Obligations of trust for privacy and confidentiality in distributed transactions

Mbanaso, UM, Cooper, GS, Chadwick, DM and Anderson, A

http://dx.doi.org/10.1108/10662240910952328

Obligations of Trust 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

Purpose – This paper describes a bilateral symmetric approach to authorization, privacy protection and obligation enforcement in distributed transactions. We introduce the concept of the Obligation of Trust (OoT) protocol as a privacy assurance and authorization mechanism that is built upon the XACML standard. The OoT allows two communicating parties to dynamically exchange their privacy and authorization requirements and capabilities, which we term a Notification of Obligation (NoB), as well as their commitments to fulfilling each others requirements, which we term Signed Acceptance of Obligations (SAO). We describe some applicability of these concepts and show how they can be integrated into distributed authorization systems for stricter privacy and confidentiality control.

Design/Methodology/Approach – Existing access control and privacy protection systems are typically unilateral and provider-centric, in that the enterprise service provider assigns the access rights, makes the access control decisions, and determines the privacy policy. There is no negotiation between the client and the service provider about which access control or privacy policy to use. We adopt a symmetric, more user-centric approach to privacy protection and authorization, which treats the client and service provider as peers, in which both can stipulate their requirements and capabilities, and hence negotiate terms which are equally acceptable to both parties.

Findings – We demonstrate how the Obligation of Trust protocol can be used in a number of different scenarios to improve upon the mechanisms that are currently available today.

Practical Implications – This approach will serve to increase trust in distributed transactions since each communicating party receives a difficult to repudiate digitally signed Acceptance of Obligations, in a standard language (XACML), which can be automatically enforced by their respective computing machinery.

Originality/Value – This paper adds to current research in trust negotiation, privacy protection and authorization by combining all three together into one set of standardized protocols. Furthermore, by providing hard to repudiate Signed Acceptance of Obligations messages, this strengthens the legal case of the injured party should a dispute arise.

Keywords XACML, trust, privacy, obligations, trust negotiation, SAML, authorization

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 user's PII may become the SP's resource once it has been transferred, and 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 (Bertino *et al*. 2004).

Often, confidentiality is used as a substitute for privacy but they are not identical. It is important to accurately differentiate between them in order to identify the associated challenges and risks. Simply put, whilst confidentiality is the process of making sure that only the right ('legitimate') party gets access to information, privacy ensures that this legitimate party treats the information in the right way and only uses it for the purposes for which it is intended. Privacy is governed by factors such as legislation, laws, guidelines and principles that cut across national borders (OECD, 2000).

1.1 Overview of Privacy and Confidentiality Challenges

In distributed environments, there are three functional security components that peers in communication have to address: Authentication, Authorization and Trust. Whilst authentication addresses the question of 'who are you?', authorization answers the question 'what can you do?' after we know who you are. In contrast, trust deals with 'how reliable are you?' which has to be based on 'who says this about you?' when the trustor has no prior direct knowledge about the trustee. This requires the trustor to have confidence in the entity that makes assertions about the trustee. The new federated architecture shows that authentication and authorization can operationally be managed in autonomous security domains, and it is trust which binds the domains together. For example, authentication could be handled at the initiator's domain, whilst access control could be handled at the service provider's domain based on attributes asserted by a third domain. Consequently trust relationships must be established between all the interacting domains, so that the service provider can trust the authentication statement made by the initiator's domain, and the attribute statements asserted by the third domain. Trust provides the mechanism for validating the authenticity of the various attribute and privilege assertions that the interacting

domains use as the basis for allowing access to protected resources. Trust may be direct or indirect/transitive e.g. A trusts B (direct trust), B trusts C (direct trust) and A trusts C because of B's trust in C (indirect/transitive trust). Often, in distributed environments, the common practice is to rely on indirect trust brokered by trusted third parties (TTPs) that issue signed assertions to multiple clients. These clients can then use these assertions, and their mutual trust in the TTP, as a basis for trusting each other. This is how public key infrastructures work today, and it is a foundation upon

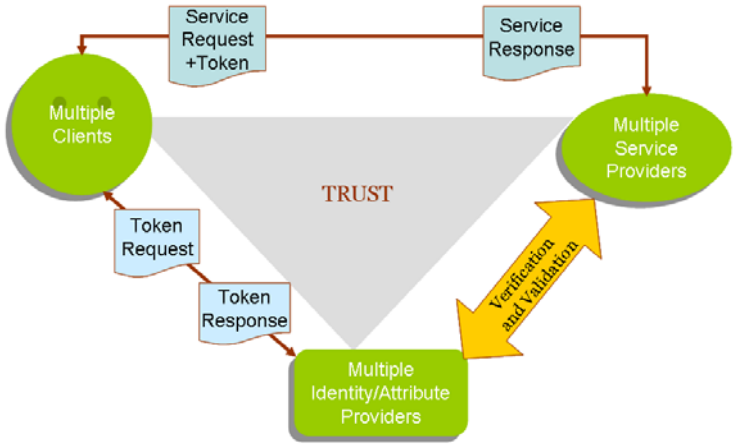

Figure 1 Actors in Federated Access Management

which federated systems are built.

