Obligations for Privacy and Confidentiality in Distributed Transactions

U.M.Mbanaso[1], G.S. Cooper[1], David Chadwick[2], Anne Anderson[3]

[1]Informatics Research Institute (IRIS), University of Salford, UK
[2]Computing Laboratory, University of Kent, UK
[3]Sun Microsystems Inc, Burlington MA USA

Abstract. Existing access control systems are typically unilateral in that the enterprise service provider assigns the access rights and makes the access control decisions, and there is no negotiation between the client and the service provider. As access management systems lean towards being user-centric, unilateral approaches can no longer adequately preserve the user's privacy, particularly where the communicating parties have no pre-existing trust relationships. Establishing sufficient trust is therefore essential before parties can exchange sensitive information. This paper describes a bilateral symmetric approach to access control which deals with privacy and confidentiality simultaneously in distributed transactions. We introduce the concept of Obligation of Trust (OoT) as a privacy assurance mechanism that is built upon the XACML standard. The OoT allows communicating parties to dynamically exchange their privacy requirements, which we term Notification of Obligations (NOB) as well as their committed obligations, which we term Signed Acceptance of Obligations (SAO). We describe some applicability of these concepts and show how they can be integrated into distributed access control systems for stricter privacy and confidentiality control.

1. Introduction

Trends in emerging access management systems raise an interesting paradox. On the one hand, service providers' applications require identity/attribute related information in order to validate a user's request. On the other hand, users may not wish to disclose their information or attributes to a remote Service Provider (SP) without determining in advance whether the service provider can be trusted to comply with their privacy preferences. Conventionally, privacy is often considered from the users' perspective, just as access control is considered from the SP's standpoint. That is, the user is concerned about the confidentiality of their personal identifying information (PII), and the resource provider is concerned about the confidentiality and integrity of the resource information. These assumptions have resulted in unilateral asymmetric approaches. Yet the SP may also have sensitive attributes such as membership certificates of consortia, or trust relationships with third parties (TTPs) or policies of various kinds that a resource user may demand to see before releasing their PII. This suggests a symmetrical approach may be more appropriate, and has led to the research topic called trust negotiation where each party's attributes are released incrementally

to the other, as trust is established between them [1]. In B2B transactions, both parties may require the dynamic exchange of service level agreements (SLA) or business level agreement (BLA) in order to assess the mutual benefits and associated risks. This may also require the establishment of trust and a guarantee of compliance to agreed business rules. One way to achieve this is for each party to issue to the other a proof of acceptance of the requirements contained in the SLA or BLA. Enabling the runtime exchange of these requires a bilateral symmetric approach to allow the communicating parties to indicate their willingness to accept constraints imposed by the other party, before the latter is prepared to reveal their sensitive information. There is therefore some overlap between user privacy requirements and business requirements.

To address confidentiality and privacy problems simultaneously and symmetrically, the parties in distributed transactions should have a standard means of declaring their privacy requirements and the respect they will give to the other party's privacy requirements before sharing their resources. All parties need to evaluate the risk of giving out their PII and determine the degree to which they are prepared to trust the other participating actors. They will need to identify any constraints and obligations they may wish to place on the others. Trust negotiation [1] has been proposed to address this dilemma, but as will be pointed out later it has its limitations. We therefore approach the subject of resources control in a slightly different manner. We propose a technical solution that derives its concepts from well established standards. We describe the concept of an Obligation of Trust (OoT) protocol, whereby two parties can exchange difficult-to-repudiate[1] digitally signed *obligating constraints* (or Notification of Obligations (NOB) which detail their requirements for sending their sensitive information to the other party), and *proof of acceptances* (or Signed Acceptance of Obligations (SAO), which acknowledge the conditions they have accepted for receiving the other party's sensitive information). The OoT protocol provides the negotiating mechanism for carrying obligating constraints and proof of acceptances between security domains. Being signed, they help the communicating parties to produce difficult-to-repudiate technical evidence in the event of disputes. The OoT protocol also provides a mechanism for dynamically exchanging other obligating documents such as service level agreements (SLAs), business level agreements (BLAs), contractual documents, etc. In effect, the OoT protocol merges technical solutions (mechanical exchange and matching, digital signature) with potential social/judicial solutions (non-repudiation, technical legal recourse). The rest of this paper is structured as follows. Section 2 describes related research. Section 3 presents the OoT protocol as well as how matching of obligation constraints and proof of acceptances is achieved. Section 4 describes the system architecture of a reference engine and its core subsystems, which we are currently constructing. In section 5, we provide an example use of the model and section 6 concludes the paper.

