A POLICY FRAMEWORK FOR ACCESS MANAGEMENT IN FEDERATED INFORMATION SHARING

Rafae Bhatti[1], Elisa Bertino[2], Arif Ghafoor[1]
[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907;[2]Department of Computer Sciences and CERIAS, Purdue University, West Lafayette, IN 47907

Abstract: Current mechanisms for distributed access management are limited in their capabilities to provide federated information sharing while ensuring adequate levels of resource protection. This work presents a policy-based framework designed to address these limitations for access management in federated systems. In particular, it supports: (i) decentralized administration while preserving local autonomy, (ii) fine-grained access control while avoiding rule-explosion in the policy,(iii) credential federation through the use of interoperable protocols, with support for single sign on for federated users, (iv) specification and enforcement of semantic and contextual constraints to support integrity requirements and contractual obligations, and (v) usage control in resource provisioning through effective session management. The paper highlights the significance of our policy-based approach in comparison with related mechanisms. It also presents a system architecture of our implementation prototype.

Key words: Federated Systems, Policy-based Management, XML Access Control

1. INTRODUCTION

Federated systems comprise of shared resources belonging to distributed, potentially mutually untrusted, administrative domains. A key property of federated systems is that each participating site retains local autonomy (i.e. administrative control over its resources), which is a main difference between federated and traditional distributed system concepts. Many commercial and government

organizations are increasingly adopting the federated approach to online information management, be it for critical infrastructure protection such as the DoD NetCentric Directive [1] or wide dissemination of scholarly work such as the Federated Digital Library initiative [2].

Access management in a federated system includes specification and administration of access control policies of protected information resources belonging to participating sites, and secure federation to allow seamless sharing of those resources. An effective mechanism for access management in such systems must take into consideration the access control requirements as stated in the access control policies of each participating site. However, several challenges arise in developing and enforcing access control policies in a federated paradigm.

The principle of local autonomy impacts the ability of the federation to share and acquire resources [3]. A major problem in this context is policy administration. A centralized administration approach may imply loss of autonomy for participating sites [3], and is ruled out. On the other hand, decentralizing administrative control requires that participating sites specify authorization policies for federated users [3]. This approach preserves local autonomy, but is complicated by the fact that federated systems typically involve a diverse, unseen user pool requiring granular and differentiated access to a diverse set of resources located anywhere across the federation. It, therefore, precludes the use of traditional approaches to distributed authorization (such as X.509 based PKI) that assume knowledge of user identities and resource locations. Even when knowledge of identities is available, the requirement of fine-grained access control would lead to rule-explosion in the access control policy given the size of federated population in open systems. To keep the rule set from becoming prohibitively large calls for a scalable approach.

While decentralizing administrative control requires that participating sites specify authorization policies for federated users in an appropriate format, a related challenge is to transfer the credentials of the federated users across administrative boundaries for them to obtain federated resources according to the applicable authorization policies. We refer to such a mechanism as credential federation. No federated system can achieve its access management goals unless the requirement for credential federation is satisfied. Doing so, however, requires interoperable protocols that can allow participating sites to federate user credentials. Multiple policies may be necessary to evaluate the request of a federated user, which requires the support for combining rules from multiple policies to support composite policy evaluation. A related requirement for credential federation is that of achieving Single Sign On (SSO), which enables persistent authorization support for federated users within a single login session.

The "dual" of credential federation is resource federation, i.e. availability of resources to federated users according to the applicable authorization policies.

Resource federation can occur in two modes, namely resource sharing and resource provisioning. Provisioning may be considered an advanced form of sharing where the resource is actually acquired (rather than just accessed) by the requestor for a specified period of time. It is assumed that the resource will remain within the immediate control of the owner during this time.

While credential federation is aimed at securing the authorization information of federated users, resource federation takes a more usage-oriented view, and is aimed at ensuring effective protection of accessed or acquired resources. For instance, a digital document acquired in a read-only mode by an authorized user for a specified period of time must be protected against any (unauthorized) modifications. In other words, the role of access control should not end after the resource is initially provided, but must persist for the duration of the provisioning session. Traditional access control models do not take a usage-oriented view, and hence are inadequate to capture the protection requirements associated with federated resource sharing.

Lastly, the collaborative nature of a federated system requires the specification of semantic and contextual constraints to ensure adequate protection of federated resources. Semantic constraints include high level integrity principles that need to be captured in the access control policy, such as Separation of Duty (SoD) [4]. For instance, it may be required that no user may acquire the rights to access two design documents from two competing firms. Contextual constraints include temporal or other environmental attributes surrounding an access request that must be evaluated to decide on resource provisioning. For instance, a resource access between two domains may be time-constrained to occur only during business hours. Conditions associated with provisioning and de-provisioning of resources are absolutely critical to the functioning of the federation, especially when resources are provided against some form of obligation (such as service-level agreements, etc.).

Supporting semantic and contextual constraints in the access control policy requires mechanisms for constraint specification, evaluation and enforcement in a decentralized manner. While constraints increase the expressiveness of the policy, enforcing them requires maintaining state information across all user accesses, and is much more complex in a decentralized environment than in a centralized architecture. Reducing the complexity of policy administration, therefore, becomes an immediate concern [19]. Moreover, the integrity requirements and contractual obligations within a federation might change on-demand, and an access management mechanism must be flexible enough to facilitate such adaptation.

All the above cited challenges are unique to the federated paradigm and need to be addressed to ensure effective access management. We believe that a policy-based approach to access management provides a viable solution since it is flexible and adaptable enough to meet these requirements. A key benefit of policies for systems management is that policies are interpreted rather than compiled into

program code, so can be changed dynamically without changes to application code [20]. To realize this benefit, however, it is mandated that the policy supports an interoperable and expressive specification that can support these access management needs of a federated system.

1.1 Contributions and organization

The primary objective of this paper is to study the impact of these outlined challenges on the design and administration of an access control policy. In response, we present the design and enforcement architecture of a policy-based framework that addresses them. Our design builds upon the well-known Role Based Access Control (RBAC) model which has been recognized for simplified administration [5] particularly in the context of federated access management [6], and augments it with necessary extensions to support access management in a federated system.

In particular, we support the following key extensions to basic RBAC model:

(i) Delegated administration through the use of trust relationships captured through role hierarchies. Our approach provides scalable decentralization support and preserves local autonomy.

(ii) Credential specification for an unseen, heterogeneous pool of users and resources through a combination of rule-based role assignment and role-based authorization. Our approach allows fine-grained access control while avoiding rule-explosion in the policy.

(iii) Credential federation through the use of interoperable protocols. We support combing rules from multiple policies for composite policy evaluation, and also provide single sign on for federated users.

(iv) Usage control in resource provisioning by employing usage-oriented resource protection policies, and session management mechanism.

(v) Specification and enforcement of semantic and contextual constraints needed to support integrity requirements and contractual obligations in a federated system. Our approach achieves scalability and flexibility through modularized constraint specification and maintains reduced complexity through lazy rule instantiation.

