A Multi-Privacy Policy Enforcement System

Kaniz Fatema, David W Chadwick and Stijn Lievens,

University of Kent, Canterbury, Kent, UK
{k.fatema, D.W.Chadwick, S.F.Lievens}@kent.ac.uk

Abstract. With the increase in the number of electronic services and the number of users, concerns about the privacy protection of electronic data are growing day by day. Organisations are facing a huge pressure to assure their users about the privacy protection of their personal data. Organisations need to include the privacy policies of their users when deciding who should access their personal data. The user's privacy policy will need to be combined with the organisation's own policy, as well as policies from different authorities such as the issuer of the data, and the law. The authorisation system will need to ensure the enforcement of all these policies. We have designed a system that will ensure the enforcement of multiple privacy policies within an organisation and throughout a distributed system.

Keywords: Privacy Policy, AIPEP, Master PDP, Conflict Resolution, Sticky Policy.

1 Introduction

Many web sites today collect PII (Personal Identity Information) such as name and address from users through online registration, surveys, user profiles, and online order fulfilment processes etc. Also different personal data such as educational record, health data, credit card information and so on are collected by different organisations in order to provide consumers with services. An example of such service is an online job agency where people post their CV in order to get the opportunity to hunt for jobs worldwide. Once released, users lose control over the fate of their personal data. But personal data items like CVs which contain sensitive personal information may invite not only job offers but also unwanted ID theft. Losing PII has serious consequences from significant financial loss to becoming a suspect of a worldwide crime which is committed with stolen ID. It is not uncommon for someone to get arrested due to a crime committed by an identity thief using that person's ID [1]. In the UK, the number of ID thefts is an alarming 19.86% higher in the first quarter of 2010 [2] compared with the same period in 2009. About 27,000 victims were recorded by CIFAS member during the first 3 months of 2010 [2]. As a consequence concerns for the privacy of electronic private data are also rising day by day [3, 4]. Hence the necessity for more technical control over personal data collected online in order for users to gain more confidence and trust about the use of their personal data. Technical controls will also help to protect personal data from being misused and as well as enforce privacy laws so that personal data loss from reputed organisations like HSBC bank [5] or Zurich Insurance [6] may be avoided.

Policy based systems are now well established [7, 8]. They rely on an application independent policy decision point (PDP) to make authorization decisions, and an application dependent policy enforcement point (PEP) to enforce these decisions. The model assumes that all the policies are written in the same language and are evaluated by a single PDP. However in a federated identity management system we cannot assume that every service provider (SP) and identity provider (IdP) will use the same policy language for specifying their rules. This is because different policy languages support different rule sets and hence support different requirements. Today we have many examples of different policy languages e.g. XACMLv2 [9], XACMLv3 [10], PERMIS [11], P3P [12], Keynote [13] etc. and even more PDP implementations. Differences between languages are for example that XACMLv2 does not support delegation of authority whilst XACMLv3 and PERMIS do. The XACML policy language assumes a stateless PDP hence cannot support state based policy rules such as separation of duties (SoD), whilst PERMIS is state based and can support both dynamic and static SoD. No version of XACML supports credentials (the concept is simply not mentioned in the standard), whereas Keynote does and uses the same language to describe both credentials and policy rules. PERMIS also supports credentials and has a credential validation service [14]. P3P is designed specifically to express privacy policies, whereas the others were designed as access control or authorization policy languages. It is simply not possible to construct policies that satisfy every requirement using a single policy language or PDP. Therefore we need an infrastructure that can support multiple PDPs and multiple policy languages.

Obligations are actions that must be performed when a certain event occurs. When the event is an authorization decision, then the obligations are actions that must accompany this decision. Some obligations need to be performed before the decision is enforced, some after the decision has been enforced, and some along with the enforcement of the authorization decision [15]. We propose an obligations service with a standard interface that can be called from multiple places in an application, with one of these places being the application independent authorization infrastructure.

Private data should be protected by the policy of its owner. We have used the sticky policy paradigm [16] to ensure that the private data is stuck with the policy not only within the system but also while leaving the system. In attribute based access control (ABAC), the PDP makes its decisions based on the attributes of the subject, requested action, resource and environment. We propose a credential validation service (CVS) [14] that is responsible for validating subject credentials, extracting the valid attributes from them and discarding the rest.

Finally, we need to keep authorisation decision making as simple as possible for application developers. The complexity should be hidden behind a standard interface and orchestration of the different authorisation components should be done by the infrastructure itself. We propose an application independent PEP (AIPEP) for this.

