Enterprise privacy promises and enforcement

Adam Barth
Stanford University
abarth@cs.stanford.edu

John C. Mitchell
Stanford University
mitchell@cs.stanford.edu

ABSTRACT

Several formal languages have been proposed to encode privacy policies, ranging from the Platform for Privacy Preferences (P3P), intended for communicating privacy policies to consumers over the web, to the Enterprise Privacy Authorization Language (EPAL), intended to enable policy enforcement within an enterprise. However, current technology does not allow an enterprise to determine whether its detailed, internal enforcement policy meets its published privacy promises. We present a data-centric, unified model for privacy, equipped with a modal logic for reasoning about permission inheritance across data hierarchies. We use this model to critique two privacy preference languages (APPEL and XPref), to justify P3P's policy summarization algorithm, and to connect privacy policy languages, such as P3P, with privacy policy enforcement languages, such as EPAL. Specifically, we characterize when one policy enforces another and provide an algorithm for generating the most specific privacy promises, at a given level of detail, guaranteed by a more detailed enforcement policy.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security, Theory, Languages

Keywords

Privacy policy, EPAL, P3P, Policy summary, Modal logic

1. INTRODUCTION

Privacy is a significant concern among online consumers [1]. In response to this concern, online service providers have developed privacy policies. Initially, service providers posted privacy policies on their web sites in free text. However, free-text policies are difficult for consumers to understand [16]. The World Wide Web Consortium has proposed the Platform for Privacy Preferences (P3P) to enable service providers to post machine-readable privacy policies [13]. Service providers who have posted P3P policies promise specific data practices, but they still require internal mechanisms to enforce those promises. One approach is to detail data practices in a formal authorization language. IBM has proposed the Enterprise Privacy Authorization Language (EPAL) as one such enforcement language [17]. Instead of using EPAL as our example enforcement language, however, we use DPAL, a similar language with several advantages [6], including straightforward ways of combining policies. Regardless of the enforcement language used, it is not obvious how enforcement policies relate to privacy promises.

Our goals are fourfold. First, to describe the semantics of privacy languages, such as P3P and DPAL, in a rigorous, uniform model. Such a description is essential in relating policies written in different languages. Second, to validate our model, the Data-Hierarchy/Action Model for Privacy (DAMP), by analyzing the existing relations between P3P and languages designed to interoperate with P3P, such as APPEL, XPref, and compact P3P policies. Third, to determine whether a policy, such as one written in DPAL, enforces the promises made in another policy, such as one written in P3P, thereby connecting privacy promises with privacy enforcement. Finally, to provide an algorithm for summarizing detailed policies using a given vocabulary, enabling enterprises to translate their operative DPAL policies into P3P. Achieving these goals will provide enterprises with the tools to ensure their privacy enforcement mechanisms actually enforce their announced privacy policies.

Figure 1 depicts one usage scenario for the main relations and algorithms. A service provider first writes a DPAL policy detailing its data practices, then employs our summarization algorithm to generate the most specific P3P policy enforced by this DPAL policy. The service provider may then, optionally, generate a compact P3P policy using the algorithm in the P3P specification. We prove that each compact policy is enforced by the P3P policies that generate it. Either the full or compact P3P policy is then transmitted to a consumer's user agent, which is configured using its user's privacy preferences, expressed in APPEL. If the consumer's user agent accepts the received policy, the service provider's data practices conform to the consumer's privacy preferences.

WITS'05, January 10, 2005, Long Beach, CA, USA.

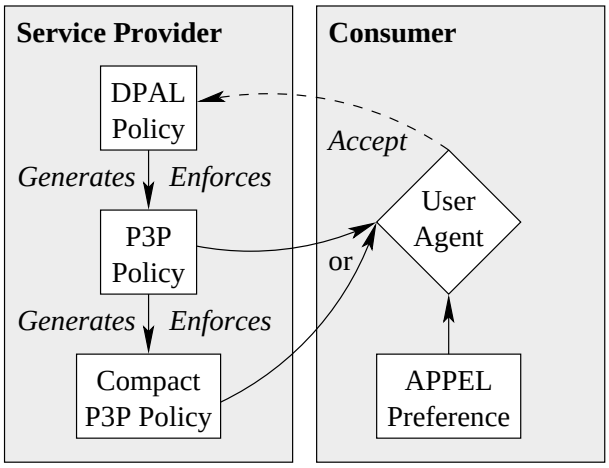

Figure 1: The intended usage scenario for the tools in this paper.

1.1 Perspectives and modalities

It is important to understand that different privacy languages address privacy concerns from different perspectives. P3P policies are intended to be interpreted by consumers (or their automated agents), whereas DPAL policies are intended to be interpreted by the issuing service provider. Consumers and service providers are interested in answering different types of questions about privacy policies, and consequently may understand a policy summary as allowing different actions. P3P is designed to easily answer consumer questions; DPAL is designed to easily answer service provider questions. Previous attempts to model privacy policies [17] do not distinguish between these two perspectives. Such a distinction, however, is necessary in order to relate privacy preferences to privacy enforcement.

Consumers desire privacy policies that permit as little use of their personal data as possible. Most consumers, however, are willing to give up some privacy in exchange for a service. In our model, consumers reject candidate privacy policies that exceed some level of permissiveness. For example, a consumer might reject a policy because it permits telemarketing. In the face of incomplete or conflicting information, consumers use an upper bound for their determinations. A consumer considering a policy permitting the use of some, but not all, personal data for telemarketing would summarize the policy as allowing telemarketing, and therefore might reject it. In our model, elaborated in this paper, the ability to perform an action using *some parts* of a large piece of data is expressed using the $\Diamond$ modality.

