A Framework for Flexible Access Control in Digital Library Systems*

Indrajit Ray and Sudip Chakraborty

Colorado State University
Fort Collins, CO 80523, USA
{indrajit, sudip}@cs.colostate.edu

Abstract. Traditional access control models are often found to be inadequate for digital libraries. This is because the user population for digital libraries is very dynamic and not completely known in advance. In addition, the objects stored in a digital library are characterized by fine-grained behavioral interfaces and highly-contextualized access restrictions that require a user's access privileges to be updated dynamically. These motivate us to propose a trust-based authorization model for digital libraries. Access privileges can be associated with both objects and content classes. Trust levels associated with these specify the minimum acceptable level of trust needed of a user to allow access to the objects. We use a vector trust model to calculate the system's trust about a user. The model uses a number of different types of information about a user, for example, prior usage history, credentials, recommendations etc., to calculate the trust level in a dynamic manner and thus achieve a fine-grained access control.

1 Introduction

Access control is one of the major concerns for content-providers on the Internet. Without a proper access control mechanism confidentiality and integrity of information cannot be guaranteed. Different models exist for specifying access control policies like discretionary access control, mandatory access control and role-based access control. However, with increasing complexity of systems and security concerns, a single model does not suffice to provide access control in all systems. In this work we address the problem of access control in digital libraries.

Conventional access control models specify an access control policy as a triple $\langle subject, object, permission \rangle$. This states that that a subject (user) is authorized to exercise some permission on an object. The traditional models implicitly assume that the user population is known a-priori. In a digital library system (DLS) the user population is vast and dynamic. It is almost next to impossible to know all the users before hand. Thus traditional access control mechanisms that rely on knowing the user and

* This work was partially supported by the U.S. Air Force Research Laboratory (AFRL and the Federal Aviation Administration (FAA) under contract F30602-03-1-0101 and by the National Science Foundation (NSF) of the USA under grant IIS-0242258. Any opinions, findings, and conclusions or recommendations expressed in this publication are solely those of the authors and do not necessarily represent those of the AFRL, the FAA, or the NSF.

associating permissions with them fail significantly in digital libraries. A digital library environment poses some additional challenges for access control [1]. The users of a digital library often need access from remote locations or by following links from remote documents. Thus it does not suffice to merely control access to documents local to the digital library. The access control policies are often based on user qualifications and characteristics. For example, a user can be given access to R-rated movies only if she is older than 18 years. Last, but not the least, a digital library needs to support access control to its objects based on the object content in addition to object identity. For example, high resolution satellite images of nuclear power plants can be made available only to citizens of the country.

In one of the early works on access control in digital libraries, Gladney [2] proposes a scheme called DACM (Document Access Control Methods). The basic idea is geared toward discretionary access control with some extensions to handle mandatory access control. Though it is a scalable mechanism, it does not have the provision to dynamically change user privileges. Researchers have also proposed credential-based access control [3–5], to address the problem of unknown users. In these models a user has to produce one or more credentials that have been certified by one or more third parties. The credential provides information about the rights, qualifications, responsibilities and other characteristics attributable to its bearer by the third parties. These third parties need to be trusted by the service provider. Bertino et al [1] develops a credential based system for enforcing access control in digital library system. Winslett et al. [6] also propose a credential-based mechanism to assure security and privacy for digital library transactions. Skogsrud et al. [7] introduce a model-driven trust negotiation framework called Trust-Serv for digital library environments. It uses credentials for establishing trust relationships. Ryutov et al. [8] present a framework named ATNAC (Adaptive Trust Negotiation and Access control) to protect sensitive resources in e-commerce. It is designed by integrating two existing systems – TrustBuilder with an adaptive access control API called, GAA-API (Generic Authorization and Access control). In [9], Adam et. al propose a content-based authorization model for digital library environments. Authorization is specified based on positive and negative qualifications and characteristics of the user which are expressed using credentials. Bonatti and Samarati [10] propose a uniform formal framework to regulate service access and information disclosure on the Internet. The regulation is based on credentials.