In this paper we propose an advanced multi-policy authorization infrastructure that will provide privacy of personal data. The rest of the paper is structured as follows. Section 2 reviews related research. Section 3 discusses the architecture and components of the proposed system. Section 4 discusses the Sticky Policy implementation strategy and Section 5 describes the conflict resolution policy. Some use case scenarios are provided in Section 6. Details of our implementation to date are provided in Section 7 and finally Section 8 concludes by discussing our future plans.

2 Related Research

IBM's security research group has performed research on privacy protection of customer's data collected by enterprises [17-21]. They used the sticky policy paradigm where personal data is associated with its privacy policy and they are passed together when exchanging data among enterprises [17-19,21]. But they did not provide a way to accommodate different policy languages. Also the obligations they are providing are just activity names such as 'log', 'notify', 'getConsent' etc. [8]. They also did not provide a way to actually enforce the obligation which our system does.

HP [22, 23] have also been working on providing privacy to PII by enforcing obligations. They have also provided a way of transmitting encrypted confidential data with obligations to other parties by obfuscation of the data [22]. Nevertheless, the work has only described obligations related to privacy and does not provide a uniform solution to both access control and privacy. Their work does not consider policies from different authorities nor does it integrate multiple policy languages.

Qun Ni et al [24, 25] have defined the privacy related access control model P-RBAC to support privacy related policies. This model theoretically associates data permissions with purposes, conditions and obligations. However, the model is too complex to be implemented practically.

While private data can move between organizations with its sticky privacy policy, the enforcement of the privacy policy is only ensured if either all the organizations support the same policy language, which is not feasible in practice, or the organizations have support for multiple policy languages. Our model supports multiple policy languages as well as policies from different authorities.

It has been claimed [26, 27] that the privacy policy defined by the owner of data should have the highest priority. But the fact is that the Law should have the highest priority. No one should be able to break the Law. No other previous work has focused on this issue. In our system we have implemented the Law PDP by converting the legal requirements into an XACML policy and this Law PDP is always given the highest priority. For example if there is a court order for seeing someone's personal data neither the person nor the data controller can deny access to the data. To the best of our knowledge, no previous work has been concerned with integrating the policies of the law, data subject or data controller which is done by our system.

3 The Authorisation System

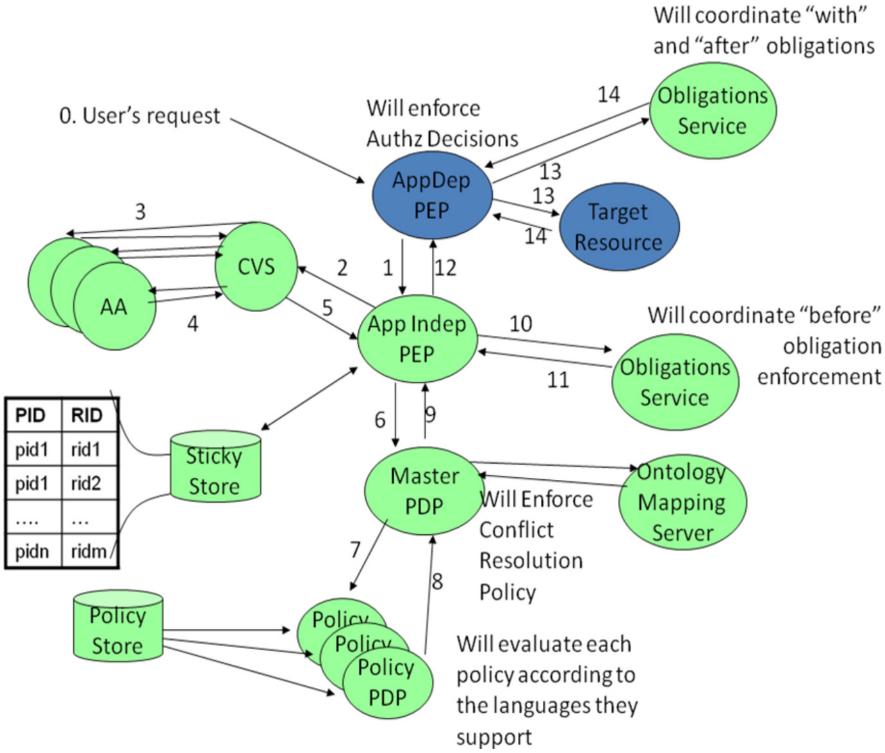
In order to satisfy the various requirements presented above we introduce several new components into the privacy preserving advanced authorization infrastructure.