Service providers desire privacy policies that permit as much use of their customers' personal data as is useful. Many service providers, however, are willing to limit their use of personal data in order to do business with privacy-conscious consumers. Before performing an action, a service provider must determine whether it has promised not to perform that action. For example, a service provider might need to determine whether it is allowed to use a customer's personal data for telemarketing. In the face of incomplete or conflicting information, service providers use a lower bound for their determinations. A service provider operating under a policy permitting the use of some, but not all, personal data for telemarketing could summarize this policy as not allowing all telemarketing, and therefore could restrict its actions conservatively. The ability to perform an action using *all parts* of a large piece of data is modelled using the $\Box$ modality.

Notice that for a single detailed policy allowing some but not all data to be used for telemarketing, a privacy-conscious consumer would like this policy to be summarized (using the $\Diamond$ modality) as "allows telemarketing" so the consumer can reject the policy, whereas a service provider who has committed to the policy must summarize the policy (using the $\Box$ modality) as "prohibits telemarketing" in order to avoid violating the policy when it is enforced using the policy summary.

We have developed the Data-Hierarchy/Action Model for Privacy (DAMP), a data-centric model for privacy policies. DAMP distinguishes privacy promises about an individual data object from restrictions inherited from promises about other data objects. This formal separation is necessary because consumers and service providers have different perspectives on privacy and therefore on interpreting promises about related data objects. Their dual perspectives are captured in the dual $\Diamond$ and $\Box$ modalities of a modal logic for reasoning about privacy policies.

1.2 Applications

In our usage scenario, consumers require a language, such as APPEL [13], for expressing their privacy preferences. Critics have argued that APPEL is difficult to use effectively and have proposed XPref as an APPEL replacement [2]. Using DAMP to analyze APPEL and XPref, we discover that both languages can express dubious preferences. For example, a consumer may express the preference to block services *not* performing telemarketing. Moreover, expressing simple preferences, such as "block services that collect my home address," in XPref is non-intuitive.

P3P specifies an algorithm for generating compact policies, which are short summaries of P3P policies. DAMP endows this algorithm with clear semantics: compact policies are answers to common consumer queries. To compute compact policies, compute the values of certain $\Diamond$ terms in the modal logic. Service providers might employ a dual notion of a compact policy, formed by replacing $\Diamond$ terms with $\Box$ terms.

Practicable enforcement languages, such as DPAL, also compute the values of $\Box$ terms; however, they do so at a finer level of detail. Seemingly ad-hoc features of DPAL, such as upward inheritance of deny rulings, can be motivated using the $\Box$ operator. Deviations from this semantics result in unsafe [6] policies. Using DAMP, we demonstrate how service providers can determine whether their operative DPAL policy actually enforces their announced P3P policy. We also provide an algorithm for generating the most specific P3P policy enforced by a given DPAL policy.

The remainder of this paper is organized as follows. Section 2 presents our privacy model, DAMP, including an extended example and rigorous definitions. Section 3 applies DAMP to understanding P3P, with attention to privacy preference languages and compact policies. Section 4 examines enforcement languages, justifying some peculiarities of DPAL and criticizing others. Section 4 also describes an algorithm for generating P3P policies from DPAL policies. Section 5 concludes.

2. A MODEL FOR PRIVACY

Previous models of privacy policies [17] do not distinguish between the perspectives of those who handle personal data and those whose personal data is being handled. Such a distinction, however, is necessary in order to relate privacy preferences to privacy enforcement. This section presents a model of privacy, the Data-Hierarchy/Action Model for Privacy (DAMP) mindful of these two perspectives.

2.1 Principals, data, and actions

Traditionally, in models of access control [15], there are two types of principals: the "good guys," and the "bad guys." Each group desires different properties of the access control system. The "good guys" desire a lower bound on the functionality of their system, whereas the "bad guys" desire maximal functionality.

Access control policies are partially ordered by the amount of functionality they permit. Viewed in terms of policies, the "good guys" require a policy more permissive than some lower bound policy, p, that permits their desired minimal functionality. Left to their own devices, the "bad guys" would select the maximally permissive policy, $\top$, in order to conduct their nefarious business. The principle of least privilege (see, for example, [7]) states that the "good guys" should enforce p itself.

Privacy policies are also partially ordered by permissiveness and may be understood in terms of two principals: the service provider and the consumer. The service provider desires a policy at least as permissive as some policy, p_s, whereas the consumer desires a policy at *most* as permissive as another policy, p_c. The service provider imposes a lower bound on functionality in order to conduct his or her business. The consumer imposes an upper bound on functionality in order to protect his or her privacy. A policy p with $p_s \sqsubseteq p \sqsubseteq p_c$ is acceptable to both parties and may be used by the service provider to govern data collected about the consumer.

The central component of this model is the data hierarchy. A service provider who collects data about a consumer, such as given name, zip code, and blood cholesterol levels, often manipulates data in aggregates, such as home address and blood test results. Data objects about an individual are thus partially ordered by inclusion and form a hierarchy. When a service provider, in a privacy policy, makes a promise about a data object, that promise also indirectly affects containing objects.

The final components of this privacy model are actions. Previous work in privacy [5, 19] modelled aspects of actions, such as purpose and recipient. For simplicity, we abstract away these aspects in our model, taking actions are parameterized only by the data object on which they act. However, this is only a modelling convenience, as actions may be complex, for example, "Disclose X to health insurer for billing," or may occur over a long duration: "Retain X for five years and then expunge." Policies state concretely which actions are permitted and which actions are prohibited of the service provider. We assume the existence of a global set of actions, A, containing every possible action.

2.2 Extended example

In the following example, Alice is the consumer and Dr. Bob is the service provider. They are concerned with the data objects blood cholesterol level, T-cell count, and blood test results. Blood test results contain both blood cholesterol level and T-cell count. They are concerned with the action "disclose X."

Alice is HIV-positive. She wishes to keep her medical records private because she fears she will be denied health insurance if prospective insurers learn her HIV status. Conservatively, she can prohibit disclosure of her entire medical history, but she can obtain a better insurance rate if she permits certain disclosures. She is willing to disclose some of her record, such as her age, weight, and x-rays, but she does not wish to disclose results of blood tests. In evaluating privacy policies, Alice decides to ask, "Does this policy permit disclosure of blood test results?"