However, web services provisioning introduces a far more extensive and dynamic environment for complex transactions that makes brokered trust insufficient to protect the privacy and confidentiality of all transactions. Figure 1 depicts the multiple actors in federated access management. Service clients can have multiple identities issued by autonomous identity providers, and the identity providers can broker trust among many clients and providers. It is imperative therefore that trust in this case has to be established between people and people, between people and services, and between services and services. This highlights the following challenges and risks:

- What is the accountability of parties in relation to compromised personal identifying information (PII)?
- What level of audit takes place regarding how PII is accessed?
- How can a user be assured that a service provider's privacy promises will be supported by robust technological means?.
- How are disputes and liabilities handled? There is a need to establish a respected channel for handling and resolving disputes.
- How can the process of contractual negotiations be automated? Traditional methods are overly time consuming and costly.

- Whose fault is it in the event of a problem with shared attributes or data? Who is financially liable? Is there any hard-to-repudiate evidence that parties can use in courts of law to support their claims?

We conclude that technological means alone will never be able to answer all the above questions, and that regulatory compliance, disputes resolution and assurance mechanisms will require underpinning with local and trans-border legislation, laws, guidelines and principles. Thus the legal and regulatory system is the only significant trusted third party that is big enough and ubiquitous enough to broker trust between all the parties involved in federated web services. However, technology should be able to contribute to the resolution of the above challenges, by providing high quality, difficult to repudiate information that can be utilized by the legal and regulatory system when the need arises.

The emerging web services provisioning may require that in B2B transactions, both parties dynamically exchange service level agreements (SLA) or business level agreement (BLA) in order to assess the mutual benefits and associated risks. This will eliminate the static contractual agreements that are too time consuming to establish in order to address the opportunities that arise in dynamic business environments. One way to achieve this would be for each party to issue to the other a proof of acceptance of the requirements contained in the SLA or BLA of the other party. Enabling the runtime exchange of these requires a bilateral symmetric approach to allow the communicating parties to indicate their willingness to accept the constraints imposed by the other party, before being 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 (Bertino *et al*., 2004) has been proposed to address this dilemma, but as will be pointed out later it has its limitations. We therefore approach the topic in a slightly different manner. We propose a technical solution that derives its concepts and message exchanges from open standards, and a business solution that derives its trust from the enforcement that is provided by the legal and regulatory infrastructure. 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

[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.

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 and also describes how the matching of obligation constraints and proof of acceptances is achieved using the XACML standard. It also describes the binding of the OoT protocol onto the standard SAML protocol. Section 4 provides an example of the use of the model whilst section 5 concludes the paper.

2. Related Research

The Platform for Privacy Preferences (P3P) (W3C, 2002b) 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 (W3C, 2002a) 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. Although P3P covers most of the basic principles of privacy (OECD, 2000), the fact that it has not satisfactorily resolved the requirements for bilateral privacy negotiation limits its use in access control.

Shibboleth (Morgan *et al.*, 2004) from Internet2 provides a mechanism for federated access management based on the SAML security standard (OASIS, 2005). 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 personally

identifying attributes to complete the service. But this is unlikely to be the case in most business 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 (OECD, 2000).

ID-WSF from the Liberty Alliance (Liberty Alliance Project, 2006) is an open standard for federated identity management that is built upon the extensibility of SAML security assertions (OASIS, 2005). It provides a framework for the discovery and communication of identity information among federated organizations. 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 federated organizations.

Microsoft, IBM and VeriSign started to work on a set of specifications for web services security (called "WS-*") for their next generation platform of Web services. The work was then passed to OASIS and W3C for standardization by a broader group of participants, which has resulted in a number of specifications being published including Web Services Security (OASIS, 2006), Web Services Trust (OASIS, 2007) and Web Services Policy (W3C, 2007). WS-Security provides mechanisms for securing web services SOAP messages with integrity and confidentiality. WS-Trust uses the messages secured by WS-Security to allow security tokens to be issued, exchanged and validated by Security Token Services (STSs) as a means of brokering trust between web services. WS-Policy is designed to enable services to advertise their requirements (especially authorization requirements) that a requesting party must satisfy in order to use the services; as well as the capabilities that their services may offer 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 meant to be consumed by Policy Decision Points (PDPs).

One approach that addresses bilateral access control is trust negotiation (Bertino *et al*., 2004), including the Automated Trust Negotiation (ATN) technique (Winsborough and Li, 2002). 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 (Seamons *et al*., 2002) (Winsborough and Ninghui, 2002) (Seamons *et al*., 2005) (Bertino and Squicciarini, 2003). Recent initiatives in preserving privacy (Pau, 2006) (Preibusch, 2006) also favour the use of negotiation techniques for solving privacy problems. 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 address the issue of sending obligations for future actions or providing a mechanism whereby the relying party can convey proof of acceptance of obligating constraints such as the assurance that the attributes contained in the assertions will be used in accordance with the party's privacy preferences. The first researchers to integrate obligations with trust negotiation were Skogsrud *et al.* (Skogsrud *et al*., 2004). Their Trust-Serv framework uses a finite state machine to model trust

negotiation and has the ability to transfer a signed contract along with credentials from the service provider to the client. The client may then sign and return the contract to the service provider. But their contracts are currently without syntax or semantics, and as the authors point out, the monitoring of these obligations/contracts is beyond the scope of their work. Our research addresses this gap. Spantzel *et al*. (Spantzel *et al*., 2007) introduce 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. We address this issue here.