The remaining of the paper is organized as follows. Section 2 introduces design principles of our policy-based approach for access management in federated systems. Section 3 presents the details of the policy framework. Section 4 presents the system architecture of an implementation prototype of our proposed framework. We apply our policy framework in a federated digital library environment (with read-only access), and illustrate design and enforcement of access control policies for secure federation of XML-based digital documents. Section 5 puts our work in perspective with related work, and highlights the particular merits of our work with respect to the outlined challenges. Section 6 concludes the paper.

2. DESIGN APPROACH

Recently, there has been a growing recognition of security problems in federated environments, and several emerging specifications in various stages of standardization have emerged [7-10]. But standards alone won't solve the problem. The answer lies in combining standards with policies that govern how shared information can be used [11]. In this paper, we provide a policy-based solution specific to access management in federated systems with the motivation to address this crucial requirement. Among our design goals is to provide an interoperable specification for expressing access control policies that is compatible with emerging security standards for information federation.

All notable emerging standards for Web-based federation are XML-based; we therefore use an XML-based policy specification language. As a consequence, our policy-based framework facilitates interoperability with complementary security protocols for federated systems. In addition to supporting high-level access control requirements, our language can be used to encode various low-level security policies (such as IPSec) through appropriate XSLT tools, and allow them to be applied in a federated system.

Another design goal is to allow modular specification of authentication and authorization credentials to provide support for pluggable authentication standards to be incorporated. Being neutral to the authentication mechanism, we do not deal with the authentication system needed to generate the authenticator. In other words, we assume that the authentication information supplied to the system is already verified, and is encoded in an authenticating credential usable in our framework.

Our design is focused on specification of policies, and therefore we do not consider certain other auxiliary issues. For example, we do not deal with credential collection and provisioning issues, which include deployment and/or discovery of credentials across multiple applications, typically through the use of directory services (such as LDAP). We also do not deal with identity aggregation issues involving multiple LDAP repositories for manipulating composite credentials. Additionally, we assume that the channels used for network communication are secured by appropriate mechanisms (such as SSL/TLS).

3. X-GTRBAC POLICY FRAMEWORK

This section describes the key features of X-GTRBAC (XML-based Generalized Temporal Role Based Access Control), our XML-based policy specification framework. Our specification language is an extension of the RBAC model suitable for addressing the access management challenges in federated systems discussed in this paper.

3.1 Language Specification

X-GTRBAC language specification is captured through a context-free grammar called X-Grammar, which follows the same notion of terminals and non-terminals as in BNF, but supports the tagging notation of XML which also allows expressing attributes within element tags. The use of attributes helps maintain compatibility with XML schema syntax, which serves as the type definition model for our language. Since it follows BNF convention, X-Grammar can be accepted by a well-defined automaton to allow automatic translation into XML schema documents. This allows automatic creation of strongly typed policy schemas based on the supplied grammar specification. We choose to use X-Grammar syntax instead of directly working with XML schemas for ease of analysis (since existing compiler tools for BNF grammars can be applied) and better readability and presentation. Examples of X-Grammar policies are given in following sections. The complete syntax of X-GTRBAC language specification appears in Appendix A.

3.2 Policy Components

We now describe the main components of our policy language. While doing so, we motivate our design decision by evaluating existing approaches against our stated requirements, and pointing out the merits of our design with respect to our objectives.

3.2.1 Credentials

Credentials are a key component of an access control language. A credential encodes the authentication and authorization information for the users. We have earlier motivated that a heterogeneous and unfamiliar user and resource pool in a federated system complicates credential specification, since it precludes the use of traditional approaches to distributed authorization (such as X.509 based PKI) that assume knowledge of user identities and resource locations.

[12, 13] are well-known examples of distributed schemes that have used identity-based X.509 certificates for user authentication. The authentication information (i.e. public keys) is then used to construct an authorization credential that comprises of a set of resource-specific rules. The credentials are bound to user identities and therefore this approach to credential specification is not scalable. Even when knowledge of identities is available, the requirement of fine-grained access control would lead to rule-explosion in the access control policy given the size of federated population in open systems. Additionally, this approach tightly couples

authentication with authorization, and is therefore inflexible, and violates one of our design principles.

Our policy framework addresses this problem through the use of attribute-based (as opposed to identity-based) credential specification. We adopt a modular approach and allow independent specification of credentials used in authentication and authorization. The authenticating credential comprises of authentication information expressed in terms of user attributes which are used by the access control processor for role assignment. This idea is similar to the one used in [14]. However, unlike in [14], we do not require reliance on X.509 identity-based certificates to encode user authentication information. Instead, the user attributes may be supplied in any mutually agreed format, such as an Attribute Statement in the emerging identity federation standard SAML [7]. This supports the requirement for credential federation (See Section 3.3.3).

An authorization credential comprises of information about role attributes, role hierarchy, and role constraints. Examples of role attributes are time of day, system load, etc. [16]. They are used by the access control processor for controlling assignment of users and permissions to roles. Role hierarchy provides a means of privilege inheritance, and hierarchical role definitions can be applied to extend or specialize existing policies. Role constraints restrict the enabling, activation, and delegation (See Section 3.3.1) of roles to allow fine-grained access management.

The authenticating and authorization credentials used in our framework are included in an XML User Sheet (XUS) and an XML Role Sheet (XRS) respectively. The top-level X-Grammar syntax of XUS and XRS is shown in Figures 1 and 2.

```
<!-- XML User Sheet>                ::=
<XUS [xus_id = (id) ] >
 <CredType cred_type_id=(id) type_name= (name) >
   [<!—Header>]
   <!-- Credential Expression>
 </CredType>
</XUS>
```

Figure 1. Top-level X-Grammar for XML User Sheet: Includes definition of authenticating credential

```
<!-- XML Role Sheet>                ::=
<XRS [xrs_id = (id) ] >
 <Role role_id = (id)
  role_name = (role name)>
  [<!-- Cred Type>]
  [<Junior> (name) </Junior>]
  [<Senior> (name) </Senior>]
  [<!—(En|Dis)abling Constraint>]
  [<!—[De]Activation Constraint>]
  [<!—Delegation Constraint>]
  (<SSDRoleSetID> (id) </SSDRoleSetID>)*
  (<DSDRoleSetID> (id) </DSDRoleSetID>)3*
 </Role>
</XRS>
```

Figure 2. Top-level X-Gramamr for XML Role Sheet: Includes definition of authorization credential

The credential specification in our framework facilitates a combination of rule-based role assignment and role-based authorization (See Section 3.2.3). Our approach allows fine-grained access control while avoiding rule-explosion in the policy since users are assigned to roles and access rules are specified at per-role rather than per-user level.

3.2.2 Constraints

Constraints are essential to the expressiveness of an access control language. Specification of semantic and contextual constraints is vital to support the enforcement of integrity principles and resource provisioning contracts in a federated system. As motivated earlier, enforcing expressive constraints in a decentralized manner involves maintaining prohibitive amounts of state information and introduces significant complexity. Additionally, adapting the constraints according to on-demand changes in

[3] SSD and DSD refer to static and dynamic SoD respectively.

integrity requirements and contractual obligations within a federation requires a specification format that facilitates such adaptation.

Most well-known distributed authorization schemes [12-15] do not cover the requirements of constraint specification and enforcement as required for access management in federated systems. As mentioned before, [12, 13] tightly couple resource-specific authorization constraints with the identity-information. This method of constraint specification is clearly inflexible to allow on-demand adaptation of constraints; doing so would require issuance of a new credential for the affected users since their identity is bound to the authorization.

