

Privacy System encoded using EPAL 1.2

Michael J. May
University of Pennsylvania

August 2004

1 Privacy Systems as EPAL

The privacy system formalism as described above is useful by itself, but it is helpful to model its behaviors through a concrete implementation. The implementation helps remove some of the amor- phism that comes from dealing with a pure mathematical entity. It also helps the viewer see how data can flow between principals and how rules affect the system as a whole. Our implementation model uses the Enterprise Privacy Authorization Language (EPAL) to do that modelling.

EPAL is designed as a privacy policy implementation language, one that allows policy writers to define terms, conditions, and rules about what can and can not be done. Authors define all the important terms in the language and the rules that govern actions. The EPAL language provides a structure for those pieces and an architecture that can interpret those rules appropriately.

The abstract definition of the privacy system gives the functions and relations that must be applied to determine the outcome of a proposed event. Those functions and relations allow us to define an EPAL vocabulary document that gives us the terms that can be used by a specific privacy system example. There is not enough information in the privacy system definition to write any meaningful policy rules or conditions in EPAL. Rules and conditions require more information about the values in the system.

The mapping of privacy system terms to EPAL terms is as follows.

- Names - At each privacy system event there are at least two principals being considered - one or more actors and a subject. EPAL rules care about user roles and names, but the privacy system formalism is concerned solely with permissions. As such, there is no need for user names or classes in the vocabulary. We can make decisions based solely on the permissions that a principal has. The only exception to this rule is the W relation that defines who can set policy for a subject. We allow any subject to set policies about itself - a property that is most easily checked by making sure the name of the subject equals the name of the setter. This can also be accomplished by creating a special sigma permission equivalent to the “owner” permission in the Graham-Denning model.
- Events in a privacy system are modelled as the actions of EPAL rules. They are generic descriptions of what is going on - transfer, creation, policy establishment, and action. Rules delineate then which conditions and obligations are checked for each particular event.
- The parameters that determine the outcome of events - permissions and objects - are bound together in containers. All the data necessary for the evaluation of an event must be available for the EPAL system to make a decision. The machinery of the privacy systems functions and relations are taken care of by conditions and rules in the EPAL policy.
- Privacy systems have no notion of purpose - they allow or disallow actions based solely on permissions and time. In the sigma abstraction, however, there is room for the notion of

purpose. Thus, a sigma permission can be viewed as a tuple of the form (Action, Purpose, Time). When the T, U, V, W are evaluated, they can take all three of those parameters into account. EPAL has a purpose parameter for each of its rules, but conditions can not access that parameter. Therefore it is necessary to either include the purpose again in a data container or to just parameterize over different purposes in the rules themselves. In any case it is possible to tailor rule decisions based on purpose.

- The final decision for events are made by the EPAL rules. An EPAL rule has five parameters: User, Purpose, Action, zero or more condition(s) to be checked. When the rule evaluator finds a match, it returns both the rule's decision (approve, deny) and some obligations that rule requires. Obligations are meant to be actions that the user must perform as a consequence of being allowed to do the action. They are usually outside the scope of the EPAL rules themselves (i.e. retention requirements, getting consent of a subject).

One of the barriers in translating the privacy system abstraction to EPAL was in managing user names and privileges. EPAL provides the option of making user names and user classes that are parameters to rules. Users can either be named specifically in a vocabulary and then placed into a hierarchy using the "parent" property or just trusted to submit requests that accurately identify which user class(es) they belong to. The principals in a privacy system, however, do not fit neatly into user classes. Principals can change roles between interactions, being the subject of one event and the actor in another. Additionally, since the privacy system is entirely permissions based there is little use for the naming capabilities of EPAL. Unless an example is concretized to the point that it names its principals and their roles an EPAL translation will essentially ignore names.

Another tradeoff in translation is the choice of where to put the T function's granularity reducing operator. If, for example, the T function is required to return an object that is accurate only to the level of city, there are two ways to express that using EPAL.

- Create a condition that checks that all fields of finer granularity than "City" are left blank. A rule that references that condition will then only fire when the object being presented is in a valid state.
- Create an obligation called "City-Only" and include it in a rule. The requestor would know how to interpret that obligation and fulfill it before executing the requested event

Each of the above options has positives and negatives as we shall describe.

The first has the positive that the rule will not allow an event to happen unless the object submitted is in proper form. It provides an invariant that rules only allow events which have objects already in allowed forms. The negative to the first option is that the requestor may not know a priori what level of accuracy is required to gain approval and may have to guess and try several times to discover the allowed level.

The second has the positive that the requestor gets a more informative response - a "yes with obligation" that lets the object be transformed to the proper form by the requestor. Another positive is that the rule maker does not have to know the format of the data object being presented. It introduces a layer of abstraction that lets rules be meaningful in more circumstances. The negative to the second option is that an event log for the rules will show approval of events with objects that were not in valid forms. The requestor or a third party must keep a log of the transformation and prove that it was transformed appropriately. Using this option also requires the requestor to understand the exact intent of each obligation, requiring the rules maker to be very specific when designing obligations.