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 standard 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 in a standard format that can be automatically enforced by computing machinery.

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 addresses the problem that a requesting party currently has no means of enforcing obligations placed on a remote party. In policy based 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. XACML (OASIS, 2005) 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. Similarly the SP's PEP will have difficulty enforcing any obligations that are returned by its PDP that place requirements on the service requestor. Given this, it makes sense to address the remote enforcement of obligations by allowing each party to transfer their obligations to the other party in a standard format, and to convey back to the other party an acceptance or rejection of the obligations they have received. 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, either (i) sends back a signed acceptance (SAO) of the constraints it accepts and the commitments it requires, or (ii) initiates more service negotiations with its own NoB, or (iii) 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. Thus our system provides for the automatic strong technical enforcement of the promises that are made.

A simple example follows, illustrating how the addition of privacy related obligations could be beneficial in a transaction involving trust. The example starts from a relatively simple online transaction whereby a client is accessing a commercial web site and is required to submit personal information, including his credit card details. The client duly submits the required information to the SP and the transaction is completed. In this case, the service provider's requirements for authentication and for assurance that payment will be received are based on the provision of the credit card's details by the client. However, this is a unilateral arrangement, in which the SP's requirements have to be met by the client, but the client has no way of specifying or placing requirements on the SP. With such sensitive information as a credit card number, it could be that the client wishes to specify such requirements.

In this case, the respective requirements of the client and the SP may be specified using the OoT protocol as follows:

1. The client initiates a request to the SP asking for a service and receives a Notification of Obligation stating the enterprise's requirements for personal details and credit card details;
2. The client returns a NOB with requirements specifying that the SP must delete credit card details immediately after use and only use the personal details for internal processing purposes;
3. The SP replies with a Signed Acceptance of Obligations agreeing to both requirements, as well as agreeing to provide the requested services.

4. The user is now able to reply with his own SAO along with his credit card and personal details.
5. The SP then completes the transaction, providing the requested service to the client.

In this case, it can be seen that bilateral signed agreements are reached before either party releases information, goods or services to the other. A key point here is that each party's requirements needs to be bound to the other party's requirements in order to allow a signed agreement ("OoT") to be reached.

In web services environments, it may be that in some transactions SPs will want potential requestors to know exactly what access requirements are needed in order to invoke their services. In contrast, there may be other situations in which a web service will not want to publish the full access requirements, but a subset of them which can act as a first-level filter to minimize undesirable exchanges. This fits scenarios in which an SP considers it a business risk to release the full access requirements in a single shot to arbitrary clients, but would prefer to make the access requirements known incrementally as more trust is gained between it and the client. In fact the subset could help the potential requestors to know whether they are likely to satisfy the service requirements before disclosing information of their own, as well as to be able to calculate the potential risk in the interactions. In other scenarios, some web services may not have access to the authorization decision policies within their domain, or simply may be running within constrained environments and are unable to run a Policy Decision Point (PDP) engine. In such cases, the devices would rely on a trusted 3rd party's PDP to provide a signed authorization token to indicate that it has evaluated the requester's request against its access policies and certify that the request is allowed. In other cases, it might be that a web service or client would require a potential interacting partner to possess a particular role, e.g. 'Liberty Consortium Membership', 'certified ACCA accountant', etc. in order to interact. The client, or a service with the right privileges, can request a client's role assertion from the particular Role Attribute Authority (RAA) prior to service interactions, and the service can then validate the role assertion to determine whether the client actually possesses such a role or not. Where privacy and confidentiality are considered important, parties may rely on the facilities of WS-XACML's Requirements and Capabilities (Anderson, 2007) to specify their security preferences.

A further example in Figure 2 serves to illustrate the use of the proposed protocol. We present a scenario involving access control using policy based mechanisms, security tokens issued by trusted third parties, and then show how the OoT may be used to enrich the exchange to include additional forms of trust establishment.

In Figure 2 (i), a client requires a role attribute with the value 'purchasing officer' in order to invoke a web service to obtain a confidential price list from an Enterprise SP. The following steps describe the interactions among the various actors:

1. The client initiates and sends a role attribute request to his Role Attribute Authority (RAA)/Security Token Service (STS) (that is also trusted by the Enterprise SP). This requires the client to authenticate to the RAA/STS in order for the latter to determine the role(s) assigned to this client;
2. The RAA/STS issues a signed role assignment token to the client;

3. The client constructs a request message, containing the role assignment token, and sends it to the Enterprise SP, asking for the confidential price list;
4. The Enterprise SP (or its STS) validates the role assignment token containing the client's role as a *purchasing officer*, passes this to its internal XACML PDP (as a subject attribute) which evaluates the request against its local policy. If the decision is granted, it returns the price list to the client

In this example the client can authenticate and obtain identity or role tokens from its local security domain, which are trusted by the service provider, thereby satisfying the latter's requirements that the client is authenticated and authorized. Using our OoT

