Privacy Preserving Trust Authorization Framework Using XACML

U.M. Mbanaso, G.S. Cooper
Computing, Science and Engineering
University of Salford, UK
u.m.mbanaso@pgr.salford.ac.uk,
g.s.cooper@salford.ac.uk

D.W.Chadwick
Computer Department
University of Kent, UK
d.w.chadwick@kent.ac.uk

Seth Proctor
Sun Microsystems Laboratories
Burlington, MA USA
seth.proctor@sun.com

Abstract: *Nowadays many organisations share sensitive services through open network systems and this raises the need for an authorization framework that can interoperate even when the parties have no pre-existing relationships. Trust Negotiation is the process used to establish these first relationships, through the transfer of attributes, embedded in digital credentials, between the two parties. However, these attributes may themselves be considered sensitive and so may need protection from disclosure. In some environments, the policies that govern the protected services may also be considered sensitive and their release to arbitrary strangers may leak confidential business information. Thus, the electronic services, the policies that control access to them, and the digital credentials used to gain access may all be sensitive and require access protections. This paper describes how to unify the protection of services, sensitive credentials and policies in a synchronised trustworthy manner. We propose a trust authorization framework (TAF) that builds on the capabilities of XACML to support the bilateral exchange of policies and credentials through trust negotiation. Our framework addresses privacy and trust issues, and considers services, credentials, and authorization policies protected resources whose access is subject to credential proof and trust level validation*

Keywords: *XACML, authorization, privacy protection, trust establishment, distributed systems, access control*

1.0 Introduction

Authorization ensures that resources can be accessed only by parties who have the right privileges. Thus, the resource gatekeeper requires some level of trust be established before sensitive information can be released. Service requesters are required to submit sufficient authorization credentials before access will be granted. Wherever people are involved in the exchange of digital information, such as credentials, privacy becomes an issue of some concern. The use of personal, sensitive information to gain access to a resource in a distributed environment raises an interesting paradox. On the one hand, in order to make the services and resources accessible to legitimate users the authorization infrastructure requires the users' attributes. On the other hand, the users may not be ready to disclose their attributes to a remote service provider without determining exactly who the provider is and how personal attributes will be used. Thus, privacy [1] [2] [7] is a critical consideration for authorization environments. One approach for addressing these privacy concerns is to employ a bilateral exchange of policies and credentials between the parties involved in the transaction, so that they can decide what to give and/or get from each other. This process is known as trust negotiation in the literature [8]. However, the policies and credentials may themselves be sensitive. Consider the following motivating example. A Secret Service (SS) offers online training both for her agents and friendly secret agent services. The service requires that each participant present a role Attribute Certificate (certificate), a security assertion digitally signed by the participant's security authority which binds the holder's attributes to the holder. Whilst the policy that governs this service prevents unauthorized access to its resources, the policy does not protect the fact that SS offers training to friendly organisations, which is another sensitive piece of business information. To address this vulnerability, it is desirable that the policy that governs access to the training resources be protected from being disclosed to arbitrary strangers. To prevent arbitrary disclosure of sensitive policies, access to the policies need to be protected. On the other hand, an agent requester cannot give out her role certificate to any service that poses as the SS webserver. The agent would like some proof that shows the server can be trusted. Imperatively, to solve the arbitrary disclosure of sensitive policies [2] and digital credentials, parties require a mechanism to gradually establish a trusted relationship. Trust relationships can be established between service providers and requesters through the exchange of information in a well understood fashion [12]. The information usually contains policies and security assertions, issued by Attribute Authorities (AAs), which describe the properties of the holders. The exchange of this information is done in such a manner that the security assertions are unforgeable and can be verified and validated [22].

One promising mechanism for this problem is the eXtensible Access Control Markup Language (XACML),a standard created in OASIS [3] that is gaining prominence and enjoying wide spread support among major stakeholders in authorization technology. This standard defines a general-purpose, flexible authorization policy language and a query/response format.. Though XACML is a rich framework, it intentionally does not address how to preserve the privacy of authorization entities. For this, we require well-defined trust relationships between the participants, but first time business partners may not have pre-existing relationships. Therefore, a mechanism for gradual building of trust is desirable.

Trust negotiation management systems have been proposed by researchers as one effective way to guarantee the confidentiality of authorization information. Trust establishment is a well researched concept [5] [2] [6]. However, existing efforts in this area have not been standardized and do not fit into any authorization standard such as XACML. This work investigates how XACML can fit into trust authorization management systems. Our

approach and strategy is to explore existing concepts, and where necessary, extend them to accomplish our goal.

Adding a trust component to XACML will extend its usage in open systems where transaction parties require trust establishment before they can share their sensitive information. In this paper, we describe our proposed XACML Trust Authorization Framework (XTAF). XTAF is a loosely coupled architecture with a trust component that protects authorization information (policies and credentials) layered such that it integrates seamlessly into any XACML compliant authorization engine with minimal effort. We expose different ways that the XACML policy language can be used to support bilateral exchange of policies and credentials, and protect unauthorized access to services. We introduce a Trust Authorization Service Handler (TASH) to handle sensitive authorization information. This supports runtime bilateral authorization operations between two or more parties. The framework takes care of protecting unauthorised access to enterprise resources such that users' privacy is balanced against the business aspects and public interests that may be adversely affected by transactions in an unregulated privacy regime. Using existing standard such as XACML has the benefit of promoting interoperability and reducing the effort needed to integrate with existing applications.

The rest of this paper is organised as follows: In section 2, we highlight some of the security challenges in open systems authorization. Section 3 gives a brief overview of the XACML authorization framework. Section 4 presents an overview of the XACML Trust Authorization Framework, as well as showing how the XACML policy language can be used for trust establishment, privacy, and resource controls in a synchronized manner. In section 5 we illustrate, through a hypothetical example, the usage of our framework based on the concepts presented in this paper. Section 6 presents related research work, and section 7 concludes the paper with a summary and future research work.

2.0 Authorization in Distributed Open Systems

Authorization is the mechanics of controlling access to sensitive resources by comparing local policies that govern access to resources against the credentials submitted by requesters. Traditional authorization is a one-shot process that requires the user to submit credentials unconditionally, irrespective of whether the services adhere to the privacy preferences of the users. Traditional authorization systems also make an unrealistic assumption that service requesters have previous knowledge of access control requirements; in open systems with diverse and unbounded users and service providers, this may not be the case. Consequently, service requesters are made to submit more attributes than necessary, which potentially exposes them to privacy risks. Resources can grow and shrink, making dynamic disclosure of access control requirements desirable at runtime. This makes it possible to add or remove services without worrying about the implications for service requesters. At request time, the access requirements can be made known to service requesters and they can check whether they can meet those requirements. Conversely, the service requesters need to evaluate the risk of giving out their attributes by determining the degree to which they are prepared to trust the service providers.