Additionally, constraints in [12, 13] are inadequate to capture semantic integrity constraints, such as SoD, in a federated system since doing so at the user level would require prohibitive amount of state information to be maintained. In comparison, enforcing SoD at the granularity of role is more manageable and one has to include in the constraint definition only the roles, as opposed to all permissions, that the user may have access to. The support for contextual constraints based on temporal or other environmental attributes is also limited in [12, 13], since they do not have a formal temporal model, and rely on underlying operating system primitives to enforce temporal constraints. [14, 15] are based on basic RBAC and do not support specification of contextual constraints.

X-GTRBAC supports a variety of constraint categories to adequately capture the access management requirements in federated systems. The constraint specification in X-GTRBAC framework is primarily based on Generalized Temporal Role Based Access Control (GTRBAC) model [17]. GTRBAC is a mechanism using temporal logic to express a diverse set of fine-grained temporal constraints in an RBAC environment. The temporal constraint categories supported by GTRBAC include periodicity, interval, and duration constraints which can be used to constrain the period, interval and duration, respectively, of user-to-role and permission-to-role assignments. Another category is that of trigger-based constraints, which can be thought of as condition-action rules. As the name implies, trigger-based constraints are used to condition the occurrence of an event on another. Moreover, GTRBAC also elegantly captures the SoD constraint among roles to capture integrity requirements. Both static and dynamic SoD constraints are supported. Capturing these constraints at the role level helps reduce state information needed to enforce the constraints.

X-GTRBAC supports modular specification of all the constraints in the GTRBAC model [18]. The modular approach allows independent specification of SoD and temporal constraint definitions which can then be imported into the policy through the use of XML namespaces. Specification

of constraints separate from the policy allows reusable constraint definitions that can be used across multiple policies. Additionally, constraint definitions may be changed at one place without requiring change to all dependent policies, facilitating flexible adaptation.

X-GTRBAC additionally supports the specification of contextual constraints based on non-temporal attributes, usually associated with a role [16]. Contextual constraints on role attributes can be used in addition to temporal constraints to support finer granularity of control on user-to-role and permission-to-role assignments.

Top-level X-Grammar syntax of SoD and temporal constraint definitions is shown in Figures 3 and 4. The SoD constraints are included with the role definition in XRS (Figure 2), whereas the temporal constraints are included in assignment policies (Figure 6). An example XML instance of a temporal constraint definition appears in Appendix B.

```
<!-- Separation of Duty Definitions> ::=
  <XSoDDef [xsod_id = (id) ] >
    <!—SoDRoleSets>
 </XSoDDef>
```

Figure 3. Top-level syntax of SoD constraint definition

```
<!-- Definitions of Temporal Constraints>::=
<XTempConstDef [xtcd_id = (id) ] >
    [<!—Interval Expression>]
    [<!-- Periodic Time Expression>]
    [<!-- Duration Expression>]
 </XTempConstDef>
```

Figure 4. Top-level syntax of temporal constraint definition

Temporal constraints are of particular relevance to federated resource provisioning because it requires a set of fine-grained temporal constraints to adequately ensure resource protection while also ensuring its availability per the contractual requirements. This set includes constraints that control the periodicity, interval and duration of resource accesses (i.e. permission assignments) during and across provisioning sessions, in addition to trigger-based constraints that allow provisioning actions to be conditioned on related events. This represents a collection of stateful rules that can be configured in permission-to-role assignment policies. Doing so allows specification of usage-oriented resource protection policies to enforce usage control of federated resources.

3.2.3 Assignment Rules

An integral component of RBAC polices in our framework is the specification of rules for user-to-role and permission-to-role assignments. Rule-based assignment in RBAC policies provides a succinct declarative specification that is both scalable and flexible. It avoids the problem of rule-explosion since rules are specified at per-role (as opposed to per-user or per-resource) level. It is flexible since a declarative syntax allows rules to be modified without changing application code.

As noted earlier, the authenticating credential contains user attributes which are used by the access control processor for role assignment to users, whereas the authorization credential contains role attributes which are used by the access control processor for permission assignment to roles. (Role attributes may be used in user-to-role assignment too.) Additionally, a permission-to-role assignment policy may also include rules on resource attributes to allow specification of usage-oriented protection policy. Resource attributes capture semantic information (or meta-data) about resources, and avoid reliance on fixed resource locations. They also provide a mechanism for fine-grained permission assignment to roles based on precise resource characteristics.

To represent attributes of federated resources, we use application-specific attribute definitions (i.e. ontologies). (Each domain in the federation publishes attribute definitions of its resources to a well-known repository which may be imported for use in an assignment rule using XML namespaces.) The resource attributes are included in the object definition in an XML Permission Sheet (XPS). The top-level X-Grammar syntax of an XPS is shown in Figure 5.

```
<!-- XML Permission Sheet>::=
<XPS [xps_id = (id) ] >
 <Permission perm_id = id [prop= (prop op)] >
    <Object type= (type name) id= (id)>
    [<!-- Attributes>]
    </Object>
    <Operation> (access op) </Operation>
  </Permission>
</XPS>
```

Figure 5. Top-level X-Gramamr for XML Permission Sheet

```
<!-- XML User-Role Assignment Sheet>::=
<XURAS [xuras_id = (id) ] >
 <URA ura_id= (id) role_name= (name) >
  <AssignUsers>
   <AssignUser user_id= (id) >
  <AssignConstraint [op =(AND|OR|NOT|XOR)]>
    // opcode defaults to AND if none specified
   <AssignCondition cred_type=(type name)
   [pt_expr_id=(id) | d_expr_id=(id)] >
    <LogicalExpr [op = (AND|OR|NOT)]>
   // opcode defaults to AND if none specified
     [<!-- Predicate>]+
     </LogicalExpr>
   </AssignCondition>
    </AssignConstraint>
    </AssignUser>
 </AssignUsers>
</URA>
</XURAS>
```

Figure 6. Top-level syntax of user-to-role assignment policy

Our assignment policy schema specifies a logical expression syntax for rule specification. It does not, however, impose any restriction on the attributes that may be used for composing these rules. The existence and type checking of the queried attribute shall be done in an application-specific manner. For instance, user attributes can be verified through appropriate attribute authorities stated in the authentication credential.

The assignment policies are specified in our framework in an XML User to Role Assignment Sheet (XURAS) and XML Permission to Role Assignment Sheet (XPRAS). The top-level X-Grammar syntax of XURAS is shown in Figure 6 (XPRAS is analogous). Note that these policies include references to temporal constraint definitions. For example, `pt_expr_id` references a periodic time expression. An example XML instance of a role assignment policy appears in Appendix B.

A key feature of our rule specification format is that it allows combing rules from multiple sources to facilitate evaluation of multiple credentials. An assignment rule consists of an assignment constraint, which comprises of multiple assignment conditions. Each assignment condition contains a set of logical expressions to encode rules on a given credential type. Our logical expression syntax allows multiple logical expressions to be combined together in an appropriate rule combining mode using Boolean connectives. The modes supported by the evaluation engine in our prototype are AND (all rules must be true), OR (at least one rule must be true), and NOT (no rule must be true). Several levels of nesting are supported, each under a distinct mode, to allow a fine granularity of rule specification.