As is evident from the above discussion most access control methodologies for digital libraries use credential in one form or the other. Credential based access control, however, is not completely satisfactory. For one, a credential based system implements a binary notion of trust. If a user's credentials are accepted the corresponding privileges are allowed; if the credentials are not successfully validated the user is denied access. There is no way to implement fine-grained access control without requiring a large set of credentials. Additionally, reasoned decisions cannot be made in the face of incomplete, insufficient or inconclusive information. For example, let us assume that to validate a particular user credential three different credential certifying authorities need to be consulted. If, for any reason, one of these trusted authorities is not reachable and could not validate the credential, while the other two successfully validated the credential, the access will still be denied. Current credential based systems cannot implement

a notion of limited access. Third, the objects stored in a digital library are characterized by fine-grained behavioral interfaces and highly-contextualized access restrictions that require a user's access privileges to be updated dynamically. Credential based access control models do not keep track of a user's behavior history. Access is provided based solely on the credentials presented during the specific access request. Thus, a user's access privileges cannot be updated dynamically under this model.

Note that a basic requirement of any access control mechanism is to determine if a user can be trusted with the access privileges. The notion of *trust* thus plays a crucial role. Classical access control models establish trust in the user based on the user's identity. Credential based access control does this by means of attestations from a-priori trusted authorities. Thus, using trust relationships to enable secure interactions among computational agents or to enforce proper policy seems appropriate. This motivates us to propose a new trust-based access control framework in this work. It is based on the vector model of trust that we had proposed earlier [11]. We use a prototype digital library system – called the DLS system – that we are developing at our institution as the testbed for the new access control framework. In the DLS system the digital library contents are classified into a number of content type categories. Each content type category is associated with a trust level. A user who is trusted to the trust level of the content category or higher can access the contents. The trust level of the user can be established via a number of different means. For example, the trust level can be determined based on past interactions with the user. It can be established based on some credentials presented by the user. It can also be established by virtue of recommendations provided by a partner digital library.

The rest of the paper is organized as follows. Section 2 provides an overview of access control in the DLS digital library system. In particular, it talks about how a notion of trust is used in access control decisions. Section 3 describes the access control model. In section 4 we outline how trust relationships are established between the DLS system and its user population. Section 5 gives the architecture of the DLS access control framework. Finally, we conclude our discussion in section 6 with a summary for future work.

2 Digital library access control model

Access control in the DLS digital library system is implemented using a multi-level trust model. For a digital library, access privileges to a particular category content is restricted to the users with a certain trust level. This trust level can be determined from many different pieces of information available about the user. For example, trust level can be determined from the credentials presented during an access request. Trust levels can be established based on previous behavior of the user. Trust levels can be established from certain physical properties of the user. Changes to the 'trust-level' changes the access privileges of the user. Our model allows access privileges to be updated dynamically during a user's access session. How this change is going to affect user's authorization level depends on the digital library's policy. Similarly what information will be used in determining the trust level and how the information will be used, also depend on the digital library's policy.

Unlike other access control models, our framework keeps track of the behavior of a user. Access privileges are not assigned forever. The user may be denied access to the same resource for which she used to have access, if her trust level detoriates. If a user performs malicious task (e.g., forging credential), her trust level decreases and she gets a reduced set of privileges. In this case the user is not able to access previously accessible contents even if she presents necessary credentials. The digital library system allows the user to access those contents again after the necessary level of trust about the user is reached. Another advantage of this type of multi-level trust-based authorization is it provides finer control over specifying access privileges. The system can define as many trust levels as it wants and can assign each level to specific set of resources tied with a specific set of access privileges. The association of trust levels with set of contents defines the access control policy for the digital library system. The digital library system needs only compute and monitor the trust level of the user and the regulation of access is automatically achieved.

To achieve these goals we adapt the trust model we have proposed earlier [11]. Unlike binary trust models, trust in this new model has different degrees and is computed based on aspects of social interactions in addition to exchange of credentials rather than on just exchange of credentials. The idea is that each interaction that a user performs with the digital library system, the server discloses some portion of the resources. The digital library should have a comfort level with this disclosure. Before giving the access permission to the user for a particular category of content, the digital library needs to determine to what degree it trusts or distrusts the user to have access to those contents. We discuss how access privileges for a portion of the content can be controlled using trust levels. We propose mechanisms by which the system collects, stores, and manages information about the user. The information collected allows the system to compute a trust value for the user. The computed trust value acts as a confidence level for the digital library system for disclosing its resources to that user. Note that, we envision this system to be used in a membership based system that allows monitoring of user access and activities. Thus privacy issues related to this is not addressed in this work. The proposed scheme provides a flexible and powerful approach for the proper disclosure of contents. It offers the digital library system considerable control over how it wishes to disseminate its contents.