Figure 1 gives a simple picture of entities involved in authorization process and points of privacy concerns. To enforce privacy in authorization, a loosely coupled architecture is required to synchronise the protection of business services, as well as users' privacy in a manner that

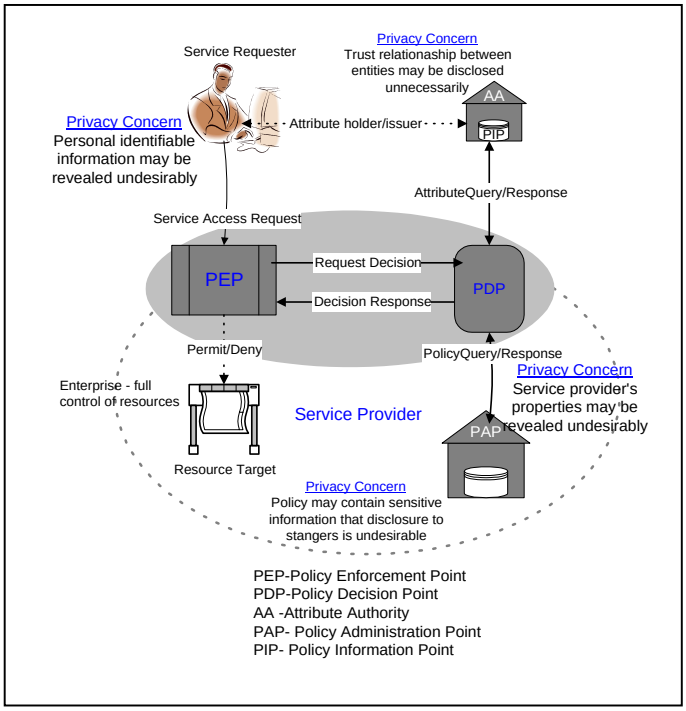

Figure 1 Privacy in Authorization Infrastructures

guarantees information flow and availability. It is worthwhile to note that these entities may not necessarily exist within the same trusted environment. Thus, the environment affects the kind of privacy concerns and expectations. For instance, with authorization in a distributed environment which is conducted over public network such as the Internet, the users' and the service providers' information is exposed to a number of threats. In this section, we highlight some of the security challenges of authorization particularly in distributed environments. .

2.1 Security Challenges

Authorization in distributed open systems presents a number of significant challenges. First, the service providers and requesters are unlikely to belong to the same security domain in all scenarios. In some of these cases, service requesters may include users with or without pre-existing relationships with the service provider. Second, the authorization credentials may not necessarily be issued by one Attribute Authority (AA), thus a chain of third parties may be involved in the authorization process. Thus, privacy and trust are critical security constructs that must be

addressed along side the effective control of service providers' resources. In this context, some of the key authorization security issues can be summarized as follows:

- Service providers' effective resource control
- Service requesters' personal sensitive attributes need privacy protection
- Service providers' authorization information requires privacy protection
- Privacy of third party affiliates – unauthorised disclosure of trust relationship between AAs and the credential holders
- Verification and validation of assertions issued by AAs about the service providers and requesters
- When service can be accessed by users with or without pre-existing relationship, it is desirable to devise an appropriate way to use the same authorization engine to deal with users across untrusted boundaries.

Authorization in distributed transaction processing systems where strangers can engage in business transactions without pre-existing relationships requires a gradual trust building scheme, so that the parties can release their attributes incrementally, while receiving the other party's attributes in an automated and synchronised manner. In this way, the risk to which a party is exposed at any point in the negotiation can be minimised. To handle this aspect of security requires a trust establishment mechanism. Thus, the XACML Policy Decision Point (PDP) needs to allow the automated, gradual, selective release of policies and comparison with locally available credential attributes during the trust session.

3.0 The XACML Framework

The eXtensible access Control Markup Language (XACML) [3] is a general purpose policy language and framework that includes common datatypes, functions, and decision combining logic, and a query/response format, expressed in XML. The XACML standard uses a generic access control framework based on the IETF/DMTF model that allows an enterprise to specify and deploy an access control policy to match its access control requirements for a variety of resources. The request/response language describes the form of query and answer to flow between the Policy Enforcement Point (PEP) and Policy Decision Point (PDP) during the access control process. The wider acceptance of XACML results, apart from its rich capabilities, from the benefits of using a framework that can interoperate in open systems with minimal effort. Again, the XACML standard has defined profiles to integrate and interoperate with other security protocols and requirements such as SAML [27] and RBAC [28]. Interoperability is critical in distributed open systems and can be addressed by using and sharing common functionality within a standard framework.

Figure 2 shows a simplified view of the XACML authorization framework. The PEP is the mechanism which provides access to resources, and which forms access requests used to query the PDP. The Requests are represented to the PDP through an abstract entity called the Context Handler, which provides access to attributes from the Request as well as other sources called Policy Information Points (PIP). The PDP resolves an applicable policy from its Policy Administration Points (PAPs), evaluates the policy against the Context, and renders a Decision that is passed back to the PEP.

In this general approach, the PIPs are made to submit the requester's credentials unconditionally or else the service cannot be provided. A PIP has no way to verify that the service provider can be trusted with the requester's attributes. This provides only a unilateral access control scheme, which is not sufficient to protect the privacy of the

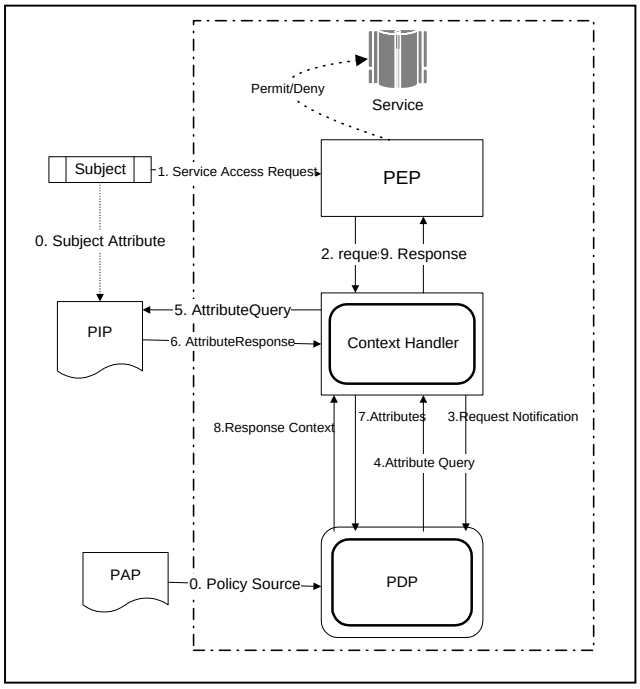

Figure 2 XACML Access Control Model

subject. Furthermore, such a one-sided authorization scheme cannot address some legal issues, since the service requester has no way of guaranteeing or proving that the service provider's privacy statements can be trusted. Thus, the requester's credentials are vulnerable to exploitation and abuse.