After framing her question, Alice consults the privacy policy of her physician, Dr. Bob, to determine if he will respect her preference to keep blood test results confidential. In his policy, Dr. Bob promises not to disclose blood cholesterol levels. This is not sufficient to satisfy Alice because it does not preclude Dr. Bob from disclosing her T-cell count, a blood test result. Alice, therefore, concludes that Dr. Bob's policy does permit disclosure of some important blood test results.

Dr. Bob's perspective on his policy is different from Alice's perspective. In order to provide quality care for his patients, Dr. Bob wishes to disclose certain records, such as blood test results. In evaluating his privacy policy, he decides to ask, like Alice, "Does this policy permit disclosure of blood test results?" His policy promises not to disclose blood cholesterol levels, and therefore, prohibits him from disclosing blood test results in their entirety. He concludes his policy does not permit disclosure of blood test results.

Alice and Dr. Bob appear to be asking the same question, but their questions differ in their use of quantifiers. Alice is worried about Dr. Bob disclosing part of her blood test results, so she may restate her question as, "Does this policy permit disclosure of *any* blood test results?" Dr. Bob is worried about respecting each of his promises about blood test results, so he may restate his question as, "Does this policy permit disclosure of *all* blood test results?" Their different questions lead to different answers.

2.3 The model

We model privacy policies as functions mapping data objects to sets of actions. The issuing service provider promises to restrict his or her actions on a data object to the set of actions indicated by the policy. This unusual policy model separates promises about a data object from promises about related data objects, allowing us to understand the inheritance of restrictions in data hierarchies and to explore some proposed privacy preference languages.

Let A be the set of all potential actions, and let D be the set of data objects, ordered by $\leq_D$ such that if $d_1, d_2 \in D$ with $d_1 \leq_D d_2$, then d_1 is a component of d_2. For example, "disclose" is an action, and "blood cholesterol level" is a data object, which is also a component of the data object "blood test result." A privacy policy, p, is a function from D to subsets of A, indicating that the issuing service provider promises to confine its actions regarding each d to the set $p(d)$.

A policy p does not directly state whether an action is permitted. Instead, it serves an adjudicatory function. A service provider who performs action a on data object d has violated p if $a \notin p(d)$. However, $a \in p(d)$ does not imply a

may be performed on d. When a service provider performs action a on datum d, that provider effectively performs a on all $d' \leq_D d$. For example, if Dr. Bob discloses blood test results, he necessarily discloses blood cholesterol levels. Therefore, when contemplating performing action a on datum d, the policy's issuer must not only check that $a \in p(d)$, but also that $a \in p(d')$ for all $d' \leq_D d$. Suppose this second condition is violated at d' because $d' \leq_D d$ and $a \notin p(d')$. If the policy's issuer performs a on d, he or she effectively has performed a on d', violating a promise set forth in p. Dr. Bob, in disclosing blood test results, violates his policy to restrict his actions on blood cholesterol levels to a set not containing "disclose."

Consider the consumer's perspective in this model. He or she is interested in understanding p in order to determine whether it sufficiently restricts the issuing service provider's actions. A natural criterion is whether the provider has promised *not* to perform action a on data object d. Naïvely, the consumer might ask whether $a \in p(d)$. However, for many common actions governed by privacy policies, such as information disclosure, this question is insufficient to protect the consumer's privacy. Although Dr. Bob will not disclose complete blood test results, he might disclose T-cell counts. To be conservative, consumers must, therefore, ask whether $a \in p(d')$ for each $d' \leq_D d$.

The quantifiers in these questions can be viewed as modalities. The partial order on D can be interpreted as the accessibility relation in the Kripke frame [9] $(D, \leq_D)$, where d_1 is accessible from d_2 if and only if $d_1 \leq_D d_2$. [1] Individual actions can be viewed as atomic propositions.

Def. The **Kripke model**, $\mathfrak{K}_p$, for policy p is $(D, \leq_D, \Vdash_p)$, where for all $d \in D$ and all $a \in A$,

$$d \Vdash_p a \iff a \in p(d).$$

We extend the $\Vdash$ relation to modal formulae in the standard manner,

$$\begin{aligned} d \Vdash_p \varphi_1 \wedge \varphi_2 &\iff d \Vdash_p \varphi_1 \text{ and } d \Vdash_p \varphi_2 \\ d \Vdash_p \neg\varphi &\iff d \nVdash_p \varphi \\ d \Vdash_p \Box\varphi &\iff (\forall d' \leq_D d)(d' \Vdash_p \varphi), \end{aligned}$$

and adopt the convention that $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, and $\Diamond\varphi \equiv \neg\Box\neg\varphi$.

To determine whether he or she may perform action a on the entire data object d under policy p, a service provider evaluates $d \Vdash_p \Box a$. Dually, a consumer can avoid service providers who might perform a on a component of d by evaluating $d \Vdash_p \Diamond a$.

Our use of modal logic differs from previous work [14, 18] using modalities to reason about security. In [14], modalities were used to reason about knowledge and future events. [18] used modal logic to understand the evolution of policies over time. In the present work, modalities are used to reason about components of data objects. [11] examined rights inheritance in role-based access control hierarchies. However, there the modalities occur only implicitly.

[1] This notation is non-standard. Usually, aRb indicates b is accessible from a.

2.4 Enforcement

Def. Given privacy policies p and q with a common data hierarchy D, p **entails** q, written $p \sqsubseteq q$, if for all $d \in D$,

$$p(d) \subseteq q(d).$$

The $\sqsubseteq$ relation captures the intuitive notion that one policy is less permissive than another if the former makes a stronger promise, for every data object, than the latter. $\sqsubseteq$ does not compare policies without a common data hierarchy, which is unfortunate because service providers often wish to compare policies dealing with data at different levels of detail. For example, an enterprise's operative enforcement policy is often more detailed than the privacy policy it announces on its web site. A certain class of modal formulae are used to generalize this relation to policies dealing with data at different levels of detail.