An assignment condition is satisfied if all of its included rules encoded using logical expressions are satisfied according to the respective mode. The multiple assignment conditions within an assignment constraint are combined according to a similar set of modes. This effectively allows an assignment constraint to be composed from multiple assignment conditions according to an appropriate combining mode. Role assignment occurs as a consequence of an assignment constraint being satisfied.

We note that the "AND" mode essentially implements "deny-overrides", whereas "OR" mode implements "permit-overrides". The NOT mode allows one to condition a role assignment based on negation. This is useful in instances when it is easier to express exclusion rather than inclusion criterion for membership in a role. Although negation is allowed in the body, it is not allowed in the consequence of a rule. This prevents contradictory rule sets to exist in the specification. This property is helpful when combining rules aimed at a given consequence, since one can always be sure that new rules will not clash with existing rules in the policy.

3.3 Salient Features

In this subsection, we describe the salient features of our policy framework that enables the solution to access management challenges in federated systems outlined in the paper. These features build upon the policy components discussed in the previous subsection.

3.3.1 Delegated Administration

The requirement for local autonomy, and hence decentralization of administrative control, in federated systems poses major challenges in developing access control policies, as has been earlier discussed. The mechanisms for credential and constraint specification in our framework help alleviate part of this problem. The other aspect is related to the policy administration and enforcement mechanism.

Decentralization of policy administration in X-GTRBAC is achieved through the notion of administrative domains. An administrative domain (or domain for short) is a unit of administrative authority. A federated system is then a multi-domain environment where each domain is responsible for managing the users and resources under its administrative control [16].

Delegation of responsibilities is essential to scalable decentralization. Delegation in federated systems is captured through some form of trust relationships [12]. In X-GTRBAC framework, the notion of delegation is elegantly captured through the use of role hierarchies: a junior role inherits all privileges of a senior role. An optional Delegation Constraint may be used in the role definition (See Figure 2) to limit the extent of delegation (in terms of time and associated privileges);

unrestricted delegation is otherwise assumed. This role-based delegation serves as the basis of trust in creating role mappings across multiple domains for federated information sharing. (Each domain in the federation publishes its role definitions to a well-known repository which may be imported for establishing appropriate role mappings using XML namespaces.) For instance, an Employee role from domain B that is mapped as junior to a Manager role in domain A would be allowed to exercise the Manager-level privileges in domain A per its delegation policy, without requiring explicit knowledge of domain B's access control policy.

Delegated administration requires access to a local compliance checker that can compute correspondence between mapped roles with respect to the local domain policies. The use of a compliance checker to ensure compliance of federated requests with local policies is a recognized mechanism for preserving local autonomy in distributed systems [19]. In our X-GTRBAC prototype, the compliance checker is incorporated into an authorization engine residing in each domain. It internally maintains a domain-specific mapping from the foreign (i.e. federated) roles to local roles according to the delegation policies of the local domain.

3.3.2 Lazy Instantiation

Domains, together with delegation, allow scalable decentralization of policy administration. However, the complexity of policy administration remains a concern, since an expressive access control policy would require the local authorization engine to maintain prohibitive amount of state information.

The use of credential-based specification in our policy framework helps mitigate this concern. A credential-based approach allows state information to be reduced since the requestor supplies the credentials relevant to the access request, facilitating lazy instantiation of policy rules. Therefore, the policy does not need to be distributed synchronously to all enforcement points [19]. In our X-GTRBAC prototype, lazy instantiation helps in state-reduction while enforcing the policy, since there is no need on part of the authorization engine to maintain persistent state information, i.e. store the assignment policies for all users and resources.

3.3.3 Credential Federation

The credential specification in a federated system must support federation requirement, as outlined earlier. Many existing distributed authorization schemes [12-15] do not address this requirement due to inherent limitation of their credential specification formats, as discussed in Section 3.2.1.

The modular, attribute-based credential specification in the X-GTRBAC framework allows credential federation through the use of interoperable protocols. In fact, the on-going work on our prototype has assumed a SAML-compliant format

for authenticating credentials. SAML standard [7] states that authentication information may be available in various forms, such as X.509 Attribute Certificates, Kerberos tickets or passwords. We employ appropriate translation mechanisms for them to be used with our X-GTRBAC syntax. Since our rule specification supports combing rules from multiple sources, this allows use of our specification in situations when multiple policies are necessary to evaluate the request of a federated user.

For example, consider an extension to the assignment policy in Figure B.2 in Appendix B. This policy requires the user to provide an X.509 Attribute Certificate including certain attributes. Another policy might require the user to also provide a Kerberos ticket. Using our specification, this can be achieved by simply adding another `<AssignCondition>` tag with `cred_type="Kerberos"`, and including the desired rules on credential attributes. The rule combining mechanism discussed earlier can then be used for evaluating this composite policy.

In addition to credential federation, a related requirement is that of providing single sign on (SSO). SSO enables persistent authorization support for federated users within a single login session. Our framework supports SSO through the use of digitally signed SAML statements that capture an authorization decision already issued by a domain corresponding to a user request. This decision statement can subsequently be reused by the user at a domain within the federation without getting re-authenticated, subject to the acceptance of the digital signature.

3.3.4 Usage Control

Persistent protection of federated resources requires effective usage control mechanisms. Traditional access control models do not take this usage-oriented view, and hence are inadequate to capture the protection requirements associated with federated resource sharing.

X-GTRBAC framework allows the specification of usage-oriented resource protection policies as discussed in Section 3.2.2. However, enforcement of these policies requires effective session management mechanism. In our X-GTRBAC prototype, this session management support is provided through the implementation of periodicity, interval and duration constraints associated with resource provisioning and de-provisioning. In addition, it also implements trigger-based constraints that allow provisioning and de-provisioning actions to be conditioned on related events. For example, the provisioning of a resource may be automatically discontinued when the associated duration constraint expires. This set of constraints represents a collection of stateful rules that are configured in permission-to-role assignment policies, and enforced by the session management

mechanism. Stateful rules help keep the complexity of maintaining the policy low.

3.4 Policy Composition

An overall X-GTRBAC policy is composed[4] from these individual policy components as shown in Figure 7. The complete X-Grammar policy syntax is provided in Appendix A.

```
<!-- Policy Definition>      ::=
<Policy policy_id =(id)>
     <PolicyName> (name) </PolicyName>
     <!-- XML User Sheet>
     <!-- XML Role Sheet>
     <!-- XML Permission Sheet>
     <!-- XML User-Role Assignment Sheet>
     <!-- XML Permission-Role Assignment Sheet>
</Policy>
```

Figure 7. The overall X-GTRBAC policy

4. SYSTEM ARCHITECTURE

In this section, we present the system architecture of our X-GTRBAC prototype designed for access management in a federated environment. In particular, we apply our policy framework in a federated digital library environment (with read-only access). The use of this prototype illustrates the design and enforcement of access control policies for secure federation of XML-based digital documents. XML is increasingly being used as the preferred digital format on the Web; therefore we work with XML documents.

[4] Policy composition as used here refers to combining multiple distinct components of a policy together into a single document. This should not be confused with composing a single policy from multiple, potentially overlapping, policies through formal analysis.