3.1 XACML Policy Language

The XACML policy language structure depicted in figure 3 comprises at the top level: *PolicySet*, *Policy Combining Algorithm*, *Target*, and *Policy*. The *PolicySet* is used to encapsulate a set of *Policies* and *PolicySets*. A *Policy Combining Algorithm* is logic that resolves a single decision out of multiple decisions (e.g., a single decision of Permit overrides any number of decisions to Deny). The *Target* defines simple applicability rules based on the *Subjects* (the attributes of the potential resource requesters), the *Resources* (attributes of the objects to which be access is

controlled), the *Actions* (the attributes of the action the subject intends to perform on the resources), and the Environments (any environmental or unclassifiable values). A *Policy* comprises at the top level two sub-policy components: *Target* and *Rule*. The *Rule* element comprises an Effect (Permit or Deny), a *Target*, and an optional *Condition* element, used to express the evaluation logic.

The XACML P*olicySet* can be considered as a tree, which contains one or more children: *PolicySet* or *Policy*. A policy in turn contains one or more child elements: *Rule*. In an authorization process, the *Target* at each level in the tree acts as a pre-condition for evaluating that part of the tree. If a *PolicySet Target* is false, then none of the child *Policies* or *PolicySets* are evaluated and the process returns "Not Applicable" for that branch of the tree.

When matching Resource values in a Target, the attributes of the *Resources* can match any of the *Resource Types* the underlying policy governs. In our case, the resource can be a credential attribute, a computing resource, or a policy. If the resource match is a policy, then the policy is considered sensitive and is not to be disclosed to arbitrary strangers. To allow disclosure of the policy, subjects must show sufficient credential attributes to satisfy the policy governing the protected policy.

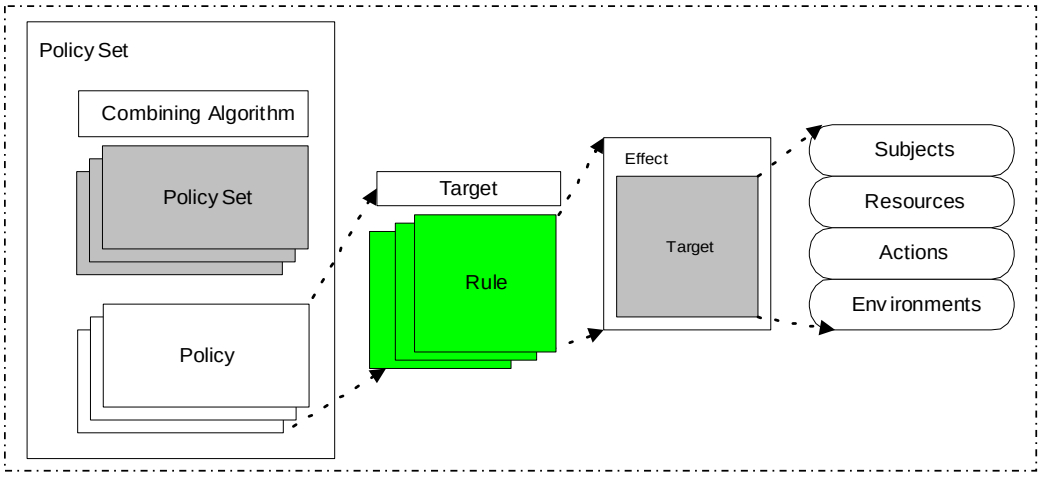

Figure 3 XACML Policy Language Structure

According to the XACML 2.0 specification, a PEP SHALL grant access to the protected resources only if a valid XACML Response Decision of "Permit" is returned by the PDP. Conversely, the default behaviour of a PEP is that it SHALL deny access to the protected resources in all other cases of XACML, including "Indeterminate". In XACML a *Request* SHALL be evaluated as "Indeterminate" if the PDP is "unable to evaluate the requested access". XACML interprets this behaviour in many ways, including cases such as missing attributes, network errors while retrieving policies, syntax errors in the decision request or in the policy, etc. In privacy aware environments, where entities are not sure whether the remote party can be trusted, they may not include their sensitive attributes initially in the service access Request(s). This can result in an "Indeterminate" response if the PIP is unable to supply them, which can be interpreted as an indicator to negotiate for the values. This behaviour can be leveraged to enable privacy and trust by controlling access to the attributes in the PIP.

4.0 XACML Trust Authorization Framework (XTAF)

Simultaneously, to protect the privacy of parties in a transaction and to control access to a service provider's resources, an architecture is required that can support trust and confidentiality at the same time. Access control techniques can be used to protect access to a party's credentials, but to establish trust requires a gradual and progressive approach in the exchange of a party's credentials. To enable trust and privacy, a bilateral process will empower both parties to use access control policies to determine the way their attributes are given to each other. This approach brings flexibility into the way privacy is protected, so that parties can explicitly specify who, how and when their credentials can be disclosed to others. Thus, with trust negotiation, the exchange of policies and credentials must be repeated several times as trust is progressively increased. But in order to know which credentials to release, a subject must be sent a policy of the resource. If the subject is happy with the policy, it will release further credentials. To address the privacy concerns of both parties, we adopt a model in which the client and server exchange sensitive credentials only after they know that they are talking to the right party. To enable fine-grained privacy control of sensitive information, we adopt selective and progressive exchange of policies and credentials, so that they can incrementally and sufficiently learn about each other. This will allow each party to determine what the other party proposes to do with their sensitive resources. Rather than taking all the risk of releasing sensitive attributes at once, parties are subject to smaller risk on an incremental basis and are able to withdraw at any point.

4.1 Trust Authorization Architecture

Figure 4 shows the basic building block of a Trust Authorization Service Handler (TASH), a component added to the core XACML model to address the aspects of privacy and trust in distributed authorization environments. This service is being implemented as a Trust Negotiation (TN) server and SunXACML Attribute Finder Module (AFM) [21] concept will interface it with core XACML engine. The Negotiation Protocol Module (NPM) handles the trust negotiation protocols and ordering of messages [12] during the building of a trust relationship. The Attribute Validation Engine (AVE) verifies and validates every credential attribute and policy that is received by the system before passing it to the trust decision engine. The Trust Information Handler (TIH) is responsible for the canonical representation of the inputs consumed by the TrustPDP and the outputs from it.

