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. Organisations are facing huge pressure to assure their users about the privacy protection of their personal data. Organisations may need to consult 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. The current paper is an enhanced version of [1] and it takes the research one step further.

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 records, health data, credit card information and so on are collected by different organisations in order to provide consumers with services. An example of such a service is an online job agency where people post their CVs in order to hunt for jobs worldwide. Once released, users lose control of their personal data. But personal data like CVs which contain sensitive personal information may invite not only job offers but also identity theft. Losing PII has serious consequences ranging from significant financial loss to becoming a suspect in a crime which was committed with a stolen ID. Innocent people have been arrested due to a crime committed by an identity thief [2]. In the UK, the number of ID thefts is an alarming 19.86% higher in the first quarter of 2010 compared with the same period in 2009 [3]. About 27,000 victims were recorded by CIFAS members during the first 3 months of 2010 [3]. As a consequence concerns for the privacy of electronic private data are rising day by day [4, 5]. Hence the necessity for more technical controls 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 help to protect personal data from being misused as well as enforce privacy laws so that personal data loss from reputable organisations such as HSBC bank [6] or Zurich Insurance [7] may be avoided in future.

Policy based systems are now well established [8, 9]. 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 [10], XACMLv3 [11], PERMIS [12], P3P [13], Keynote [14] etc. and even more PDP implementations. For example, P3P was 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 access control and privacy requirement using a single policy language or PDP. Therefore we need an infrastructure that can support multiple PDPs and multiple policy languages.

Private data should be protected by the policy of its owner. The sticky policy paradigm [15] ensures that private data is stuck with its policy not only within the initial system but also when transferred between systems.

Obligations are actions that must be performed when a certain event occurs. When the event is reading PII, then an obligation may require the PII subject to be notified. Some obligations may need to be performed along with the enforcement of an authorization decision, others before or after the enforcement [16]. Privacy protecting systems thus need an obligations service, ideally with a standard interface so that it can be called from multiple places in an application.

In this paper we propose an advanced multi-policy authorization infrastructure that will provide privacy protection of personal data using multiple PDPs, sticky policies and obligations. 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 whilst Section 5 describes the conflict resolution policy. Some use case scenarios are provided in Section 6. Details of our implementation are provided in Section 7. 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. [18]. They also did not provide a way to actually enforce the obligations which our system does.

HP researchers [22, 23] have also been working on providing privacy to PII by enforcing obligations. They have provided a way of transmitting encrypted confidential data with obligations to other parties by obfuscation of the data [22]. Nevertheless, the work only describes 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.

It has been claimed [26, 27] that the privacy policy defined by the owner of personal 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 a 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 and data controller together.

The Primelife Policy Language [28], [29] is an extension of XACML v3 which offers access control and usage control under the same structure. PPL has a new obligation handling mechanism which integrates a set of acceptable values for each obligation parameter and thus it solves the problem of "overdoing" obligations [30]. Their work mainly considered two groups of authors of privacy policy, the Data controller and Data Subject [29] where we considered Law and Issuer as well. An automated matching is performed between the Data controller and Data subject's policy so that any mismatching elements do not appear in the final sticky policy [29]. In contrast, our system allows all the policies from all types of authors, and we have a sophisticated, automated, dynamic conflict resolution policy that resolves the conflicts of the decisions returned by the different policies. Moreover, we have considered that these policies can be written in different languages which lead to the need for multiple policy language support.

3 The Authorisation System

Suppose that a health service provider wants to protect the privacy of personal health data (i.e. the Dr's note, history of treatment of the patient, diagnostic test report and so on) through their authorisation system. To provide privacy the authorisation system will need to include privacy rules from different authors such as the law, issuer (i.e. Dr) and data subject (the person). Suppose the data subject wants to share a part of his personal data with a health insurance company to recover the cost of treatment. When the data is shared with the insurance company it is expected that the policies related to the data will also be passed with the data and the receiving system will be capable of handling policies from different health service providers, which may be written in different policy languages. Also it is expected that the authorisation system will be able to enforce obligations such as sending email or keeping a secure log of accesses.

Consequently the authorisation system will need to provide the following features:

a. Enforcing multiple policies from multiple authors
b. Handling sticky policies
c. Distributed policy enforcement
d. Support for multiple policy languages
e. Obligation enforcement.

