17th Annual Computer Security Applications Conference*, pages 309–318, 2001.
6. P. A. Bonatti, S. De Capitani di Vimercati, and P. Samarati. A modular approach to composing access control policies. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 164–173, 2000.
7. P. A. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, 2002.
8. A. Cavoukian and T. J. Hamilton. *The Privacy Payoff: How successful businesses build customer trust*. McGraw-Hill/Ryerson, 2002.
9. S. De Capitani di Vimercati and P. Samarati. An authorization model for federated systems. In *Proc. 4th European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 99–117. Springer, 1996.

10. S. Fischer-Hübner. *IT-security and privacy: Design and use of privacy-enhancing security mechanisms*, volume 1958 of *Lecture Notes in Computer Science*. Springer, 2002.
11. Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu. IPSec/VPN security policy: Correctness, conflict detection and resolution. In *Proc. 2nd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, volume 1995 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2001.
12. V. Gligor, H. Khurana, R. Koleva, V. Bharadwaj, and J. Baras. On the negotiation of access control policies. In *Proc. 9th International Workshop on Security Protocols*, 2002.
13. H. Hosmer. The multipolicy paradigm. In *Proc. 15th National Computer Security Conference*, pages 409–422, 1993.
14. S. Jajodia, M. Kudo, and V. S. Subrahmanian. Provisional authorization. In *Proc. E-commerce Security and Privacy*, pages 133–159. Kluwer Academic Publishers, 2001.
15. S. Jajodia, P. Samarati, M. L. Sapino, and V. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(4):216–260, 2001.
16. G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 271–281, 2002.
17. G. Karjoth, M. Schunter, and M. Waidner. The platform for enterprise privacy practices – privacy-enabled management of customer data. In *Proc. Privacy Enhancing Technologies Conference*, volume 2482 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2002.
18. J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed systems management. *IEEE JSAC Special Issue on Network Management*, 11(9):1404–31414, 1993.
19. Platform for Privacy Preferences (P3P). W3C Recommendation, Apr. 2002. `http://www.w3.org/TR/2002/REC-P3P-20020416/`.
20. C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes. SPL: An access control language for security policies with complex constraints. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2001.
21. TRUSTe. Privacy Certification. Available at `www.truste.com`.
22. eXtensible Access Control Markup Language (XACML). OASIS Committee Specification 1.0, Dec. 2002. `www.oasis-open.org/committees/xacml`.