The TrustPDP handles trust access management decisions by comparing local policies with received credentials and received policies with local credentials. The local policies say what local resources (policies and credentials) can be unlocked by the received credentials. The received policies say what will become available by releasing further local

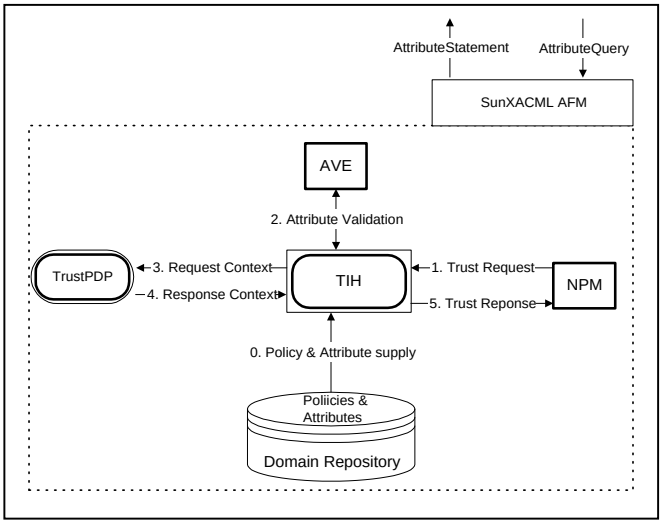

Figure 4 A Simplified TASH Architecture

credentials. The TrustPDP performs access management decisions in two ways:

- It checks if there are any local credentials (and policies) that can be disclosed by comparing the received credentials with the local policy. This is a necessary but not sufficient step for releasing further local credentials (and policies). It means that the remote party is sufficiently trusted to receive them.
- It checks the received policy to see if there is sufficient benefit to be gained from releasing further local credentials. When the recipient is a human user, he or she can be asked to make a decision. When the recipient is a service being accessed by a user, then there may be no received policy but it is still beneficial to the service to release further local credentials and policies.

Figure 5 depicts the high level architecture for how our proposed XTAF can perform trust negotiation in order to preserve the service requester's privacy and control access to the enterprise's services. Theoretically, as seen in figure 5, the TASH replaces the PIP in figure 2. In this way the TASH serves as a gatekeeper to the attribute store and filters all requests for authorization information. The two parties in the authorization course require a TASH at both endpoints to engage in a trust building session until trust is or fails to be established. The XACML standard specifies that the PDP shall request A*ttribute Values* from the ContextHandler. Then, the ContextHandler can query the PIP for the attributes. This provision makes it possible to layer TASH to work seamlessly with the access control engine in a manner that privacy and trust are enabled. What follows is a simplified illustration of how a client can preserve its privacy while requesting access to a service that requires its credentials. In step 1, Alice, sends a request for a particular service to Bob, and the request has missing or insufficient Subject credentials. The PDP attempts to evaluate the

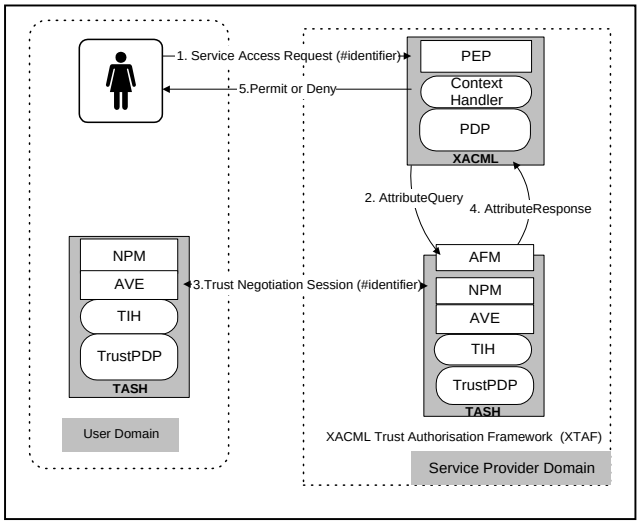

Figure 5 XTAF in Peer Mode

request, and during evaluation calls its ContextHandler to retrieve attributes, as shown by step2. The TASH uses the parameters passed by the ContextHandler to contact a similar TASH in Alice's domain or sends its TASH details via Alice's client to get a TN session credential, as depicted in step 4. Note that step 4 can take any number of rounds provided trust negotiation can advance through the exchanges of policies and credentials. If the session is successful, Bob's TASH returns a "TN successful" session credential with the attributes requested to Bob's ContextHndler which passes the attributes to his PDP, which will Permit or Deny access based on Alice's credentials.

4.2 XACML Trust Policy Set

We examine two ways in which the XACML policy language can be used to form an effective Trust Negotiation Policy Set. One approach is to use the existing *PolicySet* provision depicted in figure 6a, which can contain one or more *PolicySets* or *Policies*. We mentioned in section 3.1 that *PolicySet* is a tree, which can contain other trees. Each Target at any node of the tree is an intersection of Targets in the path that leads to that branch of the tree. During the policy evaluation, an ancestor *Target* is a pre-condition for evaluating a descendant *Target*. Thus, a evaluates to false, then none of the *Policies* are evaluated. Likewise, if the *Policy Target* evaluates to false, then none of the *Rules* in the *Policy* can be evaluated. Each *Policy* in the *PolicySet* has a *Target* element and may have *a* number of *Rule Targets*, n. So if we assume that policies *Id1, 2, ..n* are a Trust Policy Set, in a trust session, the *PolicySet Target* will be evaluated as a pre-condition, in addition to each *Policy Target* as another pre-condition, as well as the evaluation of each *Rule Target* in the *Policy*. Thus, the concept of *PolicySet*, in theory can be used to construct effective Trust Negotiation Policy sets. The *PolicySet* can be