Firstly we introduce an application **independent policy enforcement point, the AIPEP**. The AIPEP is responsible for coordinating the actions of the various components of the application independent authorization infrastructure. When the AIPEP receives either an authorization decision query message (step 1 in figure 1), it first calls the CVS to validate any credentials that are contained in the message (step 2 in figure 1). If the message contains a sticky policy/ies (see figure 2) then this/these will be stored in the policy store. The AIPEP retains a manifest which records which CVSs and PDPs are currently spawned and which policies each is configured with. The AIPEP tells the Master PDP which set of spawned PDPs to use for a particular authorization decision request.

The **Credential Validation Service (CVS)** is the component that validates credentials by checking that each credential issuer is mentioned in the credential validation policy directly, or that the credential issuer has been delegated a privilege by a trusted Attribute Authority (AA) either directly or indirectly (i.e. a chain of trusted issuers is dynamically established controlled by the Delegation Policies of the Source of Authority and the intermediate AAs in the chain). The Credential Validation Policy, written by the SOA, contains rules that govern which attributes different AAs are trusted to issue to which user group, along with a Delegation Policy for each AA.

In order to evaluate multiple authorization policies in different languages we introduce a new conceptual component called the **Master PDP**. The Master PDP is responsible for calling multiple PDPs (step 7) as directed by the AIPEP, obtaining their authorization decisions (step 8), and then resolving any conflicts between these decisions, before returning the overall authorization decision and any resulting obligations to the AIPEP (in step 9). Each of the policy PDPs supports the same interface, which is the SAML profile of XACML. This allows the Master PDP to call any number of subordinate PDPs, each configured with its own policy in its own language. This design isolates the used policy languages from the rest of the authorization infrastructure, and the Master PDP will not be affected by any changes to any policy language as it evolves or by the introduction of any new policy language. Of course, new policy languages will require new PDPs to be written to interpret them, and these new PDPs will require new code in the PDP/CVS factory object so that it knows how to spawn them on demand. But this is a one-off occurrence for each new policy language and PDP that needs to be supported by the infrastructure.

The **policy store** is the location where policies can be safely stored and retrieved. If the store is trusted then policies can be stored there in an unsecured manner. If the store is not trustworthy then policies will need to be protected e.g. digitally signed and/or encrypted, to ensure that they are not tampered with and/or remain confidential. When the AIPEP stores a policy in the policy store, it provides the store with the StickyPolicy element, see Figure 2, and is returned a locally unique storage reference to the policy, called the policy store handle (PSH). The AIPEP can subsequently use this handle to pass the policy to the PDP/CVS factory in order to spawn a new PDP or CVS. This design cleanly separates the implementation details of the policy store from the rest of the infrastructure, and allows different types of policy store to be constructed e.g. built on an LDAP directory or RDBMS.


Will coordinate "with" and "after" obligations
0. User's request
Will enforce Authz Decisions
14
Obligations Service
AppDep PEP
13
13
Target Resource
14
3
CVS
AA
2
1
12
5
4
App Indep PEP
10
Will coordinate "before" obligation enforcement
Obligations Service
11
Sticky Store
6
9
Master PDP
Ontology Mapping Server
Will Enforce Conflict Resolution Policy
7
8
Policy Store
Policy
Policy
Policy PDP
Will evaluate each policy according to the languages they support

PID	RID
pid1	rid1
pid1	rid2
….	…
pidn	ridm

Fig. 1. The privacy preserving advanced authorization system

Obligations may be required *before* the user's action is performed, *after* the user's action has been performed, or simultaneously *with* the performance of the user's action [15]. We call this the *temporal type* of the obligation. Examples are as follows: before the user is given access get the consent of owner; after the user has been given access, email the data owner that his/her data is accessed; simultaneously with the user's access, write on the log the activities he/she is doing.

According to the XACML model, each obligation has a unique ID (a URI). We follow this scheme in our infrastructure. Each obligations service is configured at construction time with the obligation IDs it can enforce and the obligation handling services that are responsible for enacting them. It is also configured with the temporal type(s) of the obligations it is to enforce. When passed a set of obligations by the AIPEP, the obligations service will walk through this set, ignore any obligations of the wrong temporal type or unknown ID, and call the appropriate obligation handling service for the others. If any single obligation handling service returns an error, then the obligations service stops further processing and returns an error to the AIPEP. If all obligations are processed successfully, a success result is returned. Each of the obligations enforced by the AIPEP must be of temporal type *before*. **The Ontology**