3 Content dependent access control in DLS

The DLS supports content dependent as well as content independent access control. The basic idea of content dependent access control in DLS is that a user's trust level determines which portion of content she can access with the allowed privileges on that portion. To do this DLS classifies its entire content into sub-categories.

Definition 1. *Each DLS object $o_k \in O$ (where O is the set of DLS objects, and o_k is the identity of the k^{th} object) has a set $\mathscr{P}_o^k = \{p_o^1, p_o^2, \ldots, p_o^k\}$ of properties that specifies the content characteristics of the object. These properties are drawn from a larger set of (potentially hierarchically organized) concepts called* object properties.

Some examples of object properties are “journal articles”, “magazines”, “free content”, “premium content”, “fiction”, “non-fiction”, “drama”, “comedy”, “adult”, “mp3-music” etc. The DLS defines a set *CC* of *content classes* for classifying its objects. A subset of properties from the set of object properties define a content class. Every DLS object is assigned to one or more content classes.

Definition 2. *Let* $prop(cc_i) = \{p_k, \ldots, p_n\}$ *be the object properties corresponding to the content class* cc_i*. An object* o_k *is classified to the content class* cc_i *if* $prop(cc_i) \subseteq \mathscr{P}_o^k$*.*

The function $OC : O \rightarrow \mathbb{P}(CC)$ maps an object to some subset of content classes. The function $OC^{-1} : CC \rightarrow \mathbb{P}(O)$ gives the objects that belong to any content class in *CC*.

Definition 3. *Two objects* o_i *and* o_j *belong to the same content class* cc_n *if and only if* $\mathscr{P}_o^i \cap \mathscr{P}_o^j \neq \varnothing$ *and* $\mathscr{P}_o^i \cap \mathscr{P}_o^j = \{p_{n_k} \ldots p_{n_m}\}$ *contains all the properties for* cc_n *i.e.,* $prop(cc_n) \subseteq \mathscr{P}_o^i \cap \mathscr{P}_o^j$*.*

The content classes are organized in a hierarchy. Figure 1 gives an example of content classes in the DLS system. We define the *content class hierarchy* as follows.

Definition 4. *Content class hierarchy* $CCH \subseteq CC \times CC$ *is a partial order on* CC*. For any two content classes* $(cc_1, cc_2) \in CCH$*, we say* cc_1 dominates cc_2*, denoted by* $cc_1 \succeq cc_2$ *if all the object properties that are in* cc_2 *are also in* cc_1*.*

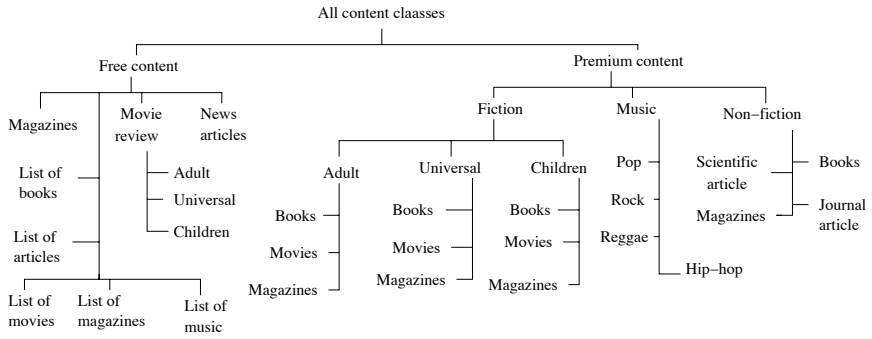

Fig. 1. Example of content class hierarchy in DLS system

Access privileges are associated with content classes. We formally define an access privilege as follows.

Definition 5. *An* access privilege, ap_i*, is specified as the tuple* $\langle$ action, sign, constraints, exceptions $\rangle$*, where*

1. action *is a set of possible operations on digital library objects such as browsing, authoring, retrieving, etc,*

2. sign $\in$ *(+,-), denotes whether the privilege is positive or negative,*
3. constraints *define a set of pre-conditions for the actions; the pre-conditions can include spatial and temporal conditions,*
4. exceptions *define conditions under which the constraints can be overridden.*

The access privilege "deny browsing if age less than 18 years unless supervised by adult" will be expressed as $\langle$ browse, -, age $<$ 18, adult-supervision $\rangle$. The set APC defines the set of all possible access privileges for the DLS. What type of access privileges would be associated with which content class depends on the *content class access policy* of the DLS.