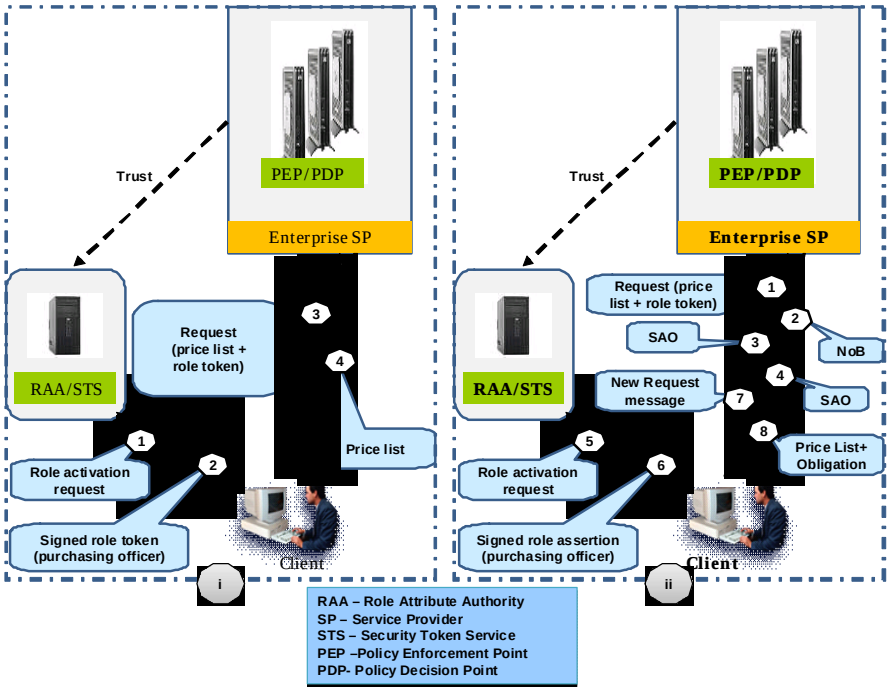

Figure 2 WS-XACML Contexts in Distributed Environments

protocol, it would be possible for the service to express its role requirements in the *Requirements* section of a WS-XACML assertion, whilst the client records its capability to activate the role in the *Capabilities* section. Whilst this approach is not really necessary in this simple case, it does provide an important generalization that will allow for more complicated requirements than simple access control to be supported. For example, it might be that the service provider also requires an assurance that the confidentiality of its price list will be maintained. This requirement

could equally be represented using the WS-XACML assertions mentioned earlier. In this case, depicted in Figure 2(ii), the exchange will proceed as follows:

1. The client initiates a request to the Enterprise SP for the confidential price list.
2. The Enterprise SP sends a Notification of Obligation stating the enterprise's requirements for the *purchasing officer* role and confirmation that the price list will not be disclosed by the recipient
3. The client returns a Signed Acceptance of Obligation confirming that it will keep the price list confidential and that it is able to furnish the required role.
4. The Enterprise SP replies confirming the SAO.
5. The client initiates and sends a role attribute request to its Role Attribute Authority (RAA)/STS (that also is trusted by the Enterprise SP). This requires the client to authenticate to the RAA/STS in order for the latter to determine the role(s) assigned to this client;
6. The RAA/STS issues a short lived signed SAML attribute assertion (containing the role) to the client;
7. The client constructs a request message, containing the role assertion, and sends it to the Enterprise SP, asking for the confidential price list;
8. The Enterprise SP validates the client's role assertion as a *purchasing officer* of the Enterprise and passes the client's valid role to its internal XACML PDP (as a subject attribute) which evaluates the request against its local policy and returns granted and an obligation to keep the price list confidential. The SP returns the price list and obligation to the client.

In an even more complicated variant of above, the client might wish that the Enterprise SP does not publicize that it has requested the price list. In this case the client, in step 3, would send a NoB to the Enterprise requiring it to keep its identity information private and to use it only in the current transaction. The Enterprise could then decide to either accept this obligation or not. The examples shown in this section illustrate just some of the needs that are addressed by the OoT protocol. A more substantial example, involving more complex combinations of requirements, will be given in section 4.

3.1 OoT Encoding Scheme

The Web Services Profile of XACML (WS-XACML) (Anderson, 2007) describes a way of encoding XACML related information so that it can be transferred between communicating parties. WS-XACML specifies the format of four types of information:

- an authorization token or credential for carrying an authorization decision between web services,
- a policy assertion for carrying policy requirements and capabilities between clients and servers. Both authorization and privacy policy assertion types are specified,

- P3P policies using the privacy policy assertion type mentioned above, so that both parties can specify their privacy preferences and have them matched by the receiving party, and
- XACML Attributes as SAML assertions, to be carried in SOAP Message Headers.

More formally, the WS-XACML Assertion Type is an abstract framework that describes an entity's Web Services 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, i.e. XACMLPrivacyAssertion 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 own Requirements are satisfied). The inner (green) box in Figure 3 depicts the WS-XACML Privacy Assertion. 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 3. 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 the *XACMLAssertion* 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.