[1] We use the term "difficult-to-repudiate" rather than non-repudiation, since repudiation is a legal issue that has to be determined in a court of law. The technical constructs proposed in this paper should make it more difficult for an entity to repudiate their actions.

2. Related Research

The Platform for Privacy Preferences (P3P) [2] is one approach that attempts to address privacy in commercial service provider (SP) websites. Whilst it has provided some degree of privacy awareness, it has not particularly addressed privacy concerns in distributed access control systems. The fact that P3P is widely implemented by most websites and processed by compliant user-agents by comparing the P3P policy statement against an APPEL [3] statement that describes the user's privacy preferences is beneficial. By contrast, in distributed access control systems, SPs don't usually convey their privacy policy statements to the service users during access request. Even if a user in a distributed access control system retrieves the remote P3P policy, the policy may not necessarily meet the user's preference. Thus, the user may abort the service or continue without the choice for further negotiations. Also P3P doesn't support provider-side requirements; the SP may have some privacy constraints that require enforcement at the client's side. The main components of a P3P privacy statement include the *recipient* of the data, the *purpose* for which that data is requested, the *retention period* at the collector's store, and the *data category*. It can include other components such as *disputes* and *remedies*, as well as whether *disclosure to third parties* is allowed. Though P3P covers most of the basic principles of privacy [4], the fact that it has not satisfactorily resolved the requirements for bilateral privacy negotiation [5] limits its use in access control.

Shibboleth [6] from Internet2 provides a mechanism for federated access management based on the SAML security standard [7]. Shibboleth provide single sign on (SSO) and a mechanism for an IdP in one security domain to securely convey attributes about a web-browsing user to a SP in another security domain. In Shibboleth, privacy is addressed in two ways. Firstly, after the user authenticates to the IdP, the Shibboleth authentication service generates a one time handle to identify the user and transmits this to the SP. Secondly, the IdP uses Attribute Release Policies (ARP's) to decide whether to release specific attributes to the SP or not. This is fine as long as the remote site doesn't require any identifying attributes to complete the service. But this is unlikely to be the case in most transaction scenarios. Furthermore, the Shibboleth infrastructure doesn't provide any support for bilateral negotiation of service parameters. If the user doesn't provide the requested attributes, access to the services is unilaterally denied. Another significant privacy flaw is that the ARP is coarse and doesn't support most of the known privacy principles [4].

ID-WSF from the Liberty Alliance is an open standard for federated identity management that is built upon the extensibility of SAML security assertions [7]. It provides a framework for the discovery and communication of identity information among federated domains. When a client authenticates to an IdP, a SAML-based assertion handle (SSO) is generated and communicated to a relying party or SP with optional information which the relying party may use to call-back the user's IdP. The ID-WSF framework provides a flexible security model for a highly distributed set of IdPs.

Microsoft, IBM and VeriSign have been working on a set of specifications (called "WS-Security roadmap" or "WS-Identity Policy Framework") for their next generation platform of Web services. The WS-Policy suite of policies, which includes Security Policy, Reliable Messaging Policy, etc. are not designed primarily for

implementing access control. They are predominantly designed to enable Services to advertise what requirements (especially authorization requirements) a requesting party must satisfy in order to use the services. The idea is that a requesting party can consider what it is willing and able to accept, before sending attributes that can satisfy the requirements. However, WS-policies do not necessarily provide a means to enforce access control policies since typically they are not to be consumed by Policy Decision Points (PDPs).