Our proposed system has the following capabilities.

a. **Enforcing multiple policies from multiple authors:** In a traditional access control system only the organisation's authority can set the access control policy which makes the system unsuitable for privacy protection. Our system will accept policies from different authorities and will resolve conflicts between decisions returned by different policies according to a sophisticated, automated, dynamic conflict resolution policy.
b. **Handling sticky policies:** Our system will accept policies stuck to PII, will store the policies, enforce them every time the PII is accessed, and will return them when the PII is transferred to a remote site.
c. **Distributed policy enforcement:** Our system ensures that sticky privacy policy are enforced within the current system and also in the receiving system, by a binding legal agreement between the two parties, such that if an organisation accepts data with a sticky policy, it confirms that its system will only store the data if it can satisfy the obligation to start one or more PDPs that support the received policies.
d. **Support for multiple policy languages:** Our system supports an arbitrary number of PDPs that each utilise a different policy language.
e. **Obligation enforcement:** Our system supports the enforcement of arbitrary obligations either before, after or whilst the access decision is enforced. The system is extensible and new obligations can easily be incorporated.

In order to satisfy the various requirements presented above we introduce several new components into the privacy preserving advanced authorization infrastructure. These are explained more fully in [1]. In this section a short description is given only.

Firstly we introduce an **application independent policy enforcement point,** the AIPEP. The AIPEP is so called because it enforces the application independent obligations and coordinates all the other components of the application independent authorization infrastructure. When the AIPEP receives 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 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 users' 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 groups, along with a Delegation Policy for each AA. We recognize that in distributed systems the same credential/attribute may be known by different names e.g. PhD, D Phil, Dr.-Ing etc. For this reason we introduce an **Ontology Mapping Server** (see below) which knows the semantic relationships between attribute names. If two names are semantically related then the CVS can determine if an unknown attribute is valid or not. Once the CVS has finished validating the subject's credentials, these are returned to the AIPEP as standard XACML formatted attributes (step 5), ready to be passed to the Master PDP.

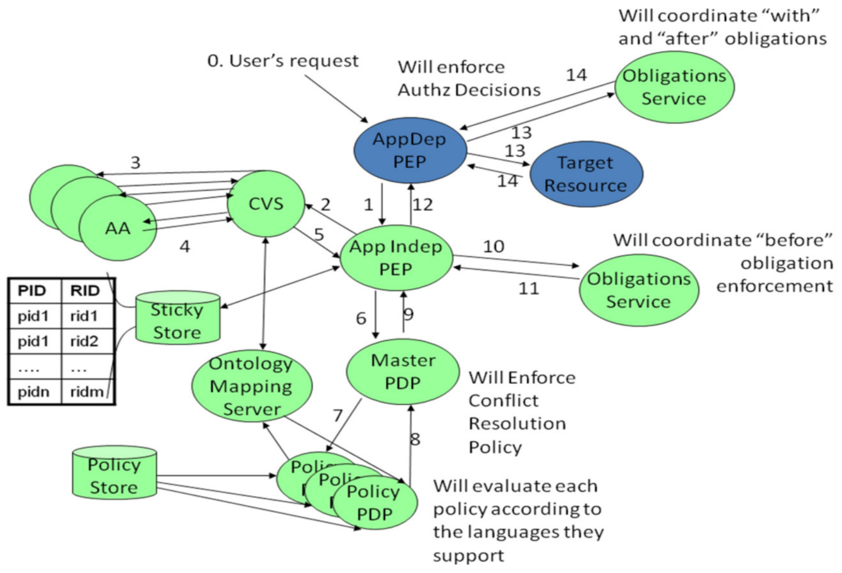
Will coordinate "with" and "after" obligations
0. User's request
Will enforce Authz Decisions
14
Obligations Service
AppDep PEP
13
13
Target Resource
14
3
CVS
2
1
12
AA
5
4
App Indep PEP
10
Will coordinate "before" obligation enforcement
Obligations Service
11
PID
RID
pid1
rid1
pid1
rid2
....
...
pidn
ridm
Sticky Store
6
9
Ontology Mapping Server
Master PDP
Will Enforce Conflict Resolution Policy
7
8
Policy Store
Policy PDP
Will evaluate each policy according to the languages they support

Fig. 1. The privacy preserving advanced authorization system

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 [31]. 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, and the globally unique Policy ID (PID). Each policy must have a globally unique id so that it can be uniquely referenced in the distributed system, primarily for performance reasons, so that when a sticky policy is moved from system to system, the receiver can determine if it needs to analyse the 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. 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.

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 resource has a locally unique resource ID (RID) which is mapped to Policy IDs. The RID is locally unique and will vary from system to system, as resources are transferred. We do not have a requirement to pass the RID from system to system so each system can compute its own.

Obligations may need to be enforced *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 [16]. 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 to 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 semantic relationship between two different terms. Multiple terms can name the same concept. The ontology concepts are held as a lattice, and the server will say if one term (equivalent concept) dominates the other in the lattice or if there is no domination relationship between them. The CVS calls this service to determine the relationship between an attribute name in an SOA's policy and the attribute name in an issued credential. The Master PDP calls this server to determine the relationship between the different terms in the policy and the request context.