Definition 6. *The* content class access policy *is a function* $CCSP : CC \rightarrow \mathbb{P}(APC)$ *that maps a content class in CC to a set of access privileges in APC. The inverse function* $CCSP^{-1}$ *defined as* $CCSP^{-1} : APC \rightarrow \mathbb{P}(CC)$ *maps an access privilege to a set of content classes.*

The set of access privileges corresponding to the content class cc_i is represented by ${cc_i}^{ap}$. Objects of the DLS are also associated with access privileges. Thus we define the *object access policy* as follows.

Definition 7. *The* object access policy *is a function* $OAP : O \rightarrow \mathbb{P}(APC)$ *that maps an object in O to a set of access privileges in APC. The inverse function* OAP^{-1} *defined as* $OAP^{-1} : APC \rightarrow \mathbb{P}(O)$ *maps an access privilege to a set of objects.*

In DLS, users get different access privileges to different resources on the basis of their 'trust-level' with DLS during access request. Before presenting the authorization framework, we would like to define what we mean by trust.

Definition 8. *Trust is defined to be the firm belief in the competence of an entity to act according to some specific rules within a specific context.*

Definition 9. *Distrust is defined as the firm belief in the competence of an entity to act contrary to some specific rules within a specified context.*

Although we define trust and distrust separately, we allow neutrality in the belief about competence of the entity. Neutrality represents a position where there is neither trust that the entity will act according to the specified rules nor distrust that the entity will act contrary to those rules.

Trust (distrust) is specified as a relationship between the DLS system – the truster that trusts the target entity – and a user (or an agent working on behalf of the user) – the trustee that is trusted. We use the following notation to specify a trust relationship – $(DLS \xrightarrow{c} U)^N_t$ where U is a specific user of DSL. This expression specifies *DLS*'s *normalized* trust on U at a given time t for a particular context c. The normalized trust relationship is obtained from the simple trust relationship – $(DLS \xrightarrow{c} U)_t$ – by combining the latter with a normalizing factor. This trust is always related to a particular context c.

Definition 10. *A* context c_i *of a trust relation in DLS is defined as a set of actions* $a_1, \ldots, a_n$ *from the set of all possible actions that can be defined on objects. The context is interpreted as the conjunction of all these actions, that is* $c_i \equiv a_1 \wedge \ldots \wedge a_n$.

Definition 11. *A trust context c_i* covers *another context c_j if $c_j \subseteq c_i$. A trust relation $(DLS \xrightarrow{c_i} U)_t^N$ is* useful *in context c_j if c_i covers c_j.*

If a trust relationship is *useful* in a context other than the one it was specified for, then the trust relationship can be used to make access control decisions for the different context. Next we introduce a concept called the *value* of a trust relationship. This is denoted by the expression $\mathbf{v}(DLS \xrightarrow{c} U)_t^N$ and is a number in $[-1,1] \cup \{\perp\}$ that is associated with the normalized trust relationship. A user is completely trusted (or distrusted) if the value of the trust relationship is 1 (-1). If the value is in the range (0,1) the user is *semi-trustworthy*; if the value is in the range (-1,0) the user is *semi-distrustworthy*. The 0 value represents trust neutrality that is, the user is neither trustworthy nor untrustworthy. The special symbol $\perp$ is used to denote the value when there is not enough information to decide about trust, distrust, or neutrality. The whole range of trust values are sub-divided into some non-overlapping intervals. Each interval represents a set of trust levels. We use the symbol $\mathscr{I}$ to represent a set of trust-intervals int_k with the properties: $\bigcup_k int_k = [-1,1] \cup \{\perp\}$ and $int_j \cap int_k = \varnothing,\ \forall j \neq k$. The function $TI : \mathbf{v}(DLS \xrightarrow{c} U)_t^N \rightarrow int_k$ maps a trust value to a trust interval.