Mapping Server is a service which returns the relationship between two different terms. The ontology is held as a lattice, and the server will say if one term dominates the other in the lattice or if there is no domination relationship between them. The Master PDP will call this server to determine the relationship of the subjects / roles of PDP rule so that the specificOverrides DCR (see later) can be implemented.

```
 <xs:element name="StickyPad" type="StickyPADType"/>
 <xs:complexType name="StickyPADType">
   <xs:sequence>
     <xs:element ref="DataResource"/>
     <xs:element name="DataResourceTypes" type="ResourceTypes"/>
     <xs:element ref="StickyPolicy" maxOccurs="unbounded"/>
     <xs:element ref="ds:Signature" minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>

 <xs:complexType name="ResourceTypes">
   <xs:sequence>
     <xs:element name="ResourceType" type="xs:anyURI" maxOccurs="unbounded"/>
   </xs:sequence>
 </xs:complexType>

 <xs:element name="StickyPolicy" type="StickyPolicyType"/>
 <xs:complexType name="StickyPolicyType">
   <xs:sequence>
     <xs:element name="PolicyAuthor" type="saml:NameIDType"/>
     <xs:element name="PolicyResourceTypes" type="ResourceTypes" />
     <xs:element ref="PolicyContents"/>
   </xs:sequence>
   <xs:attribute name="PolicyID" type="xs:anyURI" use="required"/>
   <xs:attribute name="PolicyLanguage" type="xs:anyURI" use="required"/>
   <xs:attribute name="PolicyType" type="xs:anyURI" use="required"/>
   <xs:attribute name="TimeOfCreation" type="xs:dateTime" use="required"/>
   <xs:attribute name="ExpiryTime" type="xs:dateTime" use="optional"/>
 </xs:complexType>
   <xs:element name="PolicyContents" type="AnyXMLType"/>
 <xs:element name="DataResource" type="AnyXMLType"/>

 <xs:complexType name="AnyXMLType" mixed="true">
   <xs:sequence>
     <xs:any    minOccurs="0"    maxOccurs="unbounded"    namespace="##any"
processContents="lax">
       <xs:annotation>
         <xs:documentation>
           Any xml content is allowed in this element.
         </xs:documentation>
       </xs:annotation>
     </xs:any>
   </xs:sequence>
 </xs:complexType>
 </xs:schema>
```

Fig. 2. Sticky Policy and StickyPAD Schema

4 Sticky Policy Implementation

Figure 2 provides a schema for sticky policies. A sticky policy comprises:

- The policy author i.e. the authority which wrote the policy.
- The globally unique policy ID
- The time of creation of the policy and optional expiry time.
- The type(s) of resource(s) that are covered by this policy.
- The type of policy this is (see Figure 3).
- The policy language.
- The policy itself, written in the specified policy language.

Any number of sticky policies can be stuck to a data resource in either an application dependent manner e.g. as a <Condition> in a SAML attribute assertion, or by using the StickyPAD (**sticky p**olicy(ies) **a**nd **d**ata) XML structure that we have defined in Figure 2. The policies should be stuck to the data by using a digital signature. This could be by using the XML <ds:Signature> structure in the StickyPAD and SAML attribute assertion, or it could be externally provided e.g. by using SSL/TLS when transferring the data and policy across the Internet. It is the responsibility of the sending PEP to create the equivalent of the StickyPAD structure when sending data with a sticky policy attached, and the receiving PEP to validate its signature when it receives the message in step 0 of figure 1. The PEP should then parse and unpack the contents and pass the sticky policy to the AIPEP along with the authorization decision request (step 1 of figure 1).

The **sticky store** holds the mapping between sticky policies and the resources to which they are stuck. This is a many to many mapping so that one policy can apply to many resources and one resource can have many sticky policies applied to it. The design requires that each policy has a globally unique Policy ID (PID) and each resource has a locally unique resource ID (RID). The PID was chosen to be globally unique for performance reasons, so that when a sticky policy is moved from system to system, the receiver can determine if it needs to analyse each received policy or not. Already known PIDs don't need to be analysed, whereas unknown PIDs will need to be evaluated to ensure that they can be supported, otherwise the incoming data and sticky policy will need to be rejected. The RID is locally unique and may be constructed by applying a one way hash function such as SHA1 to the resource. We currently do not have a requirement to pass the RID from system to system so each system can compute its own.

5 Conflict Resolution Policy

Our system will include many different PDPs each with policies from different authorities and possibly written in different languages. As a consequence a mechanism is needed to combine the decisions returned by these PDPs and resolve any conflicts between them. We propose a Master PDP which is the component responsible for combining the decision results returned by the subordinate PDPs and resolving the conflicts among their decisions.