One approach that addresses bilateral access control is the Automatic Trust Negotiation (ATN) technique [8, 9]. ATN introduces a trust negotiation layer for symmetrical interactions. Research efforts in this area have developed advanced ATN techniques to cover a variety of scenarios [10] [11] [12]. Recent initiatives in preserving privacy [13, 14] also favour the use of negotiation techniques for solving privacy problem. ATN is an access control technique that permits the gradual release of policies and credentials so that trust can be incrementally increased until the communicating parties are sufficiently satisfied of each others trustworthiness to send all their confidential information. However, ATN doesn't provides mechanisms whereby the relying party can convey proof of acceptance for obligating constraints - assurance that the attributes contained in the assertions will be used in accordance with the party's privacy preferences. Recent work in this area by Spantzel et al [15] introduces a framework that integrates ATN with Identity Management Systems (IdM). Based on their comparison of ATN and IdM systems, it shows that ATNs have not truly explored access security standards such as XACML, SAML, etc which may limit their practical implementation.

To the best of our knowledge, none of the above systems provides a mechanism for the remote enforcement of privacy obligations. So there is uncertainty that the receiving party will adhere to them. Further, the receiving party may not accept any liability if the sender's PII is compromised. Without privacy assurances there is the possibility that the receiving party may even misuse the sender's PII without any form of liability. Privacy negotiation will provide a mechanism that relies less on trusted external third parties and more on the communicating parties themselves. Privacy is governed by laws, legislation and principles requiring that privacy solutions should provide tenable difficult-to-repudiate technical evidence in the case of a privacy dispute. Consequently, there is a need to provide a mechanism for providing tamper-proof technical evidence that may be used in the event of disputes when parties do not conform to their commitments. One approach to achieve this is to provide a protocol to enable participating parties to exchange digitally signed commitments. We acknowledge that a technical "non-repudiable signature" on its own may not be sufficient evidence for a court of law since other factors also contribute to a digital signature being legally non-repudiable, such as: how much active participation the user had in deciding to sign, how free the user is to use the signed-for sensitive information, whether the software automatically generated the signature, and how complex the signed agreement is. However, these legal issues are not within the scope of the current paper. We consider the technical issues only that will help to provide difficult-to-repudiate evidence.

3. Obligation of Trust (OoT) Protocol

Obligation of Trust is a protocol that defines a standard mechanism enabling two or more communicating parties to exchange *obligating constraints* as well as *proof of acceptances*. The basic concept is built upon the assumption that a requesting party has no means of enforcing obligations placed on a remote party. In traditional access control systems, an obligation is an action that should be performed by a Policy Enforcement Point (PEP) in conjunction with the enforcement of an access control decision [13]. XACML [16] describes an *Obligation* element as a set of attribute assignments, with an attribute *FulFillOn* which signifies whether the consuming PEP must fulfill the *obligation* if the access control decision is "Permit" or "Deny". When a Policy Decision Point (PDP) evaluates a policy containing obligations, it returns the access control decision and set of obligations back to the PEP. However, in a distributed environment the SP's PEP is unlikely to be in the same security domain as the service requestor; therefore there is no guarantee that any obligations required by the requestor can either be incorporated into the policy used by the SP's PDP, or even if they can, be enforced by the SP's PEP. Given this, it makes sense to address the remote enforcement of obligations by allowing a SP to convey back to the requestor an acceptance or rejection of their obligating constraints. The OoT protocol addresses this interaction. We divide the OoT protocol into two steps: Notification of Obligation (NOB) (which may be signed or unsigned) and Signed Acceptance of Obligation (SAO) (which must be signed). The OoT protocol is symmetric. An initiating party sends a NOB outlining the obligating constraints it is placing on the other party and the commitments it is willing to make if the other party accepts its obligations. The other party, after evaluation, sends back either a signed acceptance (SAO) of the constraints it accepts and the commitments it requires, or initiates more service negotiations with its own NOB, or rejects the request and terminates the session. Because the NOB and SAO are constructed using standard XACML obligations elements, both communicating parties have a common language for expressing their requirements and commitments, and are able to feed these obligations directly into their PDPs for automatic decision making, and ultimate enforcement by their respective obligations services.

OoT Encoding Scheme

The Web Services Profile of XACML (WS-XACML) [17] describes a way for carrying XACML policies between communicating parties. WS-XACML specifies formats for four information types:

- an authorization token or credential for carrying an authorization decision across realms,
- a policy assertion type that is based on XACML elements which can embed WS-Policy or other XML constructs,
- ways to wrap P3P policy preferences and match them using XACML assertions, and
- XACML Attributes in SOAP Message Headers in such a way that they can be authenticated as having been issued by a trusted authority.