The above discussion brings out an essential tradeoff in designing EPAL policies: How much should be in the policy and how much should be left to other parts of the architecture. There is some gain in making very specific policies but that breaks away from the idiom of classical legal

privacy policy and moves it towards the policy equivalent of an access control system. Since privacy systems are designed with an eye towards classical access control systems, the natural move is to design access-control style policies more than higher level legal privacy policy style ones.

When designing conditions, another tradeoff presents itself. Conditions can be of arbitrary complexity, but always must result in a boolean value. Complex conditions may have many clauses that can be put under logical AND, OR, and NOT. Whether to create large, complex conditions tailored to specific rules or to leave them as almost atomic conditions is another decision the policy designer must make. It trades tailoring against reusability, a classical software engineering requirement.

2 Vocabularies for Direct Permissions, Direct Time Limited Permissions, and Sharing With Partners Examples

Since both examples are very similar in their functionality, a single vocabulary suffices for both the DP and DTLP examples. The vocabulary is based heavily on the one for generic privacy systems and shows how easy it is to adapt the generic vocabulary to a more specific case. The only major addition is the addition of Time attributes to events and an Expiry attribute to permissions.

There are special containers for permissions in the transfer and set policy events since there are multiple actors in both cases. The action and creation events can use a generic container that has only one actor's information.

The vocabulary for the sharing with partners example adds containers for the relation `partner` and the set $\mathcal{A}_{\text{indir}}$.

With enough abstraction one container could be used for all the events, but that would require the removal of descriptive names like "Actor", "Setter", and "Recipient". The privacy system vocabulary could just be extended to include many more possible examples and thereby remove the requirement for multiple vocabularies. For example, we could have included the containers for $\mathcal{A}_{\text{indir}}$ and `partner` in all the vocabularies and just ignored them in the policy files that were not concerned with them. Ease of understanding and a desire for more concise examples motivated our choice to use multiple vocabularies and containers with tailored names.

3 Atomic Conditions

Translating the privacy systems operations into EPAL exposes the atomic conditions that are involved in every event that a privacy system executes. Exposing those conditions helps us better understand the semantics of each event.

The number of atomic conditions that must be checked varies between examples and events. For example the Set Policy event in the Direct Permissions example needs to check only one atomic check - that the name of the setter equals the name of the subject. At the other extreme the Set Policy event under the Sharing With Partners example in which a principal grants an indir right to its partner requires six atomic checks - a name check for each of three principals, a check that the recipient is really a partner of the setter, a check that the setter has direct permission a priori, and a check that the setter is only granting indirect permission. As examples get more complex and have more requirements the policies needed to describe their interactions grow in length and complexity.

4 Differences between EPAL and Privacy Systems

With EPAL a query is sent in with general information - who is asking, what they are asking to do, and the data that will be acted on. The policy file contains information that is used to

resolve the query - rules that determine what is ok and what is not ok. But with a privacy system most of the work is already encoded in the manner of the system. In fact the queries all already contain sigmas that are the permissions themselves. The EPAL privacy system implementation just takes information that already should suffice for the decision based solely on the semantics of the privacy system formalism itself. As such, the policy spends most of its concern on checking that the permissions provided are correct and being used in a proper manner. Introducing complicated policies in a privacy system is done by distributing sigmas in an interesting way - giving to one party or class of users and not to another. But those permissions are not reflected in the privacy system itself - a user can give a permission to another principal who then can present it to the EPAL policy to ask before using that permission. In that sense the permissions are tokens that can be used for accessing rights that have been granted by a user and the policy is just a monitor that effects some policy on how those actions are done.

When converting the AdLoc EPAL vocabulary and policies to a privacy systems style format we must decide what belongs in the policy and vocabulary themselves and what should be left to the user to distribute. For example, if the user wants to allow customer service representatives to use location data to help them over the phone, they will have to give a sigma to the customer service representative (or just all customer service representatives who are currently working on their behalf) allowing them to access the data. Should that permission be reflected in the policy file? Or should the customer service representative hold onto a token that can be presented to the system when asking for permission? It seems that latter makes more sense than the former.

This leads us to an interesting difference of architecture between a privacy system and a standard EPAL policy. The EPAL policy is a know-all and end-all. It declares everything that is allowed and should be able to determine who can do what just by looking at the who and the what of the request. No other credential or information must be included to finish the request. In a privacy system, however, principals are given sigmas (tokens?) that allow them to do certain actions. A monitor checks that those tokens are valid and are being used in a proper manner, but the policy is truly generic and dumb. Users who lose their tokens must get new ones before they can do any action. They need a proof that they are authorized to do the action.