The Master PDP has a conflict resolution policy (CRP) consisting of multiple conflict resolution rules (CRRs). The default CRP is read in at program initialisation time and additional CRRs are dynamically obtained from the subjects' and issuers' sticky policies. Each conflict resolution rule (CRR) comprises:

- a condition, which is tested against the request context by the Master PDP, to see if the attached decision combining rule should be used,
- a decision combining rule (DCR),
- optionally an ordering of policy authors (to be used by FirstApplicable DCR)
- an author and
- a time of creation.

A DCR can take one of five values: FirstApplicable, DenyOverrides, GrantOverrides, SpecificOverrides or MajorityWins which applies to the decisions returned by the subordinate PDPs. The DCRs will be discussed shortly.

The Master PDP is called by the AIPEP and is passed the list of PDPs to call and the request context. From the request context it will get the information such as requester, requested resource type, issuer and data subject of the requested resource. The Master PDP has all the CRRs defined by different authors as well as a default one. From the request context it knows the issuer and data subjects and so can determine the relevant CRRs. It will order the CRRs of law, issuer, data subject and holder sequentially. For the same author the CRRs will be ordered according to the number of conditions. For example the order of CRRs for a data subject can be

CRR1= if (resourceType=PII, requester=myfriend, requestDate > 10.12.2010) DCR= DenyOverride

CRR2=if (resourceType=PII, requester=myemployer) DCR=GrantOverride

CRR3=if (resourceType=PII) DCR=MajorityWins

Here the CRR with the most conditions will come first. All the conditions of a CRR need to match with the request context for it to be applicable. The CRR from the ordered CRR queue will be tested one by one against the request context. If the CRR conditions match the request context the CRR is chosen. If the CRR conditions do not match the request context the next CRR from the queue will be tested. The default CRR (which has DCR=DenyOverrides) will be placed at the end of CRR queue and it will only be reached when no other CRR conditions match the request context. The PDPs are called according to the DCR of the chosen CRR.

Each PDP can return 5 different results –Grant, Deny, BTG, NotApplicable and Indeterminate. NotApplicable means that the PDP has no policy covering the authorisation request. Indeterminate means that the request context is either mal-formed e.g. a String value is found in place of an Integer, or is missing some vital information so that the PDP does not currently know the answer.

BTG (Break the Glass) [28] means that the requestor is currently not allowed access but can break the glass to gain access to the resource if he so wishes. In this case his activity will be monitored and he will be made accountable for his actions. BTG provides a facility for emergency access.

If DCR=FirstApplicable the CRR is accompanied by a precedence rule (OrderOfAuthors) which says the order in which to call the PDPs. For example, if (resourceType=PII, requestor=data subject) DCR=FirstApplicable, OrderOfAuthor=law, dataSubject, holder. The Master PDP calls each subordinate

PDP in order (according to the order of authors), and stops processing when the first Grant or Deny decision is obtained.

For SpecificOverrides the decision returned by the PDP containing a rule with a more specific subject/ resource has priority over the PDP with a rule containing less specific subject/resource. As the master PDP does not have the rule the PDP needs to return the rule together with the decision in order to determine which PDP has the most specific subject/resource. The Master PDP will call the Ontology Mapping Server to determine which of the returned rules has the most specific subject first. If multiple PDP rules have the most specific subject the Master PDP will call the Ontology Mapping Server again to find the most specific resource among the rules having the most specific subjects. If multiple PDP rules have the same most specific subject and resource the decision of PDP with the latest creation time will be chosen. If any of the PDP does not return a rule but a decision only then it is not possible to implement SpecificOverrides as there is no way to determine whether the non rule returning PDP had the most specific subject or not. In that case a default rule (Deny Override) will be implemented as a fallback strategy.

For DenyOverrides and GrantOverrides the Master PDP will call all the subordinate PDPs and will combine the decisions using the following semantics:

- DenyOverrides – A Deny result overrides all other results. The precedence of results for deny override is Deny>Indeterminate>BTG>Grant>NotApplicable.
- GrantOverrides – A Grant result overrides all other results. The precedence of results for grant override is Grant>BTG>Indeterminate>Deny>NotApplicable

When a final result returned by the Master PDP is Grant (or Deny) the obligations of all the PDPs returning a Grant (or Deny) result are merged to form the final obligation.

For MajorityWins all the PDPs will be called and the final decision (Grant/Deny) will depend on the returned decision of majority number of PDPs. If the same numbers of PDP return Grant and Deny then Deny will be the final answer. If none of the PDP return Grant/Deny then Indeterminate will override NotApplicable.

Initially the system will have the law and controller PDPs running as these two are common for all request contexts. Based on the request context the issuer and the data subject's PDP may be started.