The WS-XACML Assertion Type is an abstract framework that describes an entity's Web Service's policy in the context of different policy domains, such as authorization or privacy domains. The name of the Assertion's element indicates the domain to which it applies, such as XCMLPrivacyAssertion for the privacy domain and XACMLAuthzAssertion for the authorization domain. The XACMLPrivacyAssertion deals with privacy specific Assertions which can carry Requirements i.e. what the asserter requires of the other party, and Capabilities i.e. what the asserter is willing and able to do for the other party if its Requirements are satisfied. The inner box in Figure 1 depicts the WS-XACML model which defines an *XACMLAssertionAbstractType.* This allows constraints on a policy vocabulary to be expressed as XACML *Apply* functions. The *XACMLAssertion* contains two sets of constraints as shown in figure 1. The first set, called *Requirements*, describes the information or behavior that the policy owner requires from the other party. The second set, called *Capabilities*, describes the information or behavior that the policy owner is willing and able to provide to the other party. One instance of this type is the *XACMLPrivacyAssertion* whose *Capabilities* element describes the *Obligations* that are being accepted and the information that will be provided. The *Requirements* element specifies the *Obligations* that the sender requires of the other party in order to proceed.

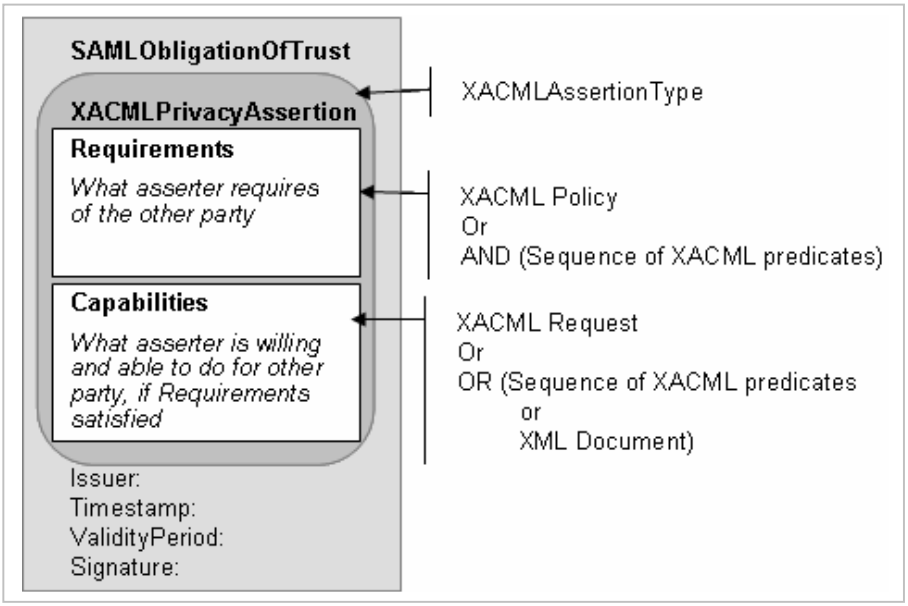

Fig. 1. SAML Obligation Of Trust Model

Using the built-in extensibility mechanism of WS-XACML and SAML Assertions, we can conveniently encode the components of the OoT protocol as extensions of standard elements. The NOB can be expressed as an instance of a *XACMLPrivacyAssertion* in which the desired obligating constraints are placed in the

```
<Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:xpath-expression-subset">
    <AttributeSelector
    RequestContextPath="//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/*"
    DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression" />
    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:xpath-expression-bag">
        <AttributeValue DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
        expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/current</At
        tributeValue
        <AttributeValueDataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
        expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/admin</Att
        ributeValue>
        <AttributeValueDataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
        expression">//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours</Att
        ributeValue>
    </Apply>
</Apply>
```

Fig. 2. Example of WS-XACML constraint on P3P PURPOSE.