4 Sticky Policy Contents

A sticky policy comprises the following elements:

- 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.
- 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. 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 our 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).

5 Conflict Resolution Policy

Our system includes 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 have introduced 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 time of creation so that the latest CRR always comes first in the ordered list.

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) [32] 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 or access over-ride and is particularly important in medical applications.

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 the resourceType=PII and the requestor=data subject) then the DCR=FirstApplicable and the (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 most specific policy will get preference. We define a policy to be the most specific if it is assigned to the most specific resource, as identified by its RID. We use the containment model in which the resource with the longest pathname is the most specific resource, for example C:/MyDocument/MyFile is more specific than C:/MyDocument. All the resources in the system have their RIDs formatted in the form of the URL hierarchy e.g. Kent.ac.uk/issrg2/C:/MyFiles. Each policy applicable to a resource is linked by its PID to that RID. In this containment model, polies applied to a less specific resource will also be applied to the resource contained in that; but policies applied to the more specific resource will not be applicable for the containing resource. For example, policies applied to the kent.ac.uk/issrg2/C: will be applied to kent.ac.uk/issrg2/C:/MyFiles but policies applicable to kent.ac.uk/issrg2/C:/MyFiles will not be applied to kent.ac.uk/issrg2/C:. If multiple most specific policies exist then all the most specific policies will be evaluated and the Deny result will get precedence; in other words DenyOverrides will be applied on the Most specific set of policies.

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 set of obligations.

For MajorityWins all the PDPs will be called and the final decision (Grant/Deny/BTG) will depend on the returned decision of the majority number of PDPs. If the same numbers of PDPs return Grant and Deny and there is at least one BTG, then BTG will be the final answer, otherwise Deny. If none of the PDPs return Grant, Deny or BTG 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 receive treatment from the X-Health Centre and for this he has to be registered. During the registration process he is 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. Other Drs at this health centre, indicating that the patient will accept a one to many relationship with the health centre staff (as opposed to a one to one relationship with a specific Dr.).
2. Other registered Drs/Consultants of other Organisations and a place where the name of the doctor and organisation can be written. If this box is ticked and no Dr's name and organisation are specified then the consent will be for any Dr in general.
3. Health Insurance Companies (with a place for specifying the name(s) of the company(ies) or can say all)
4. Research organisation/ researcher. (A note will say that all medical data used for research purposes will be anonymised prior to release.)
5. Other organisations for promotional offers. These are for example organisations offering samples and promotions for newborn babies and their parents. In this case not all of the medical record will be available to the interested companies. What portion of medical data will be available is determined by the organisation's policy.
6. Other person (for specifying the name of someone such as next of kin.)

Mr K has a policy with the Health Insurance Company HIC1 to cover his treatment costs. So he puts a tick in box 3 only and mentions HIC1 there and finishes his registration with X-Health Centre.

It is important to note that HIC1 will not have access to the complete medical records of Mr K. The policy of X-Health Centre determines what portion of the medical data can be made available to HIC1.

Mr K undergoes some treatment and HIC1 submits a request to X-Health Centre for (a portion of) the medical record of Mr K. The Master PDP of X-Health Centre's authorisation system consults the CRRs of Law, issuer, data subject and holder sequentially. The law CRRs say if resourceType=MedicalData and requestor=data subject and resourceClassification is different from Drs notes then DCR=GrantOverrides; anything else for resourceType=MedicalData leads to a DCR of DenyOverrides. Therefore in the case of medical data the CRRs of the other entities will never be consulted. Since the requestor is not the data subject the DCR is DenyOverrides. The PDPs give the following results:

- The law PDP returns NotApplicable because it only has rules pertaining to cases where the requestor is either the data subject or the creator of the data.
- Depending on the actual medical data requested, the issuer (X-Health Centre) PDP returns either Grant or Deny.
- The data subject PDP returns Grant because Mr K has allowed this on his registration with the health centre.

The overall decision is therefore the same as that of the issuer's PDP. Assuming this is Grant, then the requested medical data is passed to HIC1 together with the sticky policies from the data subject, the law and the issuer.

After receiving the medical data and sticky policies the receiving application will make a call to the authorisation system of HIC1 to see whether it is permitted to store the data. The authorisation system will reply Grant and will start two new PDPs with the received policies of the data subject (Mr K) and the issuer (X-Health Centre) – assuming it can process them. If it cannot, it will return Deny. At HIC1's site a law PDP is already running containing the same legal policy as that sent and therefore it does not need to start a new law PDP. If HIC1 had been in a different jurisdiction to X-Health Centre, then it would have needed to run a new legal PDP for Mr. X's medical data. The policy to data mappings are duly recorded in the sticky store. HIC1's authorisation system subscribes for updates to these policies, so that if the patient or X-Health Centre should change their policies, the new ones will be notified to HIC1's authorisation system..