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 these standard elements. The NoB can be expressed as an instance of a *XACMLPrivacyAssertion* in which the desired obligating constraints are placed in the *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 sender 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. The recipient duly returns its SOA to the sender, only now the placement of the obligations has been reversed, since its signed *Capabilities* contain the obligations from the original sender's *Requirements* and vice versa. This is equivalent to today's exchanges of signed paper contracts, which contain the commitments and obligations of both parties. Figure 3 shows the extensions of WS-XACML and SAML that map into our Obligation of Trust model. The OoT schema is available at (University of Salford, 2006), 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).

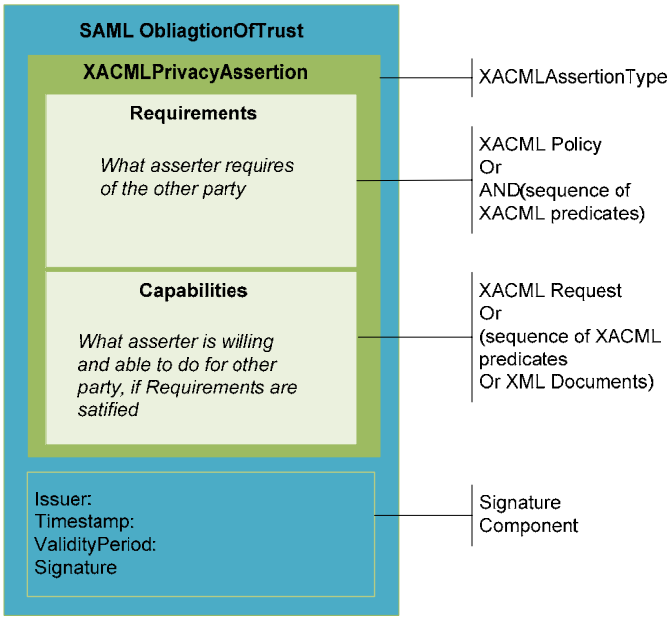

Figure 3 SAML Obligation of Trust Model

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 this recipient, this can be sent as a P3P policy STATEMENT of PURPOSE expressed as a WS-XACML constraint as shown in Figure 4.

```
<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</AttributeValue

                <AttributeValueDataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
                expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/admin</AttributeValue>

                <AttributeValueDataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
                expression">//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours</AttributeValue>

</Apply>

</Apply>
```

Figure 4 Example of WS-XACML constraint on P3P PURPOSE.

3.2 OoT Protocol Scheme

Figure 5 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 6 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 unacceptable or insufficient for Party A, then A will either terminate the session or return a NoB with *Requirements* that supercede