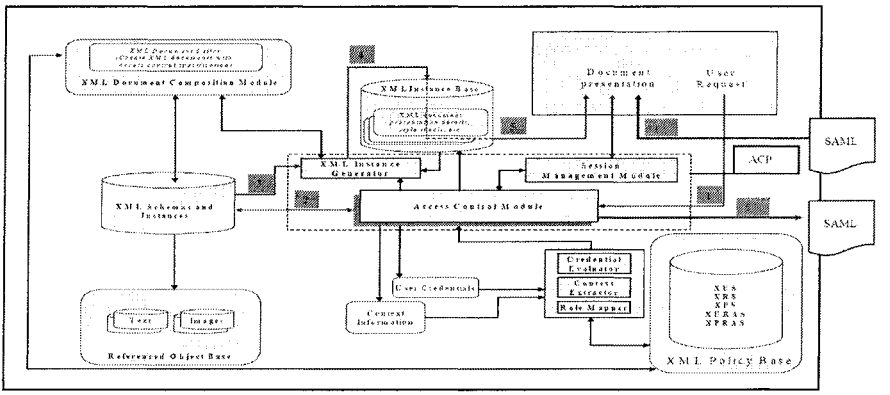

Figure 8. The system architecture for federated digital library prototype

The system architecture is shown in Figure 8. This architecture is implemented at each participating site in the federation. We now highlight the role of the key components of the prototype.

4.1 Policy specification

XML Document Composition Module (XDCM) is used by each participating site to compose policy documents. Each site first encodes its X-Grammar policy definitions which are then translated into XML schemas using a custom translator and exported to XDCM. The policy documents are then composed in XML inside the XDCM, and verified against the imported schema definitions.

XML Policy Base (XPB) contains all policy related XML documents composed by XDCM. These include XML User Sheet (XUS), XML Role Sheet (XRS), XML Permission Sheet (XPS), XML User to Role Assignment Sheet (XURAS), and XML Permission to Role Assignment Sheet (XPRAS). Also stored in XPB are the constraint definitions, including XSoDDef (Figure 3) and XTempConstDef (Figure 4).

XML Schemas and Instances contains actual XML documents at a participating site to which the users of the federated library will be requesting access. Each site also includes a list of objects that may not be available locally but are known to exist in the federation, so that the users can request access to them through their local site. *Referenced Object Base* constitutes the physical objects present in the local system that are referenced from within the XML documents. Note that binary encoding allows objects to be embedded within XML documents, and those objects may themselves be protected resources.

As is the usual case, the default policy of the federation is no authorization, i.e. no user is authorized to access any document unless there exists an explicit rule granting him/her an authorization.

4.2 Policy enforcement

Upon receiving an access request, the *Access Control Module (ACM)* extracts the policy information from the policy base and works closely with the XPB to enforce the authorization constraints on the release of the request resource. The access request may either be from a local or a federated user. The request includes the requesting subject, the requested resource and the requested permissions on the resource. It may optionally include an authenticating or authorization credential to assist in authorization decision[5]. We use a SAML-compliant format for all access requests. If the requested resource is not available within the system, the ACM simply returns (or appropriately redirects) the request. Otherwise, it proceeds as follows.

As a first step, the ACM forwards the access request (expressed as a SAML Authorization Decision Query) to the *Credential Evaluator (CrE)*. It simultaneously forwards the currently available contextual information to the *Context Evaluator (CoE)*. CrE first translates the credential included in the request from the SAML format to X-GTRBAC format. Based on the kind of credential, CrE does the following.

If it is an authenticating credential (expressed as a SAML Attribute Statement included in the query), the CrE assigns the user to an appropriate role within the system according to the user-to-role assignment policy after consulting the XPB. This process requires input from the CoE regarding (successful) evaluation of temporal and non-temporal contextual constraints before the role assignment takes place. The non-temporal constraints not only involve rules on user attributes but may also involve rules on attributes of the role for a fine-grained access control. Following a successful role assignment, the user is issued an authorization credential by the system.

If it is an authorization credential (expressed as a previously issued SAML Authorization Decision Statement included in the query), CrE inspects if the role information in the credential corresponds to a local role. If it doesn't, it means that it is an SSO request and includes the role of the federated user in his/her original domain. In this case, CrE invokes the *Role Mapper (RM)* to map the user to a local

[5] In our current prototype, the supporting credentials are explicitly included in the access request. The support for credential collection (using mechanisms as SAML Authentication Request protocol) is explicitly stated earlier to be outside the scope of this work.

role according to the delegation policy of the system. After this step, the user acquires the privileges of the assigned or mapped role in the local system.

After establishing the role of the user, the next step is to determine the authorization of the user to obtain the requested permission on the requested resource. At this stage, the CrE evaluates the authorization credential of the user according to the permission-to-role assignment policy after consulting the XPB. This process requires input from the CoE regarding (successful) evaluation of temporal and non-temporal contextual constraints before the permission assignment takes place. The non-temporal constraints not only involve rules on role attributes but may also involve rules on attributes of the resource for a fine-grained access control. The result of the evaluation is returned to ACM (expressed as a SAML Authorization Decision Statement), together with any applicable resource provisioning constraints retrieved from XPB, as discussed in Section 3.2.2.

As a final step, the ACM forwards the authorization information for the user to the XIG. XIG consults the access rights of the requesting user on the requested XML document, and accordingly generates XML views in response to the request. Such XML views are cached in *XML Instance Base (XIB). Session Management Module (SMM)* is responsible for monitoring the provisioning and de-provisioning constraints associated with the requested document, as described in Section 3.3.4. The ACM, SMM, and XIG together constitute the *XML Access Control Processor (ACP).*

5. RELATED WORK

While using policies for management of systems is not an entirely novel concept, and has been applied previously in the context of network systems management [20], the policy-based approach for access management in federated systems has not been deeply investigated.

One notable example of policy-based language for systems management is Ponder [21]. Ponder is a declarative policy language with the ability to support authorization and delegation policies, as well as obligation policies (which are condition-action rules, much like trigger-based constraints in our framework). However, authorization policies in Ponder are primarily aimed at allowing network users to manage network objects, with known user groups and object locations, and are therefore inadequate for a federated environment where users and resources are not identified in advance. It therefore does not support credential specification and federation requirements for access management in federated systems discussed in the paper.

Ponder supports specification of contextual constraints, based on temporal and non-temporal parameters. However, contextual constraint specification is tied

into the authorization policies, which reduces their modularity, and hence flexibility. Ponder also supports specification of SoD constraints through the use of meta-policies. However, the specification is at user-level, and is more complicated to maintain in a federated environment as opposed to a role-based SoD constraint. On the other hand, Ponder is well suited to the task that it is designed for, i.e. network services management. It has a well-developed management toolkit that allows policy specification, deployment, and dynamic adaptation suitable for a network environment.

The access control model for federated systems presented in [3] is based on a tightly coupled architecture. It concerns with defining principles for designing access control policies in federated systems, and does not deal with policy-based management issues. It therefore does not address the particular issues related to credential specification, credential federation, usage control or session management highlighted in this paper.

[22] presents an RBAC model for federated information systems. This system supports credential federation and SSO. However, it does not support all of our design requirements, including specification of semantic or contextual constraints, and usage-oriented resource protection policies.