Requirements section of the Assertion, and any obligations that the sender is willing and able to fulfill in the *Capabilities* section. The SAO can be expressed as an instance of a *XACMLPrivacyAssertion* in which the *Requirements* section specifies the sender's understanding of what the recipient has committed to do and the *Capabilities* section specifies the obligations that the sender has committed to undertake. By signing the SOA the signer is stating in a difficult-to-repudiate form their commitment to fulfill the *Obligations* contained in the *Capabilities* element, so long as their *Requirements* are satisfied. Figure 1 shows the extensions of WS-XACML and SAML that map into our Obligation of Trust model. The OoT schema is available at [18], but basically it defines a new SAML protocol request type (the Obligation of Trust Query Type) and a new SAML statement type (the Obligation of Trust Statement Type).

In the privacy domain, these elements can be used to describe either the acceptable (Requirements) or supported (Capabilities) P3P policy contents. For example, if a recipient will only use the sender's sensitive information for the "current" transaction and "admin" purposes, and the information is only for the designated recipient, this can be sent as a P3P policy STATEMENT of PURPOSE expressed as a WS-XACML constraint as shown in figure 2.

OoT Protocol Scheme

Figure 3 is a simplified sketch of the OoT protocol in operation, and shows how two parties may exchange signed components of the OoT. Party A wishes to access item X from party B, but it is assumed that party A knows nothing about the privacy or access control requirements for item X. Similarly, Party B knows nothing about the privacy requirements of Party A's attributes. Party A sends a request for item X and Party B responds with a NOB containing its *Requirements* and *Capabilities*. Figure 4

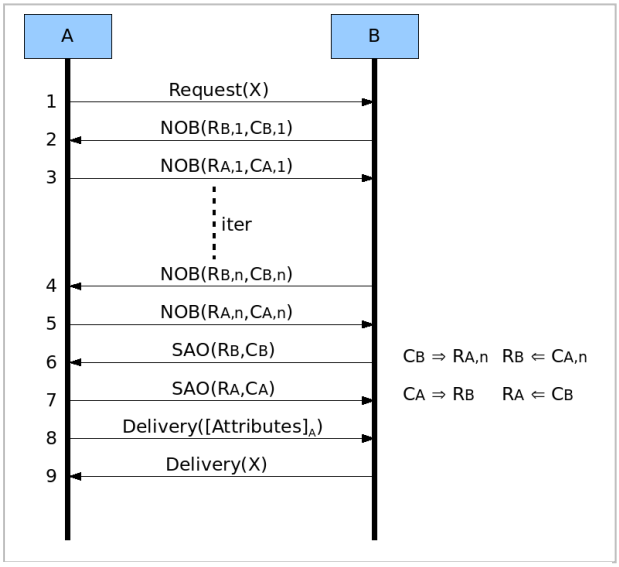

Fig. 3. The OoT Protocol Sketch

shows an outline of an algorithm for the decision making when a party receives a NOB. Party A checks whether it can satisfy Party B's *Requirements*, and whether party B's *Capabilitie*s can satisfy its own (party A's) *Requirements*. If Party B's *Capabilities* are acceptable and sufficient for Party A, and A can fully meet B's requirements, then A can send an SAO to B stating its pick of the offered capabilities and its own capabilities to meet party B's requirements. If B's capabilities are acceptable but not sufficient, or A has additional requirements, A may send a counter NOB to B containing its additional or alternative *Requirements.* A's *Requirements* will determine the subset of B's *Capabilities* that it requires, and A may supplement them with additional ones of its own. A's *Capabilities* will include the subset of B's *Requirements* that it can provide, along with any additional ones it may be willing to provide. If Party B's *Capabilities* are insufficient for Party A, then A will either terminate the session or return a NOB with *Requirements* that supercede B's stated *Capabilities*. If A cannot meet all the stated requirements of B, then A may decide to terminate the session or add a reduced set of Capabilities to the NOB.