6 Use Case Scenarios

Mr K wants to get service from the X-Health Centre and for that he has to be registered at the X-Health Centre by authenticating himself with his ID. During the registration process he is also presented with a consent form where he indicates with whom he is prepared to share his medical data. This form includes tick boxes such as:

1. Registered Dr/Consultant of other Organisation and a place where the name of the doctor can be written if it is known. If this box is ticked and no Dr's name is specified then the consent will be for any Dr in general.
2. Health Insurance Company (with a place for specifying the names of the company or can say all)

3. Research organisation/ researcher. (A note will say that all the medical data used for research purpose will be anonymised or encoded.)
4. Other organisations for promotional offers. Other organisations can for example be organisations offering samples and promotions for new born babies and their parents. In this case not all of the medical record will be available to the interested companies. It may be only the information that this person has recently become a parent. What portion of medical data will be available will be determined by the organisation's policy.
5. Other person (a place for specifying the name of the person.)

Mr K has done registration with a Health Insurance Company (HIC1) to share his treatment cost. So he puts a tick on box 2 only and mentions HIC1 there and finishes his registration with X Health Centre.

Here it is mentionable that the Health Insurance Company will not have access to all medical records of the patient. The policy of Health Centre will decide about what portion of the medical data is sufficient and available to Health Insurance Company.

The HIC1 submits a request for the medical record of the data subject to the X-Health Centre. The Master PDP of X-Health Centre's authorisation system consults the CRRs of Law, issuer, data subject and holder sequentially. A law CRR says if resourceType=MedicalData then the DCR is DenyOverride. So this DCR is chosen and all the PDPs are consulted:

- The law PDP returns decision N/A.
- The issuer (health Centre) PDP returns decision N/A.
- The data subject PDP returns decision grant.

The final result is thus grant. The medical data is passed to HIC1 together with the policies from the data subject and the issuer. The issuer has two PDPs. One PDP has the internal access control rules such as only doctors are allowed to view the treatment files and doctors are not allowed to view the billing info and administrative persons are allowed to view the billing info only and no treatment info and another PDP says which external person are allowed to view the data such as the patient, doctors from another organisation so on. The PDP rules of data subject along with the issuer's PDP rules containing only the external rules are sent to HIC1.

After receiving the medical data and PDP rules the receiving application will make a call to the authorisation system of HIC1 to see whether it can store the data. The authorisation system will reply grant with the obligation to start two new PDPs with the received policies and as a result the data is stored and the two new PDPs are started at HIC1's site, one for the data subject (Mr. K) and one for the issuer (X Health Centre). At HIC1's site the law and holder's (HIC1) PDPs already exist.

HIC1 updates its records periodically and whenever the patient contacts it for clearing a payment. If the patient changes his PDP rules in the meantime by changing his preferences at the Health Centre the new policies are transferred to the HIC1 while transferring the data.

Mr K did not allow researcher to view his medical record now. The researcher Mr R asks for medical record at the HIC1's system and is rejected by the data subject's PDP.

Mr K now changes his rules at the site of X-Health Centre and ticks at the box 3 to allow access to data by the researcher. Mr K's PDP at the X-Health Centre is updated with the new rules saying researchers are allowed to view his medical data and there will be an “before” obligation added to it saying the data to be anonymised. When the HIC1 updates the data also gets the new PDP rules with it and updates the PDP rules at its site. If a researcher now asks for access at the HIC1’s site all other PDPs will return N/a and data subject's PDP will return grant with a “before” obligation to anonymise the data. This obligation will be passed to the Obligations Service of PEP. If this obligation can be enforced successfully a grant decision will be returned and the anonymised data will be passed to the researcher. If the obligation to anonymise the data can’t be enforced a deny decision will be returned to the researcher. It is mentionable that a researcher should be authenticated before granting access to the data. To be authenticated the researcher should have a “researcher” role provided by a trusted research organisation (eg. a University).

7 Implementation Details

Our advanced authorization infrastructure is implemented in Java, and is being used and developed as part of the EC TAS³ Integrated Project (www.tas3.eu). The first beta version is available for download from the PERMIS web site[1]. This contains the AIPEP, CVS, the Obligations Service, a stub Master PDP, a policy store, and multiple PDPs of different types.