Def. A modal formula is **positive** if its symbols are among actions, $\wedge$, $\vee$, $\Box$, and $\Diamond$ (i.e., it must not contain $\neg$ or $\rightarrow$).

A positive modal formula true of a policy is true of all less restrictive policies. Positive formulae are the basic building blocks for reasoning about policies and are used by both consumers and service providers.

LEMMA 1. *Given privacy policies p and q, with a common data hierarchy D, if $p \sqsubseteq q$,*

$$d \Vdash_p \varphi \implies d \Vdash_q \varphi$$

for all $d \in D$ and all positive modal formulae φ.

PROOF. Assume $p \sqsubseteq q$. Proceed by induction on positive, simple, modal formulae. Assume $\varphi = a$. Given $d \in D$, assume $d \Vdash_p a \Leftrightarrow a \in p(d) \Rightarrow a \in q(d) \Leftrightarrow d \Vdash_q a$. The inductive steps for $\wedge$ and $\vee$ are immediate.

Assume $\varphi = \Box\psi$. Given $d \in D$, assume $d \Vdash_p \Box\psi$. Given $d' \leq_D d$, $d' \Vdash_p \psi$ and, by induction, $d' \Vdash_q \psi$. Therefore, $d \Vdash_q \Box\psi$.

Assume $\varphi = \Diamond\psi$. Given $d \in D$, assume $d \Vdash_p \Diamond\psi$. Fix $d' \leq_D d$ such that $d' \Vdash_p \psi$. By induction, $d' \Vdash_q \psi$ and, therefore, $d \Vdash_q \Diamond\psi$. □

Consumers and service providers are interested in simple formulae, such as $\Box a$ and $\Diamond a$. We generalize $\sqsubseteq$ by requiring simple formulae to carry over from the more restrictive policy to the less restrictive policy.

Def. A modal formula is **simple** if it does not contain nested modalities.

Intuitively, one policy enforces another if any action permitted under the former is announced under the latter. Enforced policies constitute an "upper bound" on enforcement policies. This allows consumers to determine whether the enforcing policy conforms to their preferences by examining the enforced policy.

Def. Given policies p and q, with data hierarchies D_p and D_q, respectively, and $D_p \subseteq D_q$, q **enforces** p if and only if

$$d \Vdash_q \varphi \implies d \Vdash_p \varphi$$

for all $d \in D_p$ and all positive, simple, modal formulae φ.

The "enforces" relation is a direct generalization of the $\sqsubseteq$ relation. If p and q share a common data hierarchy, the two relations are identical.

LEMMA 2. *Given privacy policies p and q, with a common data hierarchy D,*

$$q \sqsubseteq p \iff q \text{ enforces } p.$$

PROOF. The left-to-right direction is a special case of Lemma 1. For the right-to-left direction, given a and $d \in D$, assume $a \in q(d)$. $d \Vdash_q a$ and, because q enforces p, $d \Vdash_p a$. Therefore, $a \in p(d)$ and $q \sqsubseteq p$. $\square$

The enforces relation is similar to the policy refinement relation in [4], but there are two key differences. First, policy refinement is EPAL-specific and depends crucially on EPAL's semantics. This prevents policy refinement from relating two policies written in different languages. Second, whether one policy refines another depends on the particular syntactic expression used to represent each policy. Whether one policy enforces another depends solely on the policies themselves, and not their syntactic expression. Both are generalizations of $\sqsubseteq$ in the sense of Lemma 2.

3. UNDERSTANDING P3P POLICIES

A World Wide Web Consortium standard, the Platform for Privacy Preferences, or P3P, is a broadly adopted [8] formal language for communicating privacy promises to consumers [13]. A P3P policy is a promise by a service provider to limit the use of certain data to certain purposes, recipients, and retention periods.

Prior to retrieving a web page, a consumer's web browser first downloads the site's P3P policy, and then compares the downloaded policy against its user's privacy preferences. If the policy respects the user's preferences, the web browser retrieves the web page. However, if the policy does not respect the user's preferences, the browser may block the site or notify the user. When manipulating data, the web site operator is obligated to adhere to the P3P policy under which it collected the data.

3.1 Modelling P3P policies

P3P policies govern a data hierarchy, the base data schema, or an extension thereof. P3P policies are comprised of statements, each of which attaches a set of purposes, a set of recipients, and a set of retention periods to each of a collection of data objects, indicating the issuer promises to restrict its actions concerning those data objects to those purposes, those recipients, and those retention periods.

In the P3P statement in Figure 2, `webmd.com` promises to restrict use of the user's name and birthday to the "current" purpose, as well as individual analysis. `webmd.com` may disclose the user's name and birthday to delivery agents, but this disclosure will be made only if the user opts in. Finally, `webmd.com` may retain the user's name and birthday to meet legal requirements on data retention.

A P3P statement attaching a promise to a node in the data hierarchy (e.g. `#user.name`) also attaches that promise to all children of that node (e.g. `#user.name.given`). If two statements attach different promises to the same data object, the issuer is permitted to perform actions permissible under either promise.