Party B evaluates party A's NOB and if satisfied with A's *Capabilities* and *Requirements* it returns a signed SAO stating in its *Capabilities* that it can fullfil all of party A's *Requirements,* and in its *Requirements* which of Party A's *Capabilities* it has chosen. If B is satisfied with A's *Capabilities* but not with A's *Requirements*, B may either send another NOB to A showing less *Capabilities* than A requires (along with its own *Requirements*), or terminate the session. If B is not satisfied with the *Capabilities* of A's NOB, it will either terminate the session or return a NOB with increased *Requirements*. If Party A receives another NOB, and this is satisfactory, it returns a signed SAO, otherwise it behaves as last time around. If Party A receives party B's SAO, and if satisfied with it, it returns its own signed SAO. Thus the parties

- Set flag initially to "SAO"
- Evaluate received requirements to determine whether I can meet them with my capabilities
  - If so, construct offered Capabilities to match received requirements
  - If not, either
    - terminate or
      - determine* whether additional capabilities should be offered to match, and/or
      - construct capabilities to match a subset of the received requirements, plus additional alternative capabilities to be offered, and set flag to "NOB"
- Analyse capabilities to be offered by me (as determined above) and construct a revised list of (my) requirements.
- Analyse sets of capabilities received and compare with my list(s) of requirements (as determined above).
  - If all my requirements are met from one set of offered capabilities, keep the above-defined requirements.
  - If all my requirements are met from merged sets of offered capabilities, construct Requirements from these, set flag to "NOB"
  - If my requirements are not met, either
    - terminate or
    - determine* whether requirements can be relaxed due to alternative capabilities being offered and modify requirements accordingly and set flag to "NOB"
- If SAO flagged, send SAO, else send NOB.

("determine" could include the possibility to ask a human operator.)*

Fig. 4. Outline Algorithm for handling a NOB

continue to exchange NOBs until either one party terminates the session (negotiated agreement not possible) or returns a signed SAO. Once a signed SAO has been delivered the recipient must either accept this by returning its own signed SAO or terminate the session. It is not allowed to return a NOB in response to a signed SAO, since this is in effect rejecting what one had previously offered in a prior protocol exchange. Once the negotiation is complete, and each party is in possession of the signed SAO of the other party, then Party A delivers the attribute values defined in Requirement B and Party B delivers item X to A.

As indicated above, in some transactions it will be the case that either a user's configured capabilities are insufficient to match an SP's requirements, or a user's requirements are too great for an SP's capabilities. In this case the software might indicate to the user that the SP's (or user's) requirements are not covered by any of the user's (or SP's) sets of capabilities. The user should be able to view the NOB request and possibly extend their capabilities or reduce their requirements. As an example, suppose a user has configured his requirement's policy so that recipients are not to reveal the user's PII to 3rd parties, but a Service X offers very generous compensation to Service C's users who are willing to sign up for X's new services. In

this case, Service C could send the user a NOB containing a *Requirement* to provide permission for Service C to release PII to Service X, in exchange for compensation. The user's agent does not have a *Capability* to match this *Requirement*, so the user's client software could display Service C's *Requirement* for the granting of permission to forward the PII to Service X, along with Service C's *Capability* to offer compensation to the user. If the user dynamically chooses to accept this contract, a new *Capability* is added to the user's set of XACMLPrivacyAssertions, for this and

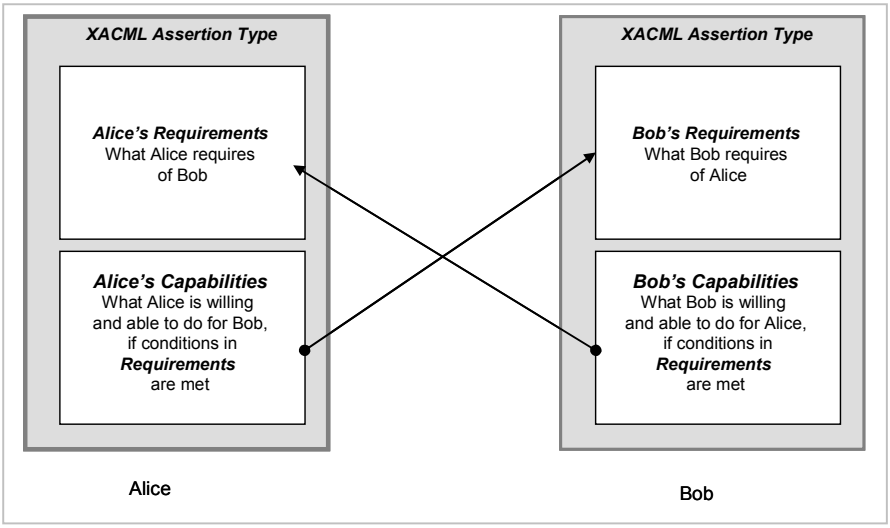

Fig. 5. Matching of Two WS-XACML Assertion Type

future use, and a signed SAO is sent to Service C.