A number of different obligation handling services have been written that are called by the obligations service, and these can perform a variety of tasks such as write the authorization decision to a secure audit trail, send an email notification to a security officer, and update the internal state information (called retained ADI in ISO/IEC 10181-3 (1996)). We have implemented state based Break The Glass (BTG) policies [27] using the AIPEP, the obligations service and a stateless PDP. A live demo of BTG is available at http://issrg-testbed-2.cs.kent.ac.uk/. The performance of the obligation state handling BTG wrapper adds between 10% and 200% overhead to the performance of a stateless PDP that does not support BTG. The large overhead is caused because the stateless PDP has to be called more than once in some circumstances. A paper presenting the complete results is currently under preparation.

We have constructed an ontology mapping server, which, when given two class names (such as Visa card and credit card) will return the relationship between them. The output says if either node is more specific than the other or if no such relationship exists between them. The Master PDP will call this ontology mapping server to determine the relationship of the subjects / roles of PDP rule so that specificOverrides DCR can be implemented.

The authorization infrastructure has been tested with three different PDPs: Sun’s XACML PDP[2], the PERMIS PDP[3] and a behavioral trust PDP from TU-Eindhoven[4]. Each of these PDPs uses a different policy language. Sun’s PDP uses the XACML

[1] Advanced authz software available from http://sec.cs.kent.ac.uk/permis/downloads/Level3/standalone.shtml

[2] Sun’s XACML PDP. Available from http://sunxacml.sourceforge.net/.

[3] PERMIS PDP. Available from http://sec.cs.kent.ac.uk/permis

[4] TU-Eindhovens PDP. Available from http://w3.tue.nl/en/services/dpo/education_and_training/inleiding/pdp/

language, the PERMIS PDP uses its own XML based language whilst TU-Eindhoven's PDP uses SWI-Prolog. The next step is to write a full Master PDP so that all these PDPs can be called together in parallel and their decisions resolved into one final decision using either a configured or dynamically pushed conflict resolution policy.

8 Discussion, Conclusions and Future Plans

Our authorization infrastructure does not obviate the need for trust. Our infrastructure still requires trust between the various parties. It is not a digital rights management (DRM) system that assumes the receiving party is untrustworthy and wants to steal any received information from the sender. On the contrary, our infrastructure assumes that the various parties do trust each other to the extent that they want an automated infrastructure that can easily enforce each other's policies reliably and automatically, and if it cannot, will inform the other party of the fact. Consequently data subjects must trust the organizations that they submit their PII to, so that when an organization says it will enforce a subject's sticky policy, the subject can trust that it has every intention of doing so. Our system provides organizations with an application independent authorization infrastructure that makes it easy for them to enforce a subject's privacy policy without having to write a significant amount of new code themselves. Furthermore the user has the potential for more complete control over his/her privacy than now, in that the infrastructure allows the user to specify a complete privacy policy including a set of obligations which can notify the user when his/her data is accessed or transferred between organizations e.g. by using an *after* obligation when giving permission for the transfer of her PII to go ahead or a *before* obligation before giving permission for the PII to be read. However we expect the user interfaces for such full privacy policy creation to be too complex for most users to handle, and consequently organizations are more likely to provide their users with a policy template and a limited subset of options and boxes to tick, making the user's task much easier. This also reduces the burden on the organization, since it won't be sent user privacy policies that it cannot handle. The benefit of our infrastructure is that it does not constrain organizations in setting their privacy policy templates, as the infrastructure will enforce whatever combinations they choose.

Organizations must also trust each other to honor the sticky policies that are passed to them when they transfer data between themselves. An untrustworthy organization can always discard any sticky policies it receives and never need access the authorization infrastructure to ask for permission to receive the data, but we assume that legally binding contracts between the organizations will require them to support any sticky policies that are transferred between them. Our authorization infrastructure makes it much easier for them to do this.

Our final step is to implement the complete Master PDP and PDP/CV factory and then to perform user trials with two application demonstrators, one for the privacy protection and access to electronic medical records, the other for e-portfolios. Both of these applications require access to distributed personal information that is stored in a variety of repositories at different locations, and so a distributed sticky policy enforcement infrastructure is needed.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216287 (TAS³ - Trusted Architecture for Securely SharedServices)[5].