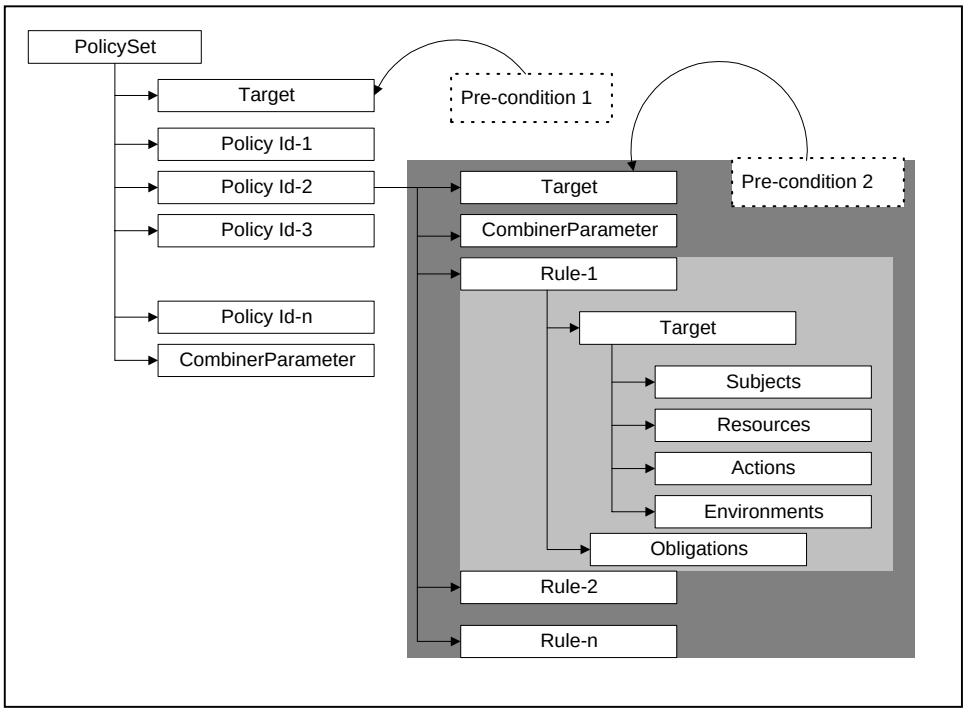

Figure 6a XACML PolicySet

PolicySet Target can be specified as a pre-condition for disclosing any part of the policies contained in that *PolicySet.* We give a simple example here.

Alice wants to access webserver1. Access to this service is defined by policy *p2*, which specifies that an accessor must be a postgraduate student in the computing department of the University of Salford. We assume that *p2* is considered sensitive, so that its disclosure is controlled by policy *p1*. *p1* specifies that to read policy *p2*, the requester must be a registered student of the University of Salford. Alice is unwilling to give up her postgraduate role certificate without determining whether the webserver can be trusted with her attributes.

We can implicitly place the requirement of *p1* in the *PolicySet Target* and the requirement of *p2* in a *Policy Rule* element. In this simple example, *p2* is disclosed only if p1 (in this case, the *PolicySet Target)* is evaluated to true. For instance, in figure 6a, *PolicySet* combines *Policies Id1, 2...n.* The *PolicySet Target* is a pre-condition for the evaluation of all the *Policies* or *PolicySets* contained in that *PolicySet. T*he *Policy Target* is a pre-condition for the evaluation of all the *Rules* in that *Policy*. If *PolicySet Target* used to define complex access control requirements that can support trust negotiation sessions.

Usually, in the trust negotiation domain, access control policies are arranged as directed policy graphs or trees [12]. It is apparent that the *Target* at any level is a pre-condition for evaluating that branch of the policy tree and for continuing to processing the other parts of the policy tree. This ensures that if the *Target* at any level evaluates to false, evaluating that branch of the tree becomes needless and negotiation can fail. The main drawback of this approach is that the structure maintains a strict hierarchy which may not be flexible enough to represent all conditions for trust building.

A second approach is to leverage the Rule *RuleId* attribute of the *Policy Rule* as shown in figure 6b. In this case, a *Rule* can be made to point to another *Rule* in order to protect that *Rule* from disclosure to arbitrary strangers. This model addresses the problems raised by the first approach, but may require the inclusion of a rule combining algorithm. However, additional constraints can be enforced in each rule by using the XACML Conditions and Obligations without extending the language. We adopt this approach as an

efficient way to construct a simple effective trust policy set. In some environments, the first approach can be more

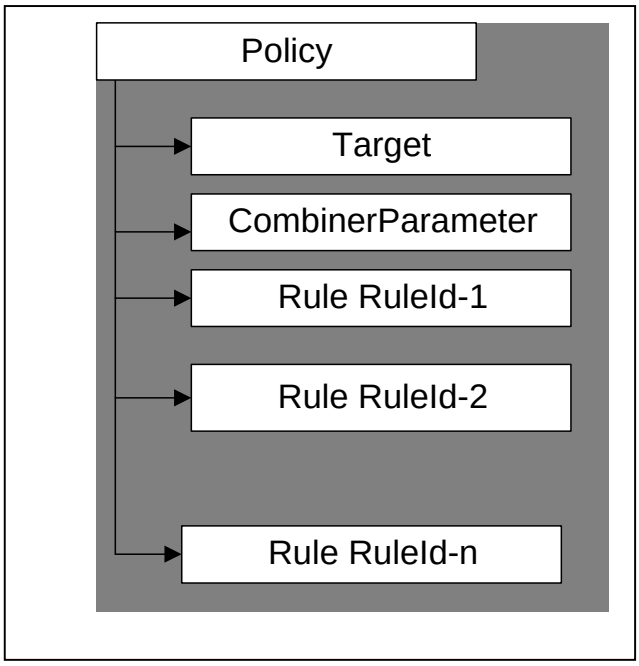

Figure 6b XACML Policy Rule elements

effective, especially where policies are defined by a hierarchy of authorities to protect authorization information flow.

At trust session runtime, the TASH builds a disclosure Trust Policy Set from applicable policies to form the Trust Policy Layers, from which it can infer the hierarchy of disclosure policies setting the source and sink nodes [14]. Once this is determined, the TASH can use the information deduced from the hierarchy to progressively negotiate trust with a remote party. The nodes satisfied during the negotiation phase are eliminated until the sink node (last layer) is satisfied or the session terminates. The ordering and sequence of the messages are important for the building of trust and confidentiality. This determines the trust level and what each party is ready to give in exchange of his own information.

5.0 Example Scenario

In this example we show how a CIA agent can surreptitiously gain access to a CIA web service that is hidden behind a publicly accessible service. Alice, the CIA agent, asks for http://www.cia.gv/training/ but is unwilling to give up her CIA X509 Attribute certificate until she is confident that she is communicating with a CIA server. But the server wishes to protect the disclosure of this access requirement to arbitrary strangers, redirects Alice to satisfy other requirements, and negotiation starts.

We describe the relevant Trust Policy using *Policy Rule containers* to define access requirements such that sensitive rules are protected from arbitrary disclosure, as well as the control of unauthorized access to the web services. What follows is the description of the web service Policy in plain English (figure 7a is a fragment of the policy).1) *Rule 4* states that a requester (Subject) with attribute *CIA X509 certificate* can read (Action) the training