Various policy models have earlier been used for access control in centralized and traditional distributed systems, but not many approaches have been designed to meet the requirements for policy-based access management in federated systems as described in this paper. Akenti [23] and Permis [24] are access control systems which use policies encoded in X.509 attribute certificates. Both assume authenticating credentials to be used for issuance of authorization certificates, much like our approach. Akenti supports discretionary access control (i.e. identity based), leading to rule explosion in policy rule set. Permis uses role-based access control; it, however, does not provide support for specification and enforcement of user-to-role and role-to-permission assignment policies. Both these schemes also provide no support for specification of semantic or contextual constraints, and usage-oriented resource protection policies.

Shibboleth [25] and Liberty Alliance [26] define protocols for attribute-based authentication in support of SSO in Web-based environments. The attributes in Shibboleth are always acquired from his/her home site by the resource provider, whereas those in Liberty Alliance protocol can be provided by any identity provider on the Internet. Liberty Alliance protocol therefore establishes a circle of trust between identity provider and resource providers. Both schemes provide particular emphasis on user privacy, and the identity of the user is not known to the resource provider. However, the role of these schemes is limited to distributed authentication, and providing attribute information for a user to be used in authorization decisions. They do not include mechanisms for specifying and enforcing authorization policies.

With reference to our emphasis on usage control, a relevant work appears in [27]. It presents a usage control specification to extend the capabilities of traditional access control models to support resource protection policies. They provide a logic defining states, authorizations, and actions relevant to resource usage, and use the notion of mutable attributes to allow state transitions and enforce usage control. The significant contribution of the model is that it provides logic-based semantics of usage control. However, it does not provide an enforcement mechanism.

SAML [7] and XACML [8] are emerging specifications aimed at addressing different aspects of distributed access management. As noted earlier, SAML primarily provides a mechanism for credential federation, but does not provide any policies for use of those credentials. Also, SAML does not incorporate a way to establish trust between business partners exchanging credentials. XACML provides support for policy specification for expressing access control policies. XACML can also combine multiple rules and multiple policies under various modes (deny -overrides, permit-overrides), which is similar to the rule and condition combining feature, respectively, in our language. XACML can be configured to support role-based access control and usage-oriented resource protection policies. It, however, primarily acts only as a PDP (Policy Decision Point) and lacks the temporal infrastructure to enforce the access control policy, such as the session management mechanism to enforce usage control in our framework. Our framework can therefore provide the functionality of both a PDP and a PEP (Policy Enforcement Point).

6. CONCLUSION

In this paper, we have presented the salient features of our X-GTRBAC policy framework for access management in federated systems. Our framework has been designed to address the key challenges for developing access control policies for federated information sharing. In particular, it supports: (i) decentralized administration while preserving local autonomy through the use of trust relationships captured through role-base delegation, (ii) fine-grained access control while avoiding rule-explosion in the policy through a succinct declarative credential specification, (iii) credential federation through the use of interoperable protocols, with support for single sign on for federated users, (iv) specification and enforcement of semantic and contextual constraints to support integrity requirements and contractual obligations, and (v) usage control in resource provisioning through effective session management.

The resource protection requirements in our framework are related to the Digital Rights Management (DRM) approach [28]. DRM, however, is a much broad

notion, and also includes mechanisms for protection of resources while outside the administrative control of the owner. This usually requires self-protection mechanisms, i.e. the use of embedded features (such as watermarks). We only deal with policies for resource protection under administrative control of the owner, and do not make assumptions about physical protection of resources.

There are other aspects of access management that need to be incorporated in our policy framework. Our current approach for role mapping abides by the local autonomy principle, and hence no form of external access mediation is necessitated. In a more general case, this may on one hand be overly-restrictive, and on the other hand lead to security breaches due to transitive establishment of undesirable delegation links. Therefore, a mediation mechanism is necessary to fairly regulate federated information sharing while ensuring security of federated resources. Composing an access mediation policy in a federated system poses considerable challenge since participating sites do not have a-priori knowledge of each other's access control polices. These challenges are likely to be addressed as part of future work.

ACKNOWLEDGEMENTS

Portions of this work have been supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University, and the National Science Foundation under NSF Grant# IIS-0242419.

REFERENCES

[1] http://www.fas.org/irp/doddir/dod/d8320_2.pdf

[2] http://www.educause.edu/ir/library/pdf/erm0348.pdf

[3] S. D. C. di Vimercati, P. Samarati, "Access control in federated systems", In proceedings of ACM New Security Paradigm Workshop, pages 87-99, Lake Arrowhead, CA, USA, 1996.

[4] D. D. Clark, D. R. Wilson, "A comparison of commercial and military computer security policies," In IEEE Symposium on Security and Privacy, pages 184-194, Oakland, April 1987.

[5] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, "Role-Based Access Control Models", IEEE Computer 29(2): 38-47, IEEE Press, 1996.

[6] http://www.enterprisenetworksandservers.com/monthly/art.php/1117

[7] http://xml.coverpages.org/saml.html

[8] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[9] http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

[10] http://www-128.ibm.com/developerworks/library/specification/ws-polfram/

[11] http://www.nwfusion.com/news/2002/0715saml.html

[12] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "KeyNote: Trust management for public-key infrastructures," in Security Protocols International Workshop, Springer LNCS, no. 1550, pp. 59-63, 1998.

[13] C. M. Ellison, "SPKI requirements," RFC 2692, Internet Engineering Task Force Draft IETF, Sept. 1999. See http://www.ietf.org/rfc/rfc2692.txt.

[14] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure, or: Assigning roles to strangers", In Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 2–14, 2000. IEEE Press.

[15] N. Li, J. C. Mitchell, W. H. Winsborough, "Design of a role-based trust management framework", In Proceedings of the 2002 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2002.

[16] J. B. D. Joshi, R. Bhatti, E. Bertino, A. Ghafoor, "An Access Control Language for Multi-Domain Environments", IEEE Internet Computing, vol. 8, no. 6, pp. 40-50, November/December 2004.

[17] J. B. D. Joshi, E. Bertino, U. Latif, A. Ghafoor, "Generalized Temporal Role Based Access Control Model (GTRBAC) ", IEEE Transaction on Knowledge and Data Engineering, vol. 17, no. 1, January 2005.

[18] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor, "X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control", ACM Transactions on Information and System Security (TISSEC), Vol. 8, No. 2.

[19] A. Keromytis, S. Ioannidis, M. Greenwald, J. Smith, "The STRONGMAN Architecture", In Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), Washington, D.C. April 22-24, 2003.

[20] L. Lymberopoulos, E. Lupu, M. Sloman, "An Adaptive Policy Based Management Framework for Network Services Management", In Special Issue on Policy Based Management of Networks and Services, Journal of Networks and Systems Management, Vol. 11, No. 3, Sep. 2003.

[21] N. Damianou, N. Dulay, E. Lupu, M Sloman, "The Ponder Specification Language", Workshop on Policies for Distributed Systems and Networks (Policy2001), HP Labs Bristol, 29-31 Jan 2001.

[22] K. Taylor, J. Murty, "Implementing role based access control for federated information systems on the web", Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, p.87-95, February 01, 2003, Adelaide, Australia.