Definition 12. *A trust-based access control policy of a digital library system, is defined as one of either $\langle CC, \mathscr{I}, A\rangle$ or $\langle O, \mathscr{I}, A\rangle$ or both where* CC *is the set of content-classes, $\mathscr{I}$ is a set of trust-intervals with each interval being a set of trust levels, and a* trust association function $A : CC \cup O \rightarrow \mathscr{I}$ *which defines the association between a content-class or an object and a trust-interval. Formally, the association is represented as:*

$$A(cc_k) = int_j \quad where\ \forall k,\ cc_k \in CC,\ and\ \forall j,\ int_j \in \mathscr{I}. \tag{1}$$

$$A(o_k) = int_j \quad where\ \forall k,\ o_k \in O,\ and\ \forall j,\ int_j \in \mathscr{I}. \tag{2}$$

This mapping actually defines the access control policy of the system. The policy specifies what trust-level allows a user to access a specific object or a set of objects. If a user's trust level is in the interval int_j, she can access any object belonging to the class cc_k with all the privileges tied to this class, provided no exception is defined on the access privilege. Decreasing the trust level beyond this interval int_j results in a change in access privileges of the user; the user may no longer have the same access rights for the same information. The system may also choose to tie special condition(s) (e.g., a mandatory credential) to allow access to a particular content-class cc_j, where $A(cc_j) = int_k$,. In this case, the user needs to have her trust level in int_k as well as has to satisfy the mandatory condition in order to have access to the content-class. Figure 2 gives the conceptual model of access control in the DLS.

4 Establishing trust relationship between DLS and a user U

To gain access to DLS resources, a user U first needs to register. The user signs in as a 'new user' and the system asks U to choose a 'username' and 'password'. Even if the user U chooses not to provide any information about herself (including name, address, phone number etc.), the registration is successful. The DLS builds a trust relationship

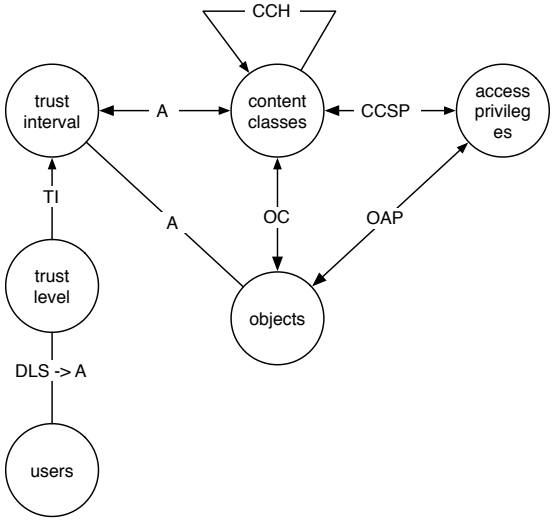

Fig. 2. DLS access control model

$(DLS \xrightarrow{c} U)_t^N$ with each registered user U. The underlying context c for the trust relationship is set to the most basic action that is possible as defined in DLS (log-in, for example). Depending partly on DLS's policy on registration information required, an initial trust level is set for the user. Typically it will be neutral. As the user continues to interact with DLS the trust level changes.

The vector trust model defines three different parameters that influences the computation of a trust level – *experience*, *knowledge* and *recommendation*.

Definition 13. *The* experience *of a truster about a trustee is defined as the cumulative effect of a number of events that occurred between the truster and the trustee over a specific period of time in the given context.*

DLS categorizes each experience as *trust-positive*, *trust-negative* or *trust-neutral* experience. A trust-positive experience increases trust degree whereas a trust-negative experience diminishes trust degree. A trust-neutral event contributes neither way.

Definition 14. *The* knowledge *of the truster regarding a trustee for a particular context is defined as a measure of the characteristic attributes or information of the trustee for which the truster can have some assertion to be truly related to the trustee.*

The trust value of DLS on a user can change because of some *knowledge* that the DLS possesses about the user. Information about the user may be obtained by the DLS in some earlier time for some purpose or, it may be a piece of information about the user for which the DLS can have a proof to be true. As with interactions, we have *trust-positive*, *trust-negative*, and *trust-neutral* knowledge.

Definition 15. *A* recommendation *about a trustee is defined as a measure of the subjective or objective judgment of a recommender about the trustee to the truster.*

It is important to note that the importance of the judgment of the third entity depends on how much the DLS trusts the third person's ability to judge others. As before we can have a *trust-positive*, *trust-negative*, and a *trust-neutral* recommendation. Finally, recommendations can be obtained by the DLS from more than one source and these together will contribute to the final trust relationship.