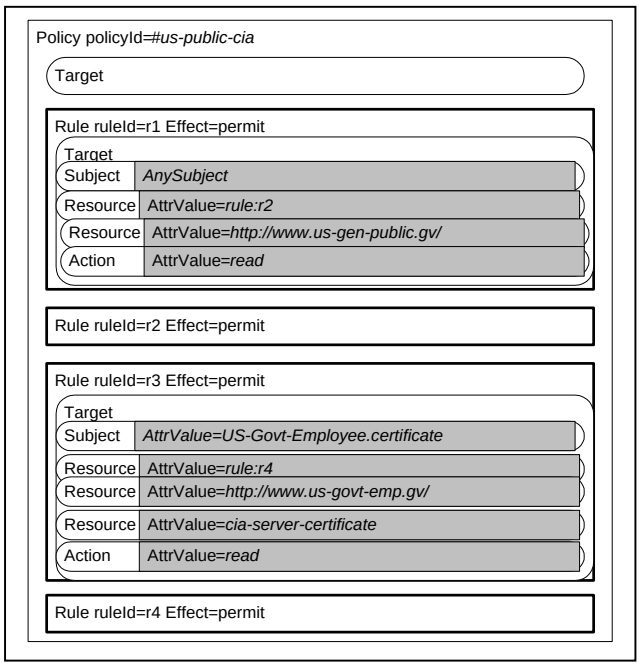

Figure 7a Fragment of Our CIA server Policy

documents (Resource). (Details not shown.). 2) *Rule 3* says that a requester with attribute *US government employee certificate* can be given access to: the *access requirement for Rule 4; the CIA server certificate;* and the resource http*://www.us.govt.gv/*. 3) *Rule 2* states that a requester with the attribute *US citizenship certificate* can see the *access requirement for Rule 3* and the *US government server certificate*, as well as access the resource *http://www.us.public.gv/*. 4) *Rule 1* says anyone can access *http://www.us-gen-public.gv/* and see the access requirements of *Rule 2*.

Conversely, Alice's Attribute Release Policies (ARPs) define the requirements for accessing her digital attribute certificates as follows (see Figure 7b): 1) *Rule 4* states that only *CIA certified servers* can be permitted to see her *CIA agent certificate*. 2) *Rule 3* says that the *access requirement for Rule 4* can only be seen by US government certified servers. 3) *Rule 2* states that only *certified US public servers* can see her *government employment certificate* and view the *access requirement for Rule 3*. 4) *Rule1* states that any requester can be shown the *access requirement for Rule 2.* From this example, it can be seen that the rules can form a chain defining the order in which they must be disclosed and evaluated in a progressive manner.

At any particular stage in the negotiation, each party must release information to allow the negotiation to move to the next stage. For example, in order to see the access requirement for the *CIA training resources* Alice must prove that she is a *US government employee*. Proving this will also allow Alice to see the server's CIA server

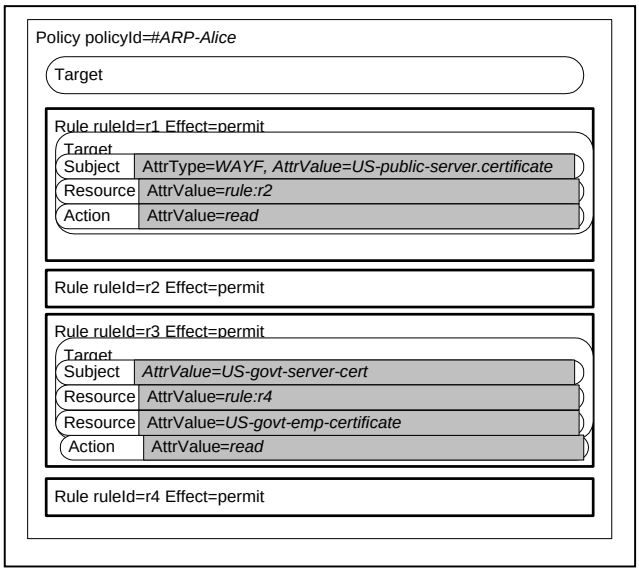

Figure 7b Fragment of Alice's ARP

certificate, which satisfies her ARP regarding release of her CIA Agent Certificate. In this way, the negotiation proceeds in increments until Alice finally gains access to the training resource.

In principle, the operation of the native XACML authorization engine is synchronized with its TN server in such a way that the engine's information is visible to the TN server. Figure 8 illustrates how the negotiation is performed iteratively to build trust, protect privacy and gain access to the training service.

5.1 Discussions

Our example shows how hierarchical resources can be protected and how, at the same time, building of trust may be enabled to adequately protect the privacy of authorization information. The secret agent requested the training resources, but the server redirected the request to initiate a trust building session. Theoretically, a native XACML PDP asks its ContextHandler for the attributes which were not submitted in the initial request context. In our case, the ContextHandler contacts the TN server and a trust session can start. However, this approach requires that negotiators must possess other credentials that can be used incrementally to satisfy successive access control requirements. Arguably, the incremental attribute release technique can also be used to hinder replay attacks [17]. In this sense, for an adversary to launch replay attacks, he must cache all the credential sets in order to succeed.

Figure 8 presents the naïve trust negotiation session. Alice is always seen issuing her privacy policies and credentials that are unlocked by the credentials provided by the server. Alice always requires the server to provide credentials before she gives hers. Similarly the server releases policies and credentials that are unlocked by Alice's credentials. The difference is that the server must have a credential it is willing to release to the public before TN can start. What is important is that the parties must co-operate enough for trust to succeed whenever possible or fail gracefully. The decision on whether to release credentials or disclose policy depends on the access control policies and the relationships between them. In [15] it was mentioned that policy disclosures are vulnerable to probing attacks. As a result an adversary can use policy disclosure techniques to learn of a party's possession or non-possession of the information being asked for. It is also possible for an attacker to lie by expressing constraints on credentials or services that (s)he does not possess in order to gather information from the attacked. In our example scenario, we have shown how these pitfalls are addressed by a finer trust policy layering. The example illustrates how Alice – a CIA agent - can surreptitiously gain access to CIA online training resources. Alice is not prepared to push her *CIA.agent.certificate* to the server and the server is not ready to disclose the policy that governs the training resources and *CIA.server.certificate* to arbitrary strangers. Thus, Alice's request to http://www.cia.gv/training/ is redirected to a TN session. From this example, the constraints in the first round of policy and credential disclosures on both sides can be reduced to:

Subject.AttributeValue ◄——— WAYF.certificate

It is apparent therefore that the first round of trust negotiation which asked for the Where Are You From certificates are not tightly coupled to any of the sensitive resources: *http://www.cia.gv/training/*, *CIA.server.certificate* and Alice's *CIA.agent.certificate*. The assumption is that the kick-off policies cannot explicitly reveal whether both parties possess the required credentials or services. This suggests that both Alice's and the server's behaviour cannot reveal non-possession or possession at the first round of iteration. Again, if the server gives out *WAYF.certificate* and Alice fails to respond with credentials that can satisfy the server's disclosure policy, the negotiation can fail at this point. What is important is that whenever trust is to succeed, it is desirable that policy and credential flow should advance the level of trust, which minimizes the effect of probing attack or lying under false policy expression. The major drawback here is that negotiators are required to possess a set of credentials (but this natural) in order that access control policies can be used to address the order in which those credentials can be released to advance the trust building session.