References

[1] BBC news on 18 June 2001, http://news.bbc.co.uk/1/hi/uk/1395109.stm
[2] CIFAS, http://www.cifas.org.uk/default.asp?edit_id=1014-57
[3] Msnbc report on 16 Jan 2008, http://www.msnbc.msn.com/id/22685515/
[4] Voice of America news report on 29th April 2008 , http://www1.voanews.com/english/news/science-technology/a-13-2008-04-29-voa44.html
[5] BBC news on 22 July 2009, http://news.bbc.co.uk/1/hi/business/8162787.stm
[6] BBC news on 24 August 2010, http://www.bbc.co.uk/news/business-11070217
[7] Zhu, Y., Keoh, S., Sloman, M., Lupu, E., Dulay, N.,Pryce, N.:A Policy System to Support Adaptability and Security on Body Sensors. In 5th International Summer School and Symposium on Medical Devices and Biosensors, pp.97—100. Hong Kong (2008)
[8] Wu, J., Leangsuksun, C. B., Rampure, V., Ong, H.: Policy-based Access Control Framework for Grid Computing. In: Proceedings of the sixth IEEE International Symposium on Cluster Computing and the Grid pp. 391-394. CCGRID, (2006)
[9] OASIS XACML 2.0. eXtensible Access Control Markup Language (XACML) Version 2.0, Oct, 2005. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20.
[10] OASIS XACML 3.0. eXtensible Access Control Markup Language (XACML) Version 3.0, 16 April, 2009. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html
[11] Chadwick, D., Zhao, G., Otenko, S., Laborde, R., Su, L. and Nguyen, T. A.: PERMIS: a modular authorization infrastructure. In: Concurrency And Computation: Practice And Experience, vol 20, issue 11, pp 1341-1357. (2008)
[12] W3C: The Platform for Privacy Preferences 1.0 (P3P 1.0). Technical Report. 2002.
[13] Blaze, M., Feigenbaum, J., Ioannidis, J.: The KeyNote Trust-Management System Version 2. RFC 2704 (1999)
[14] Chadwick, D. W., Otenko, S. and Nguyen, T.A.: Adding Support to XACML for Multi-Domain User to User Dynamic Delegation of Authority. J. International Journal of Information Security. 8, 137--152 (2009)
[15] Chadwick, D. W., Su, L., Laborde, R.: Coordinating Access Control in Grid Services. J. Concurrency and Computation: Practice and Experience. 20, 1071--1094 (2008)

[16] David W Chadwick, Stijn F. Lievens. "Enforcing “Sticky” Security Policies throughout a Distributed Application". MidSec 2008. December 1-5, 2008, Leuven, Belgium

[17] Karjoth, G., Schunter, M., Waidner, M.: Privacy-enabled services for enterprises. In: 13th International Workshop on Database and Expert Systems Applications, pp. 483-- 487. IEEE Computer Society, Washington. DC (2002)

[18] Karjoth, G., Schunter, M., Waidner, M.: Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In: 2nd Workshop on Privacy Enhancing Technologies , San Francisco (2002).

[19] Karjoth, G., Schunter, M.: A Privacy Policy Model for Enterprises.In: 15th IEEE Computer Foundations Workshop (2002).

[20] Nelson, R., Schunter, M., McCullough, M. R., Bliss, J. S.:Trust on Demand — Enabling Privacy, Security, Transparency, and Accountability in Distributed Systems. In: 33rd Research Conference on Communication, Information and Internet Policy (TPRC). Arlington VA, USA (2005).

[21] Schunter, M. and Berghe, C. V.:Privacy Injector — Automated Privacy Enforcement through Aspects. In: 6th Workshop on Privacy Enhancing Technologies, Robinson College, Cambridge, United Kingdom (2006), to be published as Lecture Notes in Computer Science, Springer Verlag, 2006.

[22] Mont, M. C.: Dealing with Privacy Obligations: Important Aspects and Technical Approaches.In: International conference on trust and privacy in digital business No1, Zaragoza (2004).

[23] Mont, M. C., Pearson, S., Bramhall, P.: Towards Accountable Management of Identity and Privacy: Sticky Policy and Privacy. Technical report,Trusted System Laboratory, HP Laboratories, Bristol, HPL-2003-49, (2003)

[24] Ni, Q., Trombetta, A., Bertino, E., Lobo, J.: Privacy aware role based access control. In: SACMAT’07, Sophia Antipolis, France (2007).

[25] Ni, Q., Bertino, E., Lobo, J.: An Obligation Model Bridging Access Control Policies and Privacy Policies. In: SACMAT'08, , Estes Park, Colorado, USA (2008)

[26] Mont, M. C.: Dealing with Privacy Obligations: Important Aspects and Technical Approaches. In: International conference on trust and privacy in digital business No1, Zaragoza (2004)

[27] Mont, M. C., Beato, F., On Parametric Obligation Policies:Enabling Privacy-aware Information Lifecycle Management in Enterprises. In: Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (2007)

[28] Ferreira, A., Chadwick, D., Farinha, P., Correia, R., Zhao, G., Chilro, R., Antunes, L.:How to securely break into RBAC: the BTG-RBAC model.In: Annual Computer Security Applications Conference, pp23-3,Honolulu, Hawaii, (2009)