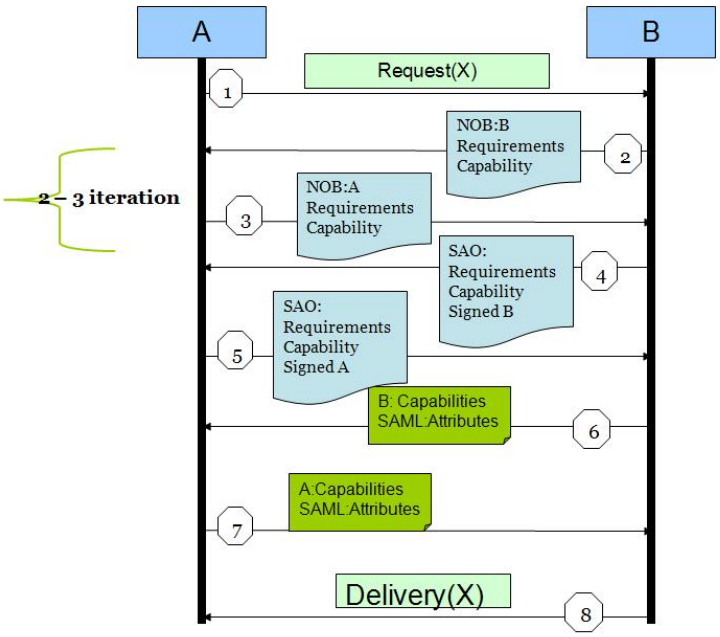

Figure 5 OoT Protocol Sketch

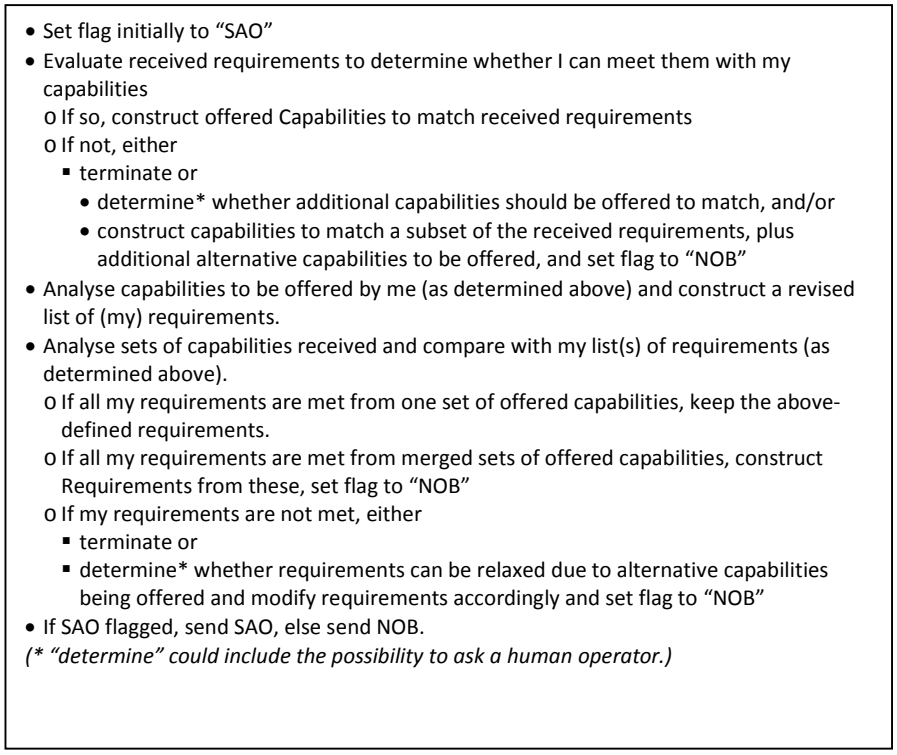

Figure 6 Outline Algorithm for handling a NoB

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 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 the parties deliver their respective commitments. Note that there is scope for efficiency gains in the combined number of OoT and application level protocol exchanges, depending upon which party is the first to send the SAO, and the nature of the application exchange. For example, in Figure 5 party A, having already received party B's SAO, could send its attribute assertions to B in the same message as its SAO i.e. combine messages 5 and 7, in which case party B could then combine messages 6 and 8 in its response. The exact sequence of OoT and application level message exchanges can be defined by each specific application according to its needs.

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 a *Capability* to provide 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 XACMLPrivacyAssertion containing a *Capability* (able to release PII to third parties) and a new *Requirement* (level of compensation required) are added to the user's set of XACMLPrivacyAssertions, for this and future use, and a signed SAO is sent to Service C.

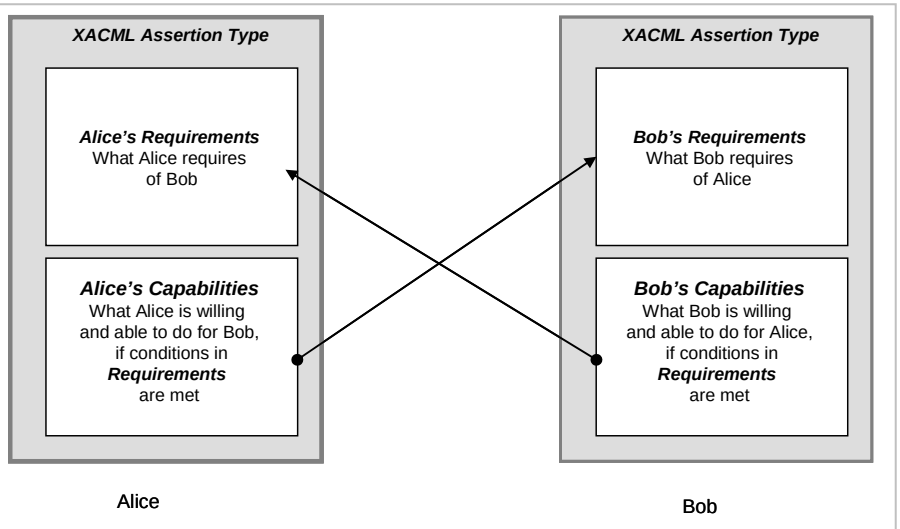

Figure 7 Matching of Two WS-XACML Assertion Type

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 7 illustrates the matching of two WS-XACML Assertions. Two *XACMLAssertions* match if, for each assertion, all constraints 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.

3.3. SAML OoT SOAP Binding Profile

The SOAP model provides "extensibility", allowing other messaging protocols to be layered on top of it in a standardized way. This convenient flexibility provides a rich mechanism for layering the OoT protocol with other existing security protocols including Web Services security (WS-Security), which are a set of specifications that describe the means for providing various types of security protection for SOAP messages. Whilst WS-Security has defined SOAP profiles for authentication, data integrity and data confidentiality at the message layer, it does not cater for privacy protection as described in this paper. Consequently there is a need to describe a standard way to use WS-XACML to address mutual privacy and confidentiality in Web services scenarios. First we define a SOAP binding profile for OoT. This provides an appropriate mechanism to ensure that independently implemented compliant systems can interoperate using standard messaging protocols. The OoT protocol is an extended variant of the SAML request-response mechanism, which allows us to layer the OoT within a <wsse:security> container in the SOAP header context as depicted in Figure 8. The consequence is that the OoT mechanism can be added to the existing suite of security provisions. Figure 8 shows the layers of individual components that provide a standard way to enable compliant systems to process the assertions in context.

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.