6 Related Works

Seamons et el [15] [16] [13] [14] and Bertino et el [6] [8] have done useful works in the area of Trust Negotiation and Management. Their work provides good theoretical background on the concepts of trust with quite a number of implementation scenarios. Seamons et el had advanced the notion of trust negotiation protocol and strategy with some practical demonstration of how they can be implemented [12]. In the area of trust policy and language, Bertino et el have proposed a number of ways to encode policies and credentials [6]. Their various works on privacy during trust negotiation have had considerable influence on our work [15]. Their various works are

proprietary and cannot interoperate; this is where the XACML framework, an open standard, is gaining prominence with great potentials.

Lorch et el presented their first experience using XCAML in distributed systems. Their work included the analysis of the performance of XACML with existing models and highlighted its limitations. They drew the

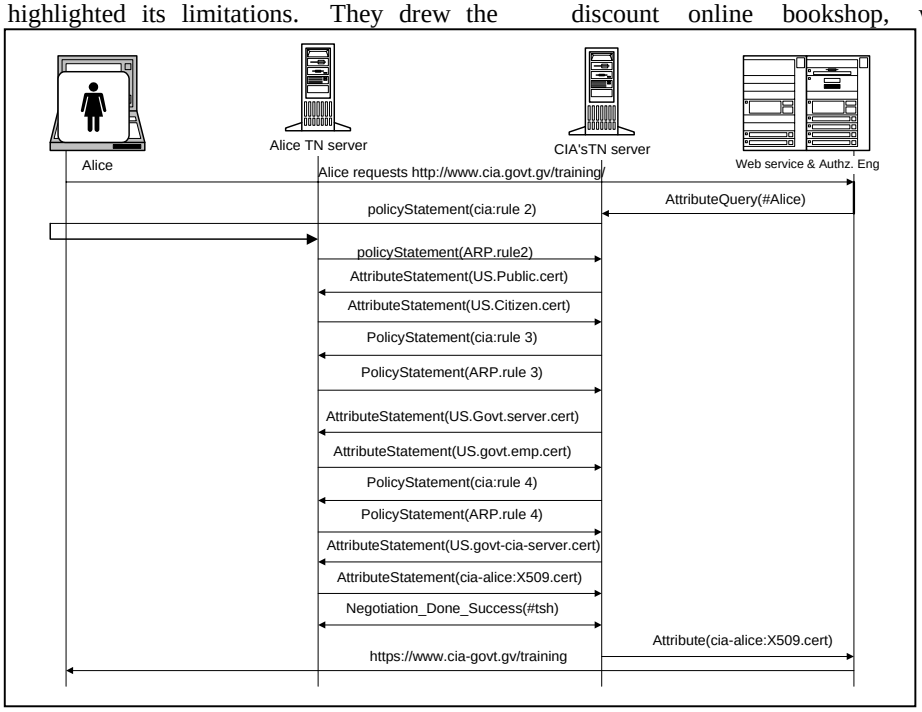

Figure 8 A Trust Negotiation Process of the Example Scenario

experienc gained in the integration of SAML and XACML in distributed open systems and performance results. Lorch et el presented how the PRIMA model [19] leveraged XACML to express policy requirements. PRIMA is specifically designed for access control in grid computing environments; users can assign and/or delegate privileges to each other without involving policy administrators. However, Lorch et el focused mainly on the analysis of XACML's performance and did not address the privacy issues and trustworthiness in distributed environments. Our work is among the first to look into how XACML can be used to build trust relationship in distributed authorization environments. This is a significant direction since XACML is a generic access control model that has continued to address wider access control requirements.

The Shibboleth infrastructure, in an attempt to address privacy in an authorization environment, proposed two kinds of policies: Attribute Release Policy (ARP) and Attribute Acceptance Policy (AAP) [20]. Shibboleth is a distributed authentication and authorization architecture whose access control is based on users' attributes. Shibboleth provides an Attribute Release Policy (ARP) on the users' home site and an Attribute Acceptance Policy (AAP) on the resource target site to protect users' privacy. Although Shibboleth provided a means by which both users and attribute authorities [20] could express their privacy preferences at the home site, all users coming from different sites visiting one target resource site may have the same AAP of that target site. Privacy in Shibboleth is primarily focused on using pseudonymity; however the use of pseudonymity does not completely protect privacy in an environment where the user may give other attributes in the cause of using the authorised resources. For instance, an institution may give a student a signed assertion to access a discount online bookshop, which is the required authorization token. But if the student wants to purchase a book, (s)he needs to provide other personal attributes such as credit card number, physical address for payment and delivery. Shibboleth's provisions for privacy still fall under a one-shot process: the parties in transaction cannot determine if a party can be trusted with sensitive attributes. Lorch et el [4] pointed out the current effort being made at Sun Microsystems and Brown University to integrate Shibboleth with XACML. This work used WSPL, a profile of XACML that supports policy intersection to determine if two policies are mutually amenable. This approach requires some policy on the server side be released without negotiation, but then provides a very simple means for calculating what the client is willing to share.

PERMIS [11] [22] is a middleware authorization framework which focuses mainly on the RBAC access control model. PERMIS has successfully been implemented in a number of application scenarios with interesting results [23] [24] [22]. It fully supports role hierarchy and its policy language is user friendly. It has a proprietary GUI policy editing tool [25] and Privilege Allocation (PA) subsystems for managing roles and permissions. The PERMIS language is limited in expressions and semantics compared to XACML which is very expressive with significant functionality. The PERMIS framework does not provide direct support for bilateral exchange of policies and credentials to address privacy issue and trustworthiness in a manner presented in this paper. The concepts presented in

this paper could be implemented using the PERMIS authorization model. PERMIS has in its architecture a subsystem that signs, verifies and validates X.509 attribute certificates used to represent authorization credentials in PERMIS model. However, to use our framework in PERMIS requires the introduction of a trust layer and the inclusion of identifiers in the Target Access Policy (TAP) element, so that a TAP can protect another TAP to form effective trust policy set.

7. Conclusion and Future Work

This paper proposes a way that the widely accepted XACML standard can address three essential security components: trust, privacy and service control in a synchronised manner. It identifies some of the necessary additions to core XAML model that allow for progressive bilateral exchange of policies and credentials between two parties in a way that privacy and trust are sufficiently preserved while protecting access to sensitive services. We have leveraged trust concepts already proposed by researchers and demonstrated how our model can protect hierarchical resources at the same time.

In particular, we examined the various ways that the XACML language can be used in trust negotiation and proposed how to construct effective trust policy sets which can optimize trust establishment sessions. We have proposed a trust layer in the primitive XACML model for gradual building of trust relationships, so that multiple parties can engage in secure, trusted transactions. The trust layer includes NPM that implements trust protocols and controls the way messages are exchanged, and the TrustPDP which handles trust management decisions. Additionally, we have introduced the trust session handle (*tsh)*, an optional parameter included in the AttributeStatement which controls the way a TN server is invoked by the native XACML authorization engine. The concepts discussed in this paper are being implemented using the SunXACML implementation [21] and the PERMIS Attribute Verifier subsystem.