A P3P statement, σ, can be interpreted as a function, $\mathbb{P}[\![\sigma]\!]$, from a data hierarchy, D, to subsets of actions, $\mathcal{P}(A)$,

```
<STATEMENT>
 <PURPOSE><current/><individual-analysis/>
 </PURPOSE><RECIPIENT>
  <delivery required="opt-in"/></RECIPIENT>
 <RETENTION><legal-requirement/></RETENTION>
 <DATA-GROUP>
  <DATA ref="#user.name" optional="yes"/>
  <DATA ref="#user.bdate"/>
 </DATA-GROUP>
</STATEMENT>
```

Figure 2: An abbreviated P3P statement from `webmd.com`. In this statement, `webmd.com` promises to restrict the use of a user's name and birth date to the "current" purpose and individual analysis. `webmd.com` may disclose a user's name and birth date to delivery agents, but only if the user opts in. Finally, `webmd.com` may retain the user's name and birthday to meet legal requirements on data retention.

indicating the issuer promises to restrict his or her use of each data object, d, to the set of actions $\mathbb{P}[\![\sigma]\!]d$. Data objects mentioned in the statement, and their children in the hierarchy, are mapped to the promise indicated. Other data objects are mapped to the most restrictive promise, the empty set of actions, which permits no action.

A P3P policy p, formally a finite set of P3P statements, may also be interpreted as a function, $\mathbb{P}[\![p]\!]$, from D to $\mathcal{P}(A)$, such that, for all $d \in D$,

$$\mathbb{P}[\![p]\!]d = \bigcup_{\sigma \in p} \mathbb{P}[\![\sigma]\!]d$$

The above use of union follows from the disjunctive semantics of P3P, which permits an action if it is permitted by at least one statement. The Kripke model for a policy p is $\mathfrak{K}_{\mathbb{P}[\![p]\!]}$.

3.2 Privacy preferences

P3P policies are intended to be interpreted by automated user agents. A consumer's agent examines the data practices promised by a P3P policy and reports whether it respects the consumer's privacy preferences. These privacy preferences, written in a formal language, allow consumers to specify which data practices they find acceptable and which they find unacceptable.

Def. f is a **privacy preference** if it is a unary predicate on policies (i.e., it is a function from policies to $\{true, false\}$).

According to our model, consumers prefer more restrictive policies. A *robust* preference that accepts a policy (i.e., it maps that policy to *true*) will also accept more restrictive policies.

Def. Privacy preference f is **robust** if, for all policies $p \sqsubseteq p'$, $f(p') \implies f(p)$.

A robust preference is an upper bound on policy permissiveness. For example, a preference to block web sites that use home telephone numbers for telemarketing is robust, whereas a preference to block web sites that *do not* use home telephone numbers for telemarketing is not robust.

```
<appel:RULE behavior="block">
 <p3p:POLICY>
  <p3p:STATEMENT appel:connective="or">
   <p3p:PURPOSE appel:connective="non-and">
    <p3p:telemarketing />
   </p3p:PURPOSE>
  </p3p:STATEMENT>
 </p3p:POLICY>
</appel:RULE>
```

Figure 3: A non-robust APPEL preference. The connective `non-and` causes the rule to fire for policies that do not disclose the `telemarketing` purpose.

Ideally, a preference language should not be able to express non-robust preference because a consumer who writes such a preference is almost certainly making an error.

LEMMA 3. *Given action a and data object d, if f is the privacy preference*

$$f(p) \iff d \not\Vdash_p \Diamond a,$$

for all policies p with data hierarchies containing d, then f is robust.

PROOF. Assume, by way of contradiction, f is not robust. There must exist policies p and q with $p \sqsubseteq q$ and $f(q)$, but $\neg f(p)$. $d \Vdash_p \Diamond a$, because $\neg f(p)$. Fix $d' \in D_p$ such that $d' \leq_{D_p} d$ and $d' \Vdash_p a$. By the definition of $\sqsubseteq$, $d' \in D_q$ and $d' \Vdash_q a$, implying $d \Vdash_q \Diamond a$. This contradicts $f(q)$. □

The P3P specification defines a language for expressing privacy preferences, named A P3P Preference Exchange Language (APPEL) [13]. An APPEL preference reports whether interactions with a service provider espousing a P3P policy should be blocked, limited, or allowed to proceed.

An APPEL preference is a set of rules. Each rule consists of a judgment (`block`, `limited`, or `request`) and a condition under which to issue that judgment. Rules are processed in order, optionally halting at the first rule whose condition is met. This early evaluation termination immediately leads to non-robust policies. However, even an APPEL preference containing a single rule might not be robust. The rule in Figure 3 blocks services that do not use collected information for telemarketing. This is not robust behaviour.

XPref is another language for expressing privacy preferences, based on XPath [10]. The designers of XPref were motivated by their difficulty expressing simple privacy preferences in APPEL [2]. Following from this motivation, they endowed XPref with significant expressive power. Unfortunately, XPref can express non-robust preferences. For example, the rule in Figure 4 blocks services that do not use collected information for telemarketing. This is not robust behaviour. Several of the example XPref policies in [2] are non-robust.

Using XPref robustly is non-intuitive. Consider expressing the preference "block services that collect my home address." Naïvely, a user might expect that preference to be expressed by the rule in Figure 5(a). However, such a preference will not block a service that discloses it collects `user.home-info` data, which includes postal address. A user's second attempt to encode this preference in XPref might be the rule in Figure 5(b). Such a preference is still

```
<RULE behavior="block"
 condition="/POLICY/STATEMENT/PURPOSE/*
  [ name(.) != "telemarketing" ]" />
```

Figure 4: A non-robust XPath policy. This rule fires for policies that do not disclose the `telemarketing` purpose.