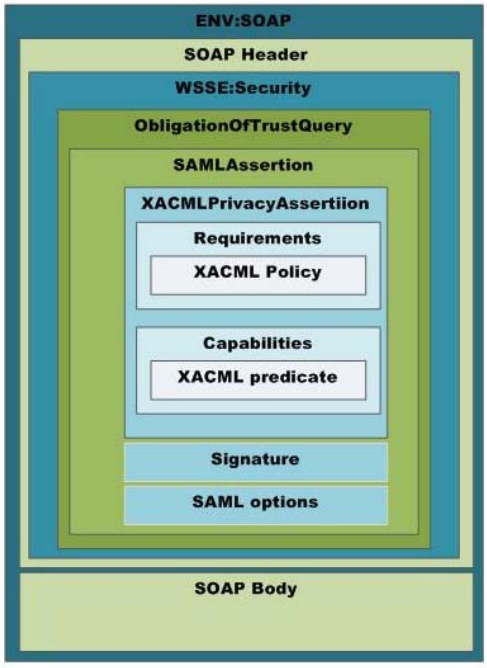

Figure 8 SAML OoT SOAP Binding Profile

XACMLPrivacyAssertions (ITS)
XACMLPrivacyAssertions1
Requirements
Client Name
Capabilities
Standard Price List
PURPOSE: PII used internally for this transaction
RETENTION: PII kept only until transaction is completed
RECIPIENT: PII not given to any 3rd party

XACMLPrivacyAssertion2
Requirements
Client Name
IATA membership certificate
Certified Quarterly Sales
Price List not publicized to 3rd parties
Capabilities
Tailored Price List
PURPOSE: PII used internally for this transaction
RETENTION: PII kept only until transaction is completed
RECIPIENT: PII not given to any 3rd party

Fig. 9. ITS's Internal XACMLPrivacyAssertions

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

Fig. 10. Customer's Internal XACMLPrivacyAssertion

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, for example, increasing discounts to customers with larger sales figures. The ITS requires prospective clients to provide 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 such as sales figures, 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 9 depicts the ITS's internal XACMLPrivacyAssertions and figure 10 is the customer's internal XACMLPrivacyAssertion. Looking at the 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 that may be used in the case of disputes.

Figure 11 illustrates the interactions between the ITS and customer.

1. The customer requests an authentication token from a third party authentication provider that is trusted by the ITS.
2. The signed token is returned to the customer.
3. The customer requests a price list from the ITS and presents the authentication token.
4. The ITS sends its two sets of requirement and capabilities as shown in Figure 9 in the form of a NoB.
5. The customer determines that the capabilities offered by the ITS can satisfy its own requirements, and that its capabilities can satisfy the ITS's stringent requirements for a Tailored Price List, so creates and signs an SAO based on this and returns it to the ITS.
6. The ITS acknowledges the SAO by returning its own
7. The customer provides the requested information, including its IATA certificate, Certificate of Incorporation and certified sales figures.

8. The ITS sends the requested price list to the customer.

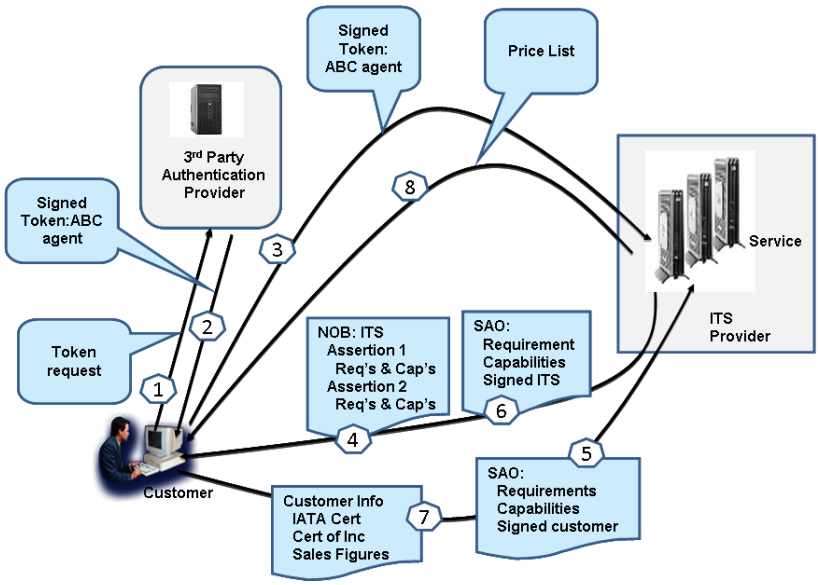

Figure 11 The ITS and Customer Interactions

Each party now has in its possession a signed agreement from the other party indicating agreement to the terms that were agreed. Other variations of this exchange are possible as determined by the specific application. For example in Step 1 the customer could contact a public service of the ITS before authenticating with a trusted third party, and could be returned a NoB based on the first Privacy Assertion only. The customer could decide this is not good enough and return its own NoB stating its requirement for a Tailored Price List based on the contents of Figure 10, to which the ITS could then return a signed SAO containing its second Privacy Assertion. The OoT protocol we have defined is flexible enough to fit any of these application requirements.

5. 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 provides technical solutions to support possible social/judicial solutions for security assurance in distributed open systems. This mechanism demonstrates a secure way of using P3P and other 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.