[23] M.Thompson, A. Essiari, S. Mudumbai, "Certificate-based Authorization Policy in a PKI Environment", ACM Transactions on Information and System Security, (TISSEC), Volume 6, Issue 4 (November 2003) pp: 566-588.

[24] D.W. Chadwick, A. Otenko, "The PERMIS X.509 role based privilege management infrastructure", In proceedings of the seventh ACM Symposium on Access Control Models and Technologies, Monterey, California, USA.

[25] http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-latest.pdf

[26] http://www.projectliberty.org/resources/specifications.php

[27] X. Zhang, J. Park, F. Parisi-Presicce, R. Sandhu, "A Logical Specification for Usage Control", In proceedings of the ninth ACM Symposium on Access Control Models and Technologies, Monterey, California, USA .

[28] B. Rosenblatt, B. Trippe, S. Mooney, "Digital Rights Management: Business and Technology", New York: Hungry Minds/John Wiley and Sons, 2001.

Appendix A:

X-GTRBAC Grammar

[Basic Definitions]

```
<!-- Policy Definition> ::=
<Policy policy_id =(id)>
 <PolicyName> (name) </PolicyName>
 <!-- XML User Sheet>
 <!-- XML Role Sheet>
 <!-- XML Permission Sheet>
 <!-- XML User-Role Assignment Sheet>
 <!-- XML Permission-Role Assignment Sheet>
 [<!-- Local Policy Definitions>]
 [<!-- Policy Relationship Definitions>]
</Policy>
<!-- XML User Sheet>                    ::=
<XUS [xus_id = (id) ]>
 [<!-- Definitions of Credential Types>]
 <!-- User Definitions>
</XUS>
<!-- Definitions of Credential Types>          ::=
<XCredType [xctd_id = (id) ] >
 [<!-- Credential Type Definition>]+
</XCredType>
<!-- Credential Type Definition>
::=     <CredType cred_type_id = (id)
          type_name= (type name) >
 <!-- Attribute List>
</CredType >
<!-- Attribute List> ::= <AttributeList>
   [<!-- Attribute Definition>]+
</AttributeList>
<!-- Attribute Definition>                  ::
 <Attribute name= (name) usage = "mand | opt"
   type = (type) />
<!-- User Definitions >                     ::=<Users>
   [<!-- User Definition>]+
</Users>
<!-- User Definition> ::=      <User user_id = (id)>
   <UserName> [(name)] </UserName>
   <!—CredType>
   <MaxRoles>(number)</MaxRoles>
</User>
 <!—CredType  > ::=
 <CredType cred_type_id = (id)
   type_name= (type name) >
 [<!—Header>]
 <!-- Credential Expression>
</CredType>
<!-- Credential Expression > ::= <CredExpr>
  <!-- AttributeValuePairs>
  <!-- DomainSet>
</CredExpr>
<!-- AttributeValuePairs> ::=
[<Attribute name= (name)> (value) </Attribute>]  +
<!-- XML Role Sheet>                        ::=
<XRS [xrs_id = (id) ]>
  [<!-- Role Definition>]+
</XRS>
<!-- Role Definition>                       ::=
<Role role_id = (id)
       role_name = (role name)>
 [<!-- Cred Type>]
 [<!—(En|Dis)abling Constraint>]
 [<!—[De]Activation Constraint>]
 (<SSDRoleSetID> (id) </SSDRoleSetID>)*
 (<DSDRoleSetID> (id) </DSDRoleSetID>)*
   [<Junior> (name) </Junior>]
   [<Senior> (name) </Senior>]
   [<LinkedRole type= (delegator |
      delegatee) >(name)</LinkedRole>]
   [<!—Delegation Constraint>]
   [<Cardinality> (number) </Cardinality>]
  </Role>
  <!-- Separation of Duty Definitions>
  ::=    <XSoDDef [xsod_id = (id) ]>
       <!—SoDRoleSets>
         </XSoDDef>
  <!-- SoDRoleSets > ::=
       [<!—SSDRoleSets>] [<!—DSDRoleSets>]
  <!-- SSDRoleSets > ::=     <SSDRoleSets>
       [<!—SSDRoleSet>]+
     </SSDRoleSets>
  <!—SSDRoleSet> ::=    <SSDRoleSet>
       [<SSDRole ssd_role_set_id =(id)
         ssd_cardinality = (number)>
         (role name)
       </SSDRole>]+
 </SSDRoleSet>
  <!-- DomainSet>  ::=     <DomainSet>
     [<!—DomainID>]+
  </DomainSet>
  <!-- DomainID>::=      <DomainID>(id)</DomainID>
  <!-- DSDRoleSets >  ::=     <DSDRoleSets>
   [<!—DSDRoleSet>]+
 </DSDRoleSets>
  <!—DSDRoleSet>::=    <DSDRoleSet>
   [<DSDRole dsd_role_set_id =(id)
        dsd_cardinality = (number)>
        (role name)
 </DSDRole>]+
 </DSDRoleSet>
  <!-- XML Permission Sheet>::=
  <XPS [xps_id = (id) ]>
   [<!-- Permission Definition>]+
  </XPS>
  <!-- Permission Definition>          ::=
  <Permission perm_id = id [prop= (prop op)] >
  <Object type= (type name) id= (id)>
   [<!-- Attributes>]
  </Object>
  <Operation  context=  (name)>  (access  op)
</Operation>
  <!-- DomainSet>
  </Permission>
  <!-- XML User-Role Assignment Sheet>::=
  <XURAS [xuras_id = (id) ]>
   [<!- User-role Assignment>]+
  </XURAS>
  <!-- User-role Assignment>::=
  <URA       ura_id= (id)       role_name= (name) >
 <AssignUsers>
     [< !—Assign User>]+
 </AssignUsers>
  </URA>
  <!—[De]Assign User >              ::=
 <[De]AssignUser user_id=(id)>
   <!—[De]Assign Constraint >
 </[De]AssignUser>
  <!-- XML Permission-Role Assignment Sheet>::=
```

```
<XPRAS [xpras_id = (id)]>
 [<!-- Permission-Role Assignment>]+
</XPRAS>
<!-- Permission-Role Assignment>::=
 <PRA pra_id=(id) role_name=(name)>
 <AssignPermissions>
    [<!—Assign Permission>]+
</AssignPermissions>
 </PRA>
 <   !—[De]Assign    Permission>                    ::=
<[De]AssignPermission perm_id=(id)>          <!—
[De]Assign Constraint >
 </[De]AssignPermission>
 <!—[De]Assign Constraint>          ::=
<[De]AssignConstraint[op =(AND|OR|NOT|XOR)]>
   // opcode defaults to AND if none specified
  [<!—[De] Assign Condition>]+
 </[De]AssignConstraint>
<!—[De]Assign Condition>                ::=
<[De]AssignCondition cred_type=(type name)
    [pt_expr_id=(id) | d_expr_id=(id)] >
    [<!-- Logical Expression>]
</[De]AssignCondition>
 <!—(En|Dis)abling Constraint>          ::=
<(En|Dis)abConstraint[op = (AND|OR|NOT)]>
    // opcode defaults to AND if none specified
   [<!-- (En|Dis)abling Condition>]+
</(En|Dis)abConstraint>
 <!—(En|Dis)abling Condition>          ::=
<(En|Dis)abCondition [pt_expr_id=(id) |
                      d_expr_id=(id)] >
   [<!-- Logical Expression>]
</(En|Dis)abCondition>
 <!—[De]Activation Constraint> ::=
 <[De] ActivConstraint[op = (AND|OR|NOT)]>
  // opcode defaults to AND if none specified
   [<!—[De]ActivationCondition>]+
</[De]ActivConstraint>
 <!—[De]Activation Condition>          ::=
<[De]ActivCondition [d_expr_id=(id)]>
  <!-- Logical Expression>]
</[De]ActivCondition >
 <!-- Logical Expression>                ::=
 <LogicalExpr [op = (AND|OR|NOT)]>
 // opcode defaults to AND if none specified
 [<!-- Predicate>]+
 </LogicalExpr>
 <!-- Predicate>                    ::= <Predicate>
  {<Operator> (gt|lt|eq|neq) </Operator>
   [<FuncName>(name)</FuncName>]
   [<ParamName>(name)</ParamName>]+
   <RetValue>(value)</RetValue> |
  | <!-LogicalExpression>
 </Predicate>
```