Our approach has the capabilities to protect complex hierarchical resources and policies, which will guarantee a finer control of sensitive services. It provides the flexibility to leverage a common framework for all the authorization needs and lessen the burden in administrative requirements. Interoperability is promoted among strangers without trust relationships whilst sharing sensitive business information. In our approach, we have a loosely coupled architecture which provides flexibility in the way trust, privacy, and services are protected synchronously. In this way, if the client chooses to store credentials locally, a light-weight Trust Establishment Engine can be deployed without losing most of the functionality of an authorization mechanism.

Though policy disclosure during trust negotiation is vulnerable to probing attacks, we have demonstrated how our progressive policy and credential disclosures can minimize this threat. That is, the progressive and incremental paradigm allows the risk to be managed such that exposure to risk at any particular point is limited. Again, since the Trust level X determines what each party can disclose, it is arguable that not much sensitive information is lost during establishment of trust using our framework. Though Seamons et el proposed the use of a dynamic policy graph through policy transforming agents to address this problem, which is similar to a static finer granularity in policy expressiveness, we did not consider that adding additional computational overhead is a better way to address the problem. The system presented in this paper can be adapted to an existing suite of negotiation strategy to allow participants the option of selecting how fast they can establish trust.

Presently, we are looking into how to include SAML interfaces to handle the trust message contexts [27]. The SAML schema provides information to identify and validate the content of assertions such as needed in the trust negotiation sessions. Already, XACML version 2 has a SAML 2.0 profile which describes its interface with XACML. We are also looking into how to address the need-to-know principle, so that only the relevant credentials that contribute towards the client's stated goals are exchanged between the client and the server. Our goal is to find more practical approaches in the way policies and credentials are exchanged without too much computational overhead.

8. References

[1] A.Anderson, "Privacy Policy Languages: XACML vs EPAL," presented at 5th Annual Privacy & Security Workshop, 2004.

[2] K.E.Seamons, M.Winslett, and T.Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," presented at Network and Distributed System Security Symposium, San Diego, CA, Feb 2001.

[3] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0," http://www.oasis.org, Feb 2005.

[4] M.Lorch, S.Proctor, R.Lepro, D.Kafura, and S.Shah, "First Experience Using XACML for Access Control in Distributed Systems," presented at ACM Workshop on XML Security, Fairfax Va US, 2003.

[5] W.H.Winsborough, K.E.Seamons, and V.E.Jones, "Negotiating Disclosure of Sensitive Credentials," presented at 2nd Conference on Security in Communication Networks, Amlfi, Italy, Sept 1999.

[6] E. F. E.Bertino, A Squicciarini, "TNL: An XML-based Language for Trust Negotiations," presented at IEEE 4th International Workshop on policies for Distributed Systems and Networks, Lake Como Italy, 2003.

[7] A.Acquisti, "Privacy and Security of Personal Information-Economics Incentives and Technological Solutions," presented at Workshop on Economics and Information Security, University of California Berkeley, 2002.

[8] E. Bertino, E.Ferrari, and A. Squicciarini, "Trust Negotiations: Concepts, Systems and Languages," IEEE Computer, pp. 27-34, July/August 2004.

[9] M.Winslett, "An Introduction to Automated Trust Establishment," presented at 1st International Conference on Trust Management, Crete, Greece, May 2003.

[10] K. E. Seamons, M.Winslett, T. Yu, L.Yu, and R.Jarvis, "Protecting Privacy during On-line Trust Negotiation," presented at 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA, April 2002.

[11] D.W.Chadwick, "The X.509 Privilege Management Infrastructure," presented at Proceedings of the NATO

Advanced Networking Workshop on Advanced Security Technologies in Networking, Bled, Slovenia, 2003.
[12] J.Holt and K.E.Seamons, "Interoperable Strategies in Automated Trust Negotiation," presented at 8th ACM Conference on Computer and Communications Security, Philadelphia Pennsylvania, Nov 2001.
[13] W. Winsborough, K. Seamons, and V. Jones, "Negotiating Disclosure of Sensitive Credentials," presented at Second Conference on security in Communication Networks, Amalfi, Italy, September 1999.
[14] T.Barlow, A.Hess, and K.E.Seamons, "Trust Negotiation in Electronic Markets," presented at Eighth Research Symposium in Emerging Electronic Markets, Maastrict Netherlands, Sept 2001.
[15] A.J. Lee, "Traust: A Trust Negotiation Based Authorization Service For Open Systems" Master Thesis, Cornell University, 2003
[16] K.E.Seamons, M.Winslett, T.Yu, B.Smith, E.Child, J.Jacobson, H.Mils, and L.Yu, "Requirements for Policy Languages for Trust Negotiation," presented at 3rd International Workshop on Policies for Distributed Systems and Networks, Moneterey, CA, June 2002.
[17] B.Schneier, Secrets and Lies-Digital Security in a Networked World, 2 ed: Wiley Publishing, Inc, 2004.
[18] W. Hommel, "Using XACML for Privacy Control in SAML-Based Identity Federations." presented at "IFIP International Federation for Information Processing CMS 2005 LNCS 3677 pp. 160-169, 2005
[19] M. Lorch and D.Kafura, "Supporting Secure Adhoc User Collaboration in Grid Environments," presented at 3rd Int. Workshop on Grid Computing, Baltimore USA, Nov. 2002.
[20] S. Nazareth and S. Smith, "Using SPKI/SDSI for Distributed Maintenance of Attribute Release Policies in Shibooleth," Computer Technical Report TR2004-485, 2004.
[21] S. Proctor, "Sun's XACML implementation APIs" http://sunxacml.sourceforge.net/
[22] D.W.Chadwick and O.Otenko, "Implementing Role Based Access Controls Using X.509 Attribute Certificates," IEEE Internet Computing, pp. 62-69, 2003.
[23] D.W.Chadwick and D.P.Mundy, "The Secure Electronic Transfer of Prescriptions," presented at HC2004, Harrogate, UK, March 2004.
[24] D.W.Chadwick and D.P.Mundy, "Policy Based Electronic Transmission of Prescriptions," presented at IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Como Italy, 2003.
[25] S. Brostoff, M. A. Sassea, D. Chadwick, J. Cunningham, U. Mbanaso, and O. Otenko, "RBAC what? Development of a role-based access control policy writing tool for e-Scientists," presented at Workshop on Grid Security Practice and Experience, Oxford UK, 2004.
[26] OASIS, "Security Assertion Markup Language (SAML) Version 2.0," http://www.oasis.org, Feb 2005.
[27] OASIS "SAML 2.0 Profile of XACML v2.0, http://www.oasis.org/ Nov 2005
[28] Anne Anderson," Core and hierarchical role based access control (RBAC) profile of XACML v2.0, OASIS, http://www.oasis-open.org/, Feb 2005