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, through the use of standard XACML policies, has the potential to 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 both parties exist as legal entities that can be sued if violations occur. This requires either a robust PKI system to exist or some other TTP mechanism to establish whether the subject of a certificate is a legal entity, and will put meaningful identifying information in the issued certificate. Extended validation certificates (CA/Browser Forum, 2008) are one such technology that are currently being rolled out to SSL/TLS web servers. 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 (Mbanaso *et al*, 2006) 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 that is 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.

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.

6. References

Anderson, A. (2007), "Web Services Profile of XACML (WS-XACML) Version 1.0", Working Draft 10, *OASIS XACML Technical Committee*, 10 August, available at http://www.oasis-open.org/committees/download.php/24950/xacml-3.0-profile-webservices-v1-wd-10.zip (accessed 24 October 2008).

Bertino, E.F.E., Squicciarini, A. (2003), "X-TNL: An XML-based Language for Trust Negotiations". *4th IEEE International Workshop on Policies for Distributed Systems and Networks*, Lake Como, Italy, June, pp 81-84.

Bertino, E., Ferrari, E., Squicciarini, A. (2004), "Trust Negotiations: Concepts, Systems and Languages", *Computing in Science & Engineering*, Vol. 06, No. 4, pp. 27-34.

CA/Browser Forum (2008), "Guidelines For The Issuance And Management Of Extended Validation Certificates", available at http://www.cabforum.org/documents.html (accessed 22nd October 2008).

Liberty Alliance Project. (2006), "Liberty ID-WSF Web Services Framework Overview Version: 2.0", available at http://www.projectliberty.org/liberty/specifications__1 (accessed 24 October 2008)

Mbanaso, U., Cooper, G.S., Chadwick, D.W., Proctor, S. (2006), "Privacy Preserving Trust Authorization using XACML", *Proc. 2nd International Workshop on Trust, Security and Privacy for Ubiquitous Computing (TSPUC 2006)*, Niagara-Falls, Buffalo-NY, June, pp. 673-678.

Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W., and Klingenstein, K. (2004), "Federated Security: The Shibboleth Approach", *Educause Quarterly*, Vol. 27, No. 4, available at http://connect.educause.edu/Library/EDUCAUSE+Quarterly/FederatedSecurityTheShibb/39889 (accessed 24 October 2008).

OASIS (2005) "Security Assertion Markup Language (SAML) V2.0", March, available at http://saml.xml.org/saml-specifications (accessed 24 October 2008).

OASIS (2005), "eXtensible Access Control Markup Language (XACML) Version 2.0". February, available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical (accessed 24 October 2008).

OASIS (2006), "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), OASIS Standard Specification", available at http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (accessed 24 October 2008)

OASIS (2007), "WS-Trust 1.3, OASIS Standard", available at http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html (accessed 24 October 2008)

OECD (2000), "Fair Information Practices In The Electronic Marketplace A Report To Congress", May.

Pau, L.-F. (2006), "Privacy Negotiation and Implications on Implementations", *Proc. W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement*, available at http://www.w3.org/2006/07/privacy-ws/papers/ (accessed 24 October 2008)

Preibusch, S. (2006), "Privacy Negotiations with P3P", *Proc. W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement*, available at http://www.w3.org/2006/07/privacy-ws/papers/ (accessed 24 October 2008)

Seamons, K. E., Winslett, M., Yu, T., Yu, L., Jarvis, R. (2003), "Protecting Privacy during On-line Trust Negotiation", *LNCS Privacy Enhancing Technologies, Vol. 2482/-1/*, Springer, Berlin / Heidelberg, pp 249-253.

Seamons, K. E., Ryutov, T., Zhou, L., Neuman, C., Leithead, T. (2005), "Adaptive Trust Negotiation and Access Control", *Proc. 10th ACM Symposium on Access Control Models and Technologies*, Stockholm, Sweden, pp 139-146.

Skogsrud, H., Benatallah, B., and Casati, F. (2004), "A trust negotiation system for digital library Web services", *International Journal on Digital Libraries*, Vol. 4, No. 3, pp. 185-207

Spantzel, A.B., Squicciarini, A.C., Bertino, E. (2007), "Trust Negotiation in Identity Management", *IEEE Security & Privacy*, Vol. 5, No. 2, pp. 55 - 63.

University of Salford (2006) "Schema for Obligation of Trust (OoT)", available at http://infosec.salford.ac.uk/names/oot/ootSchema/ (accessed 22nd October 2008).

W3C (2002a), "A P3P Preference Exchange Language 1.0 (APPEL1.0)", available at http://www.w3.org/TR/P3P-preferences/ (accessed 24 October 2008).

W3C (2002b) "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification" available at http://www.w3.org/TR/P3P/ (accessed 24 October 2008)

W3C (2007), "Web Services Policy 1.5 - Framework (WS-Policy)" available at http://www.w3.org/TR/ws-policy/ (accessed 24 October 2008)

Winsborough, W. H., Li, N. (2002), "Towards Practical Automated Trust Negotiation", *3rd IEEE International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, June, pp 92-103.

Winsborough W.H, Ninghui L., (2002), "Protecting sensitive attributes in automated trust negotiation", *Proc. 2002 ACM workshop on Privacy in the Electronic Society*, Washington DC, November, pp.41-51.