(a)
```
<RULE behavior="block"
 condition="/POLICY/STATEMENT/DATA-GROUP/*
  [ name(.) = "DATA" and
    @ref = "#user.home-info.postal" ]" />
```

(b)
```
<RULE behavior="block"
 condition="/POLICY/STATEMENT/DATA-GROUP/*
  [ name(.) = "DATA" and
    ( @ref = "#user.home-info.postal" or
      @ref = "#user.home-info" or
      @ref = "#user" ) ]" />
```

(c)
```
<RULE behavior="block"
 condition="/POLICY/STATEMENT/DATA-GROUP/*
  [ name(.) = "DATA" and
    ( starts-with(@ref,
        "#user.home-info.postal") or
      @ref = "#user.home-info" or
      @ref = "#user" ) ]" />
```

Figure 5: Approximations to a robust XPref preference. (a) does not fire for policies that disclose `#user.home-info`. (b) does not fire for policies that disclose components of `#user.home-info.postal`. (c) is robust.

lacking. A service might disclose it collects each individual subelement of `user.home-info.postal`, such as `street`, without explicitly disclosing it collects home addresses. Figure 5(c) correctly expresses this preference by blocking services whose policy, for an action a, satisfies

$$\texttt{\#user.home-info.postal} \Vdash \Diamond a.$$

3.3 Compact policies

Compact policies are terse summaries of P3P policies. Service providers can include compact policies in HTTP headers to improve performance. Microsoft Internet Explorer 6.0 uses compact policies associated with cookies to implement its privacy protections [12]. A compact policy consists of a set of terms representing the purposes, recipients, and retention periods of the P3P vocabulary. For example, the term `TEL` represents the telemarketing purpose. The P3P specification includes an algorithm for condensing full P3P policies into compact policies but does not explain its semantics. Our model endows the algorithm with clear semantics.

Let τ be a compact P3P term, aside from `NID`. The term τ is present in a compact policy if (and only if) the full policy satisfies $d \Vdash \Diamond a$ for some a permitted by τ, where d is the root (all-inclusive) data object.

The term `NID`, or `NON-IDENTIFIABLE`, differs from other P3P terms because it represents the formula $\neg\Diamond a$, where a is the act of using a piece of data to identify an individual

consumer. It is present in a compact policy if (and only if) the full policy satisfies $d \Vdash \neg\Diamond a$, where d is the root element of the P3P data hierarchy. This is a performance optimization in P3P, whose designers believed the vast majority of P3P policies would satisfy $d \Vdash \Diamond a$ [12]. A compact policy, therefore, consists of the answers to certain $\Diamond$ queries.

THEOREM 4. *All P3P policies enforce their compact representations.*

PROOF. Given P3P policy q with data hierarchy D_q, let p be its compact representation. The data hierarchy for p is the trivial, single-element hierarchy, D_p, containing only d. Proceed by induction on positive, simple, modal formulae. Given φ, assume $\varphi = a$ for some action $a \in A$ and $d \Vdash_q a$. Thus, $d \Vdash_q \Diamond a$ and $d \Vdash_p a$. The inductive steps for $\wedge$ and $\vee$ are immediate.

Assume $\varphi = \Box\psi$, where ψ is free of modalities, and $d \Vdash_q \Box\psi$. Thus, $d \Vdash_q \psi$ and, by induction, $d_p \Vdash_p \psi$. As D_p contains only one element, $d_p \Vdash_p \Box\psi$.

Assume $\varphi = \Diamond\psi$, where ψ is free of modalities, and $d \Vdash_q \Diamond\psi$. Fix $d' \in D_q$, such that $d' \leq_{D_q} d$ and $d' \Vdash_q \psi$. By the definition of compact policies, $q(d') \subseteq p(d)$. Thus, $d \Vdash_p \psi$ and $d \Vdash_p \Diamond\psi$. Combining these statements, q enforces p. □

Consumers may use compact policies to make simple privacy decisions. For example, consumers may avoid service providers who use their information for telemarketing by avoiding those providers with compact policies containing the term `TEL`. However, such compact policies are of little use to service providers.

A different policy summary, for use by a service provider, might be constructed by recording the results of $\Box$ queries. If the term `TEL` appeared in this policy summary, a service provider could conclude he or she may use any piece of data for telemarketing. In practice, however, service providers require a fine-grained privacy enforcement language, such as DPAL.

4. ENFORCING PRIVACY PROMISES

P3P is useful for communicating data practices between service providers and consumers. However, it is lacking as a language for enforcing privacy policies within the enterprise. A P3P policy is, essentially, a precise summary of an enterprise's data practices and therefore lacks the detail necessary for enforcement.

IBM has proposed EPAL as a privacy enforcement language [20]. In [6], we discuss some shortcomings of EPAL and propose an improved language, DPAL. An enterprise enforcing a DPAL policy may wish to determine whether its DPAL policy is consistent with its announced P3P policy. This section contains the tools necessary for making that determination.

Enterprises enforcing DPAL policies may wish to create P3P policies for use on their web sites. These enterprises have two criteria for the generated P3P policy. First, the P3P policy must be enforced by the DPAL policy, so as to be honest about data practices. Second, the P3P policy should be as restrictive as possible, so as to appeal to as many consumers as possible. This section presents a general algorithm for summarizing privacy policies that an enterprise can use to generate such a P3P policy from a DPAL policy.

4.1 DPAL as an enforcement language

Like a P3P policy, a DPAL policy consists of a set of statements, each of which applies to a set of data objects and restricts the actions the service provider may undertake. Unlike in a P3P policy, an action is permitted by a DPAL policy only if it is permitted by each statement in the policy.

Interpreting DPAL statements is similar to interpreting P3P statements. Each DPAL statement, σ, is interpreted as a function, $\mathbb{P}[\![\sigma]\!]$, that maps data objects to sets of actions, indicating the issuer will restrict his or her use of a given data object, d, to the set of actions $\mathbb{P}[\![\sigma]\!]d$.

A key difference between P3P and DPAL is the scope of policy statements. In both P3P and DPAL, a statement mentioning data object d affects all data objects $d' \leq_D d$. In DPAL, however, "deny" statements also affect all data objects $d'' \geq_D d$. This upward inheritance of denying statements arises precisely because DPAL is designed to answer the $\Box$ queries of service providers. Specifically, if a statement prohibits action a on d, then $d'' \not\Vdash \Box a$.