Mr K did not allow researchers to view his medical record. The researcher Mr R requests Mr K's medical record at HIC1's system and this request is denied for the following reasons:

- as in the previous case the chosen DCR will be DenyOverrides chosen by the law CRR.
- the law PDP will return NotApplicable (as before),
- the issuer's PDP returns Grant with a "with" obligation for anonymisation (because it allows the data to be used for research),
- the holder's (HIC1's) PDP returns NotApplicable because it allows data subjects to determine if researchers should have access to their data or not,
- Mr K's PDP returns Deny which makes the overall result a Deny..

Mr K now changes his preferences at the X-Health Centre and allows his data to be accessed by researchers. The X-Health Centre publishes this update and both its and HIC1's authorisation systems update the subject's policy with the new rule, which contains a "with" obligation to anonymise the data prior to release. If a researcher now asks for access to Mr K's data at HIC1's site, the DCR will be DenyOverrides chosen by the law CRR, the law PDP will return NotApplicable (as before), both the issuer's and subject's PDPs return Grant with the same "with" obligation to anonymise the data, and the holder's (HIC1's) PDP returns NotApplicable. The overall result is therefore Grant with an obligation. The obligation is passed to the Obligations Service of the PEP for it to enact simultaneously with data release. If this obligation cannot be enforced by the PEP, then it must deny the researcher's request.

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 Master PDP, a policy store and sticky 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 only a small overhead in most cases (between 0.3% and 50%,depending on the size of the policy and the actual request) to the performance of a stateless PDP that does not support BTG. 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 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 language, the PERMIS PDP uses its own XML based language whilst TU-Eindhoven's PDP uses SWI-Prolog. The next step is to integrate a secure publish/subscribe mechanism for policy updates and write a reasonably full set of

[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/

validation tests and use cases along with example policies so that all the PDPs can be called together and their decisions resolved into one final decision using a conflict resolution policy that obeys the law and the wishes of all the various actors. This is a complex task since the number of permutations is infinite.

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 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).

References

[1] Chadwick, D. W., and Fatema, K.: An advanced policy based authorisation infrastructure. In :Proceedings of the 5th ACM workshop on Digital identity management (DIM '09). ACM, New York, NY, USA, 2009.

[2] BBC news on 18 June 2001, http://news.bbc.co.uk/1/hi/uk/1395109.stm

[3] CIFAS, http://www.cifas.org.uk/default.asp?edit_id=1014-57

[4] Msnbc report on 16 Jan 2008, http://www.msnbc.msn.com/id/22685515/

[5] Voice of America news report on 29th April 2008 , http://www1.voanews.com/english/news/science-technology/a-13-2008-04-29-voa44.html

[6] BBC news on 22 July 2009, http://news.bbc.co.uk/1/hi/business/8162787.stm

[7] BBC news on 24 August 2010, http://www.bbc.co.uk/news/business-11070217

[8] 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).

[9] 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).

[10] 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.

[11] 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

[12] 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)

[13] W3C: The Platform for Privacy Preferences 1.0 (P3P 1.0). Technical Report. 2002.

[14] Blaze, M., Feigenbaum, J., Ioannidis, J.: The KeyNote Trust-Management System Version 2. RFC 2704 (1999).

[15] Chadwick, D. W., Lievens, S. F.: Enforcing "Sticky" Security Policies throughout a Distributed Application. MidSec 2008. Leuven, Belgium. December 1-5, 2008.

[16] Chadwick, D. W., Su, L., Laborde, R.: Coordinating Access Control in Grid Services. J. Concurrency and Computation: Practice and Experience. 20, 1071--1094 (2008).

[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] Ardagna, C. A., Bussard, L., Vimercati, S. D. C., Neven, G., Paraboschi, S., Pedrini, E., Preiss, F-S., Raggett, D., Samarati, P., Trabelsi, S., Verdicchio, M.: PrimeLife Policy Language, Project's position paper at W3C Workshop on Access Control Application Scenarios. November 2009.

[29] Trabelsi, S., Njeh, A., Bussard, L., Neven, G.: PPL Engine: A Symmetric Architecture for Privacy Policy Handling. Position paper at W3C Workshop on Privacy and data usage control. October 2010.

[30] Bussard, L, Neven, G., and Schallaböck, J.: Data Handling: Dependencies between Authorizations and Obligations. Position paper at W3C Workshop on Privacy and data usage control. October 2010.

[31] OASIS "SAML 2.0 profile of XACML, Version 2.0". OASIS committee specification 01, 10 August 2010.

[32] 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).