Matching and Evaluation

Requirements are logically connected by AND: the policy owner requires the other party to satisfy **all** of the constraints listed in the *Requirements* section. *Capabilities* on the other hand are logically connected by a non-exclusive OR: the policy owner is willing and able to provide any subset of the capabilities described by these constraints. Figure 5 illustrates the matching of the two WS-XACML Assertions. Two *XACMLAssertions* match if, for each assertion, all constraint in the *Requirements* section are satisfied by (at least) one of the statements in the *Capabilities* section of the other assertion. WS-XACML specifies efficient generic algorithms for determining that one constraint "satisfies" another. We can use this mechanism to evaluate an XACML-P3P policy against an XACML privacy profile (or any policy expressed in XML), provided we have matching semantics between them. Once the matching is done, the next step is to extract the capability that matches the recipient's requirements, produce the SOA and generate the signatures.

4. Example of WS-XACML Aware Applications

The OoT protocol provides a platform which permits two or more communicating parties to negotiate obligating constraints in a tamper proof manner. Privacy Negotiation is one such good example of using the OoT principles.

As an example, an Internet-based ticket service (ITS) provides online ticketing services to both consumers and partners through automated Web services. The ITS can provide special price offers to certain categories of clients in particular seasons. The ITS requires prospective clients to provide or show proof of possession of certain properties and then to make firm commitments that they will not disclose its price list to third parties (i.e. competitors) before it can decide whether they qualify for special offers. On the other hand, the clients may not wish to give out their sensitive attributes without receiving proof from the ITS that it will not disclose them. The ITS therefore needs to assure the clients that their attributes will be held according to their privacy preferences. Figure 6 depicts the ITS's internal XACMLPrivacyAssertion and figure 7 is the customer's internal XACMLPrivacyAssertion. Looking at the

XACMLPrivacyAssertion (ITS)
Requirements
Client Name
IATA membership certificate
Certified Quarterly Sales > £12,000.00
Price List not given to 3rd parties
Capabilities
PURPOSE: PII used internally for this transaction
RETENTION: PII kept only until transaction is completed
RECIPIENT: PII not given to any 3rd party

Fig. 6. ITS's Internal XACMLPrivacyAssertion

XACMLPrivacyAssertion (customer)
Requirements
RETENTION: PII kept only until transaction is completed
RECIPIENT: PII not given to any 3rd party
Capabilities
Name
IATA membership certificate
Certificate of Incorporation
Certified Quarterly Sales > £12,000.00
Price List not given to 3rd parties

Fig. 7. Customer's Internal XACMLPrivacyAssertion

assertions, the customer's *Requirements* are really "Obligations" to be fulfilled by the ITS. Similarly, the ITS's *Capabilities* are really "Obligations" that the ITS is able and willing to meet. The OoT provides the mechanism to assure each participant of the other's commitment to respecting their security preferences. Each party can save the

digitally signed XACMLPrivacyAssertion with the complete *Capabilities* as difficult-to-repudiate evidence in the case of disputes.

6. Conclusion

This paper describes one concrete approach to enhancing privacy assurance, by permitting the bilateral exchange of privacy *Requirements* and the *Capabilities* to satisfy them. The OoT mechanism merges technical solutions with possible social/judicial solutions for security assurance in distributed open systems. This mechanism demonstrates a secure way of using P3P policies in WS-XACML which provides a framework for the dynamic exchange of requirements and capabilities, meaning that this framework can support the P3P platform with minimal effort. Our solution demonstrates significant improvement in the provision of privacy in distributed transactions where technically "difficult-to-repudiate" services are vital. Again, the benefit of this framework is that the same security engine can apply to the four types of information described in WS-XACML, meaning that privacy and confidentiality can be achieved simultaneously for both service providers and consumers. This approach is currently being implemented

An additional benefit of this approach over traditional ATN is that it has the potential to reduce the number of interactions between parties and therefore the effects of network latency since both requirements and capabilities can be transmitted in a single payload rather that separately. A mechanism that assures each party that their information will be used in accordance with their wishes will increase the level of trust and confidence between the communicating parties and may even reduce the liabilities of regulated organizations.