Unlike P3P, in which policies are the disjunction of their statements, DPAL policies enforce each of their statements. A DPAL policy is the conjunction of its constituent statements. Formally, a DPAL policy, p, over a data hierarchy, D, is interpreted such that

$$\mathbb{P}[\![p]\!]d = \bigcap_{\sigma \in p} \mathbb{P}[\![\sigma]\!]d$$

for each $d \in D$. As with P3P, the Kripke model for p is $\mathfrak{K}_{\mathbb{P}[\![p]\!]}$.

DPAL is an enforcement language because each DPAL policy is invariant under $\Box$. Formally, every DPAL policy p satisfies

$$d \Vdash_{\mathbb{P}[\![p]\!]} \Box a \iff d \Vdash_{\mathbb{P}[\![p]\!]} a$$

for every $d \in D$ and every $a \in A$. This is the "safety" property of [6], recast in our model. Service providers can determine if DPAL policy p permits action a on data object d by determining if $a \in \mathbb{P}[\![p]\!]d$ because

$$a \in \mathbb{P}[\![p]\!]d \iff d \Vdash_{\mathbb{P}[\![p]\!]} a \iff d \Vdash_{\mathbb{P}[\![p]\!]} \Box a$$

LEMMA 5. *Policy p is safe if, for every $d \in D$ and every $a \in A$,*

$$d \Vdash_p \Box a \iff d \Vdash_p a.$$

PROOF. Assume, by way of contradiction, there exists $d_1, d_2 \in D$ with $d_1 \leq_D d_2$, $a \in p(d_2)$, but $a \notin p(d_1)$. Then $d_2 \Vdash_p a$, implying $d_2 \Vdash_p \Box a$. Therefore, $d_1 \Vdash_p a$, which contradicts $a \notin p(d_1)$. □

DPAL does not actually satisfy this property. Obligations attached to "allow" statements restrict the set of permissible actions: actions not meeting obligations are not permitted. However, obligations do not inherit upwards. This violation of $\Box$-invariance creates enforcement loopholes, exploitable to perform otherwise prohibited actions. This peculiarity is also present in EPAL [3, 6]. Enterprises using EPAL as an enforcement language (that is, evaluating $d \Vdash a$ in place of $d \Vdash \Box a$) may be vulnerable to malicious or accidental circumvention of their policies.

4.2 Transitivity of enforcement

The enforcement relation is transitive, so if a DPAL policy enforces a P3P policy, the DPAL policy also enforces the compact version of the P3P policy.

for all $d \in D_p$ (inductively) **do**
  $p(d) \leftarrow q(d)$
  for all $d_0 \in D_q$ (with $d_0 \leq_{D_q} d$) **do**
    if $(\forall d' \leq_{D_p} d)(q(d_0) \not\subseteq p(d'))$ **then**
      $p(d) \leftarrow p(d) \cup q(d_0)$
    end if
  end for
end for

Figure 6: Algorithm for projecting policy q, with well-founded data hierarchy D_q, onto $D_p \subseteq D_q$. Build p by iterating through the data objects in D_q, at each iteration processing a $\leq_{D_p}$-minimal unprocessed data object.

LEMMA 6. *The relation "enforces" is transitive.*

PROOF. Given policies p, q, and r, with nested data hierarchies $D_p \subseteq D_q \subseteq D_r$, respectively, assume r enforces q and q enforces r. We claim r enforces p. Given $d \in D_p$ and positive, simple, modal formula φ, assume $d \Vdash_r \varphi$. Because r enforces q, $d \Vdash_q \varphi$. Because q enforces p, $d \Vdash_p \varphi$, and r enforces p. □

4.3 Summarizing policies

Determining whether a service provider's DPAL policy enforces his or her P3P policy is useful when both policies already exist. Commonly, however, a service provider has already written a DPAL policy and wishes to generate a corresponding P3P policy for publication on his or her web site. The P3P policy he or she wishes to generate is the least permissive policy enforced by his or her DPAL policy.

Def. Given a policy q, with well-founded data hierarchy D_q, and a subset D_p of D_q, the **projection**, p, of q onto D_p is the policy resulting from the algorithm in Figure 6.

To generate a P3P policy, a service provider projects his or her DPAL policy onto P3P's base data schema. P3P cannot express exactly this projection, p, because of its limited action vocabulary. To generate P3P terms, emit each term with a non-trivial intersection with $p(d)$ for some $d \in D_p$. Redundant terms may be eliminated by traversing D_p in a depth-first search and not emitting terms previously emitted during the path back to the root data object.

THEOREM 7 (ENFORCEMENT OF PROJECTION). *For all policies q, with well-founded data hierarchy D_q, and all $D_p \subseteq D_q$, q enforces the projection, p, of q onto D_p.*

PROOF. Given $d \in D_p$, proceed by induction on positive, simple, modal formulae. Given φ, assume $\varphi = a$ for some action $a \in A$ and $d \Vdash_q a$. By the algorithm, $a \in p(d)$, and therefore, $d \Vdash_p a$. The inductive steps for $\wedge$ and $\vee$ are immediate.

Assume $\varphi = \Box\psi$, where ψ is free of modalities, and $d \Vdash_q \Box\psi$. For all $d' \in D_p$ with $d' \leq_{D_p} d$, $d' \Vdash_q \psi$ and, by induction, $d' \Vdash_p \psi$. Therefore, $d \Vdash_p \Box\psi$.

Assume $\varphi = \Diamond\psi$, where ψ is free of modalities, and $d \Vdash_q \Diamond\psi$. Fix $d_0 \in D_q$ such that $d_0 \Vdash_q \psi$. Consider the loop iteration in which d_0 and d are processed. There are two cases. First, assume the "if" condition is met. Thus, $q(d_0) \subseteq p(d)$, $d \Vdash_p \psi$, and therefore, $d \Vdash_p \Diamond\psi$.

Second, assume the "if" condition is not met. Fix $d' \leq_{D_p} d$ such that $q(d_0) \subseteq p(d')$. Thus, $d' \Vdash_p \psi$, and therefore, $d \Vdash_p \Diamond\psi$. Combining conclusions, q enforces p. □

In order to attract the most consumers, a service provider may wish to publish the most restrictive P3P policy, the tightest expressible upper bound, consistent with his or her internal DPAL policy. Projection generates such a policy.