[Temporal Definitions]

```
 <!--   Definitions   of   Temporal   Constraints>::=
<XTempConstDef [xtcd_id = (id)]>
    [<!—Interval Expression>]
    [<!-- Periodic Time Expression>]
    [<!-- Duration Expression>]
 </XTempConstDef>
 <!-- Periodic Time Expression>      ::=
<PeriodicTimeExpr pt_expr_id = (id)
  <!-- Start Time Expression>
 </PeriodicTimeExpr>
```

```
 <!—Interval Expression>              ::=
 <IntervalExpr i_expr_id = (id)>
 <begin> (date)</begin>
  <end>(date)</end>
 </IntervalExpr>
 <!-- Start Time Expression>  ::= <StartTimeExpr
 [pt_id_ref = (pt_id)]>
  [<Year> (all|odd|even) /<Year>]
  [<!--MonthSet>]
  [<!--WeekSet>]
  [<!--DaySet>]
 </StartTimeExpr>
 <!--MonthSet>
::=<MonthSet>
  (<Month>(1|..|12)</Month>)1-12
  (represents # of months from the start of current Year)
 </MonthSet >
 <!--WeekSet>                          ::=
<WeekSet>
  (<Week>(1|..|5)</Week>)1-5
   (represents # of weeks from the start of current Month)
 </WeekSet >
 <!--DaySet>                           ::=
<DaySet>
  (<Day>(1|..|7)</Day>)1-7
  (represents # of days from the start of current Week)
 </DaySet >
 <!-- Duration Expression>          ::=
 <DurationExpr d_expr_id = (id)>
  <cal>(Years|Months|Weeks|Days)</cal>
  <len> (number)</len>
 </DurationExpr>
```

[TM Credential Definitions]

```
 <!--Header>          ::=  <Header>
   <!-- Principal >
   <!-- Issuer >
   <!-- Validity>
    [<!-- Digital Signature >]
   </Header>
 <!-- Issuer>          ::=          <Issuer>
  <!-- Principal>
  </Issuer>
 <!-- Principal>::= <Principal>
<NameID mode=(saml:NameID@format)>
       (String) </NameID>
 </Principal>
 <!-- Validity>          ::= <Validity>
 <IssueTime>(xs:dateTime)</IssueTime>
  [<NotBefore>(xs:dateTime)</NotBefore>]
  [<NotOnOrAfter>(xs:dateTime)
    </NotOnOrAfter>]
 </Validity>
 <!-- Digital Signature > ::= <Dsig>
 (ds:Signature) </Dsig>
 <!—Delegation Constraint>  ::=
 <DelegationConstraint [op = (AND|OR|NOT)]>
  // opcode defaults to AND if none specified
 [<!-- Delegation Condition>]+
 <!—Delegation Condition>          ::=
 <DelegationCondition [pt_expr_id=(id)]
                      [i_expr_id=(id)] >
   [<!-- Logical Expression>]
</DelegationCondition> [i_expr_id = (id)] [i_expr_id = (id)] >
```

Appendix B

```
<?xml version="1.0" encoding="UTF-8"?>

<XTempConstDef xtcd_id="IFIP_XTCD">
  <IntervalExpr i_expr_id="Year2005">
       <begin>2005:01:01</begin>
       <end>2005:12:31</end>
  </IntervalExpr>
  <DurationExpr d_expr_id="SixWeeks">
       <cal>Weeks</cal>
       <len>6</len>
  </DurationExpr>
  <DurationExpr d_expr_id="OneWeek">
       <cal>Weeks</cal>
       <len>1</len>
  </DurationExpr>
  <PeriodicTimeExpr
pt_expr_id="PTQuarterWeekSeven"
i_expr_id="Year2005"
d_expr_id="SixWeeks">
    <StartTimeExpr>
       <Year>all</Year>
       <MonthSet>
           <Month>1</Month>
           <Month>4</Month>
           <Month>7</Month>
           <Month>10</Month>
       </MonthSet>
       <WeekSet>
           <Week>7</Week>
       </WeekSet>
    </StartTimeExpr>
  </PeriodicTimeExpr>
</XTempConstDef>
```

Figure B.1: This temporal constraint definition includes a periodic time expression (PTE) which states that the access is allowed beginning the seventh week of every quarter of year 2005, and is allowed for a duration of six weeks. Note that duration expression and interval expression are referenced inside a PTE.

```
<?xml version="1.0" encoding="UTF-8"?>

<XURAS xuras_id="IFIP_XURAS">
 <URA ura_id="uraBorrow" role_name="Borrower">
  <AssignUsers>
   <AssignUser user_id="any">
    <AssignConstraint>
      <AssignCondition cred_type="X.509 AC"
          pt_expr_id="PTQuarterWeekSeven">
       <LogicalExpr op="AND">
        <Predicate>
         <LogicalExpr op="OR">
           <Predicate>
             <Operator>neq</Operator>
             <FuncName>hasValue</FuncName>
             <ParamName>DLN</ParamName>
             <RetValue>null</RetValue>
           </Predicate>
           <Predicate>
             <Operator>neq</Operator>
             <FuncName>hasValue</FuncName>
             <ParamName>SSN</ParamName>
             <RetValue>null</RetValue>
           </Predicate>
         </LogicalExpr>
        </Predicate>
        <Predicate>
         <Operator>gt</Operator>
         <FuncName>hasValue</FuncName>
         <ParamName>valid_date</ParamName>
         <RetValue>2005:12:31</RetValue>
        </Predicate>
       </LogicalExpr>
      </AssignCondition>
    </AssignConstraint>
   </AssignUser>
  </AssignUsers>
 </URA>
</XURAS>
```

Figure B.2: This is a role assignment policy for the `Borrower` role in the federated digital library system. It states that any user (`any` is a keyword) can be assigned to this role if he/she supplies a SAML authentication credential supporting the following conditions: (i) credential has a non-null DLN *or* SSN attribute for the user, *and* (ii) the credential is valid beyond the year 2005. Additionally, the PTE of Figure B.1 is referenced in the assignment policy to constrain the applicable time period of the policy.