The OoT protocol has a couple of limitations. Firstly it assumes that the other party exists as a physical entity that can be sued if violations occur. This requires either a robust PKI system to exist or some other mechanism to establish whether the subject of a certificate is a legal entity, and will put meaningful identifying information in the issued certificate. Secondly, it is open to probing attacks. A malicious party can probe another party by providing bogus capabilities in order to gather the other party's requirements and capabilities and then terminate the connection before any actual data is transferred. In [19], we described how XACML can be used to address the probing attack by a trust negotiation involving the gradual and incremental exchange of information. This requires that the XACML policy is expressed in such a way that the level of trust established can determine what other information (policy/attributes) is released at any phase. The order and sequence are controlled by the crafting of policy rule expressions. Furthermore, we have not dealt with refinements for multiple assertions and multiple set of Capabilities. These are the subject of further work.

Work is currently being carried out on a reference implementation of the proposed approach, and the testing and evaluation of this will be published in due course.

7. References

1. Bertino, E., Ferrari, E., Squicciarini, A.: Trust Negotiations: Concepts, Systems and Languages. IEEE Computer, pp. 27-34, 2004.
2. W3C: The Platform for Privacy Preferences 1.0 (P3P 1.0). Technical Report. 2002.
3. Langheinrich, E.Z.M,: A P3P Preference Exchange Language 1.0 (APPEL1.0). W3C. 5 April 2002.
4. OECD: Fair Information Practices In The Electronic Marketplace A Report To Congress. http://www.ftc.gov/reports/privacy2000/privacy2000.pdf May 2000.
5. W3C: Platform for Privacy Preferences (P3P). 2004.
6. Cantor, S.: Shibboleth Architecture. Internet2 Middleware. http://shibboleth.internet2.edu/shibboleth-documents.html 2005.
7. Cantor, S., Kemp, J., Philpott, R., Maler, E.: Security Assertion Markup Language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf. March 2005.
8. Seamons, K. E., Ryutov, T., Zhou, L., Neuman, C., Leithead, T.: Adaptive Trust Negotiation and Access Control. 10th ACM Symposium on Access Control Models and Technologies. Stockholm, Sweden, 2005.
9. Winsborough, W. H., Li, N.: Towards Practical Automated Trust Negotiation. Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002), 2002.
10. Seamons, K. E., Winslett, M., Yu, T., Yu, L., Jarvis, R.: Protecting Privacy during On-line Trust Negotiation. 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA, 2002.
11. Winsborough, W.H., Seamons K.E., Jones, V.E.: Negotiating Disclosure of Sensitive Credentials. 2nd Conference on Security in Communication Networks., Amlfi, Italy, 1999.
12. Bertino, E.F.E., Squicciarini, A.: TNL: An XML-based Language for Trust Negotiations. IEEE 4th International Workshop on policies for Distributed Systems and Networks, Lake Como Italy, 2003.
13. Pau, L.-F.: Privacy Negotiation and Implications on Implementations. W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement, 2006.
14. Preibusch, S.: Privacy Negotiations with P3P. W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement, 2006.
15. Spantzel, A.B., Squicciarini, A.C., Bertino, E.: Trust Negotiation in Identity Management. IEEE Security & Privacy, pp. 55 - 63, 2007.
16. OASIS: eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 1 Feb 2005.
17. Anderson, A.: Web Services Profile of XACML (WS-XACML) Version 1.0, WD 8. OASIS XACML Technical Committee 12 December 2006.
18. University of Salford: Schema for Obligation of Trust (OoT). http://infosec.salford.ac.uk/names/oot/ootSchema/ December 2006.
19. Mbanaso, U., Cooper, G.S., Chadwick, D.W., Proctor, S.: Privacy Preserving Trust Authorization using XACML. Second International Workshop on Trust, Security and Privacy for Ubiquitous Computing (TSPUC 2006) Niagara-Falls, Buffalo-NY, 2006.