THEOREM 8 (MINIMALITY OF PROJECTION). *For all policies q, with finite data hierarchy D_q, and all $D_p \subseteq D_q$, the projection, p, of q onto D_p is the $\sqsubseteq$-minimal policy with data hierarchy D_p enforced by q.*

PROOF. Assume, by way of contradiction, there exists p', with data hierarchy D_p, such that p is distinct from p', q enforces p', and $p' \sqsubseteq p$. Fix an action, a, and a data object, $d \in D_p$, such that $a \in p(d)$ and $a \notin p'(d)$. Consider the point in the algorithm when a is added to $p(d)$. There are two cases.

First, assume a is added to $p(d)$ in the initialization loop. Thus, $d \Vdash_q a$. Because q enforces p', $d \Vdash_{p'} a$, but this contradicts $a \notin p'(d)$.

Second, assume a is added to $p(d)$ in the iteration of the nested loop when $d_0 \in D_q$ is considered. Let C be a set that chooses an action from $q(d_0) - p'(d')$, for each $d' \leq_{D_p} d$. C is finite because D_p is finite. Let ψ be the conjunction of elements of C. $d \Vdash_q \Diamond\psi$, but $d \not\Vdash_{p'} \Diamond\psi$, because $p(d')$, for each $d' \leq_{D_p} d$, has already been determined. This contradicts q enforcing p'. □

5. CONCLUSION

This paper proposes DAMP, a data-centric model for privacy, which distinguishes between the perspectives of service providers and consumers and is equipped with a modal logic for reasoning about privacy policies. The $\Box$ modality reflects the perspective of service providers; the $\Diamond$ modality reflects the perspective of consumers. By separating these modalities from the underlying privacy promises, we are able to connect privacy promises with privacy enforcement and provide a rigorous foundation for understanding privacy policies.

We validate DAMP by using it as a tool in understanding languages designed to interoperate with P3P, such as APPEL, XPref, and compact P3P policies. In doing so, we discover a surprising property of both APPEL and XPref: each can express dubious privacy preferences, such as "block services that do not telemarket." Moreover, robust preferences are difficult to express in XPref.

Using DAMP, we illuminate the relation between a P3P policy and its compact representation. A compact policy lists the values of certain $\Diamond$ terms in our modal logic. These semantics underpin P3P's algorithm for generating compact policies. We prove P3P policies enforce their compact representations. This enables consumers who accept compact policies to be confident that the underlying full policy conforms to their preferences.

DAMP's enforces relation connects privacy promises (for example, expressed in a P3P policy) with privacy enforcement (for example, embodied in a DPAL policy). A consumer who accepts a privacy promise can be confident the service provider's operative enforcement policy conforms to his or her preferences. Because enforcement is transitive, a consumer who accepts a compact policy enjoys the same assurances about the operative enforcement policy.

Finally, we complete the toolset by presenting an algorithm for summarizing detailed enforcement policies using a given vocabulary. Enterprises can use this algorithm to

translate their DPAL policies into P3P policies for announcement on their web sites. We prove the summary policy is enforced by the detailed policy and is the most restrictive such policy. This provides enterprises with the tools to ensure their privacy enforcement mechanisms actually enforce their announced privacy policies.

6. REFERENCES

[1] M. S. Ackerman, L. F. Cranor, and J. Reagle. Privacy in e-commerce: examining user scenarios and privacy preferences. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 1–8. ACM Press, 1999.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 629–639. ACM Press, 2003.

[3] M. Backes, M. Dürmuth, and R. Steinwandt. An algebra for composing enterprise privacy policies. In *European Symposium on Research in Computer Security (ESORICS)*. Springer Lecture Notes in Computer Science 3193, 2004.

[4] M. Backes, G. Karjoth, W. Bagga, and M. Schunter. Efficient comparison of enterprise privacy policies. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 375–382. ACM Press, 2004.

[5] M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *European Symposium on Research in Computer Security (ESORICS)*, pages 101–119. Springer Lecture Notes in Computer Science 2808, 2003.

[6] A. Barth, J. C. Mitchell, and J. Rosenstein. Conflict and combination in privacy policy languages. In *Proceedings of the 2004 Workshop on Privacy in the Electronic Society*. ACM Press, 2004.

[7] M. Bishop. *Computer Security: Art and Science*. Addison Wesley Professional, 2003.

[8] S. Byers, L. F. Cranor, and D. Kormann. Automated analysis of P3P-enabled web sites. In *Proceedings of the 5th International Conference on Electronic Commerce*, pages 326–338. ACM Press, 2003.

[9] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[10] J. Clark and S. DeRose. XML path language (XPath), 1999. http://www.w3.org/TR/xpath/.

[11] J. Crampton. On permissions, inheritance and role hierarchies. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pages 85–92. ACM Press, 2003.

[12] L. F. Cranor. *Web Privacy with P3P*. O'Reilly and Associates, Inc., 2002.

[13] L. F. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification, 2002. http://www.w3.org/TR/P3P/.

[14] J. Glasgow, G. Macewen, and P. Panangaden. A logic for reasoning about security. *ACM Trans. Comput. Syst.*, 10(3):226–264, 1992.

[15] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[16] C. Jensen and C. Potts. Privacy policies as decision-making tools: an evaluation of online privacy notices. In *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, pages 471–478. ACM Press, 2004.

[17] G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002.

[18] R. Pucella and V. Weissman. Reasoning about dynamic policies. In *Foundations of Software Science and Computation Structures (FOSSACS)*, 2004.

[19] J. Reagle and L. F. Cranor. The platform for privacy preferences. *Commun. ACM*, 42(2):48–55, 1999.

[20] M. Schunter, P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (EPAL 1.1), 2003. http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/.