To compute a trust relationship we assume that each of these three factors is expressed in terms of a numeric value in the range $[-1,1]$ and a special value $\perp$. A negative value for the component is used to indicate the *trust-negative* type for the component, whereas a positive value for the component is used to indicate the *trust-positive* type of the component. A 0 (zero) value for the component indicates *trust-neutral*. To indicate a lack of value due to insufficient information for any component we use the special symbol $\perp$. Properties of $\perp$ are: If $\mathbb{R}$ is the set of real numbers, then (i) $a \cdot \perp = \perp \cdot a = \perp,\ \forall\ a \in \mathbb{R}$; (ii) $a + \perp = \perp + a = a,\ \forall\ a \in \mathbb{R}$; (iii) $\perp + \perp = \perp$ and $\perp \cdot \perp = \perp$. We now discuss how values will be assigned to each of these components.

Evaluation of knowledge The parameter "knowledge" is difficult to compute and is, to some extent, subjective. To begin with, the DLS must define its own criteria for gradation of information (or, properties) regarding any user. After the user U registers with DLS, the system asks for several specific information from U. The user can disclose those at once or she can choose to disclose them gradually at later times. For every piece of information that DLS receives from the user, a value between $[-1,1]$ is assigned. How the values are assigned, depends on the scheme and policy (called, *knowledge evaluation policy*) of the DLS. Also the DLS solely is responsible for assigning the relative weights to different attributes or information. At any time t, the average of those values gives the value of knowledge about U. If the DLS is aware of k attributes of the user, then knowledge of user U according to the DLS in context c is evaluated as ${}_{DLS}K_U^c = \frac{\sum_{i=1}^{k} v_i}{k}$, where $v_i \in [-1,1]\ \forall i = 1,2,\ldots,k$. User's personal as well as professional information constitute the 'knowledge'. For example the following can constitute 'knowledge' about a user U:

- Personal information: Name, Address, Home phone number, Work phone number, Cell number etc.
- Financial Account information: Credit card number, validity period, credit card security code, Bank name, Bank routing number, Checking account number, etc.
- Affiliation: Name of the organization, Branch location, Organization accreditation, Designation of U in the organization, Proofs/Certificates related to affiliation, Designation of certifying authority (like, manager, CEO, advisor, department-chair, dean-of-studies) etc.

It is possible that the DLS has insufficient information to assign a value to knowledge. For these types of cases, it assigns $\perp$ to the component. Note, ${}_{DLS}K_U^c = \perp$ is different from ${}_{DLS}K_U^c = 0$. Value 0 implies that after evaluating the information according to trust policy, the DLS's decision is neutral. But the value '$\perp$' implies "lack of information", that is there is not enough data to determine 'knowledge' about the user.

Evaluation of experience Most of the information that goes toward the forming the 'knowledge' of DLS about U in context c does not necessarily enhance or degrade the

system's trust on U. This is because all the above information are provided voluntarily by the user U. There is no guarantee that U discloses all information correctly. More useful, is perhaps, the interactions between the user and DLS. The user's behavior manifests in the form of *events*. We model experience in terms of the number of events encountered by the DLS regarding a user U in the context c within a specified period of time $[t_0, t_n]$. Like knowledge, an event can be trust-positive, trust-negative or, trust-neutral. If there are events that conforms to the knowledge that the system has gathered then these events will be termed trust-positive. Every successful verification of information or every successful transaction with U can be considered as a trust-positive event. If the events are contrary to the knowledge then they are trust-negative. Otherwise they are trust-neutral. In fact, negative outcome of a verification procedure or failure of verification of a piece of information results in a trust-negative event. Every time the user logs in, the system tries to verify the information about the user that is stored in the system. The user may accept all information as correct or can edit them. The system verifies the validity of those information. If verification fails or any anomaly is found, it is considered a negative event. Note that all information may not be verifiable at once. Results of those information have the impact on the next transaction. For that instance, user U's trust level is calculated on the basis of the current available results. Some examples of events are as follows. The list is not exhaustive.

- Every successful transaction is considered to be a positive event.
- Providing invalid e-mail id, wrong home address or, wrong contact numbers are considered as negative events. Correct informations are trust positive events.
- Providing wrong credit card or invalid credit card details is a negative event. Similarly, wrong checking account information (either false routing number or account number or combination of these results in a trust-negative event). Correct information results in a trust-positive event.
- Purchase request with stolen or forged credit card/account number is a negative event. Successful purchase is a positive event.
- Forging a credential is a negative event while providing a valid credential generates a positive event.
- Posting improper, objectionable, or irrelevant remarks through *review center* is considered to be negative events.

Events far back in time does not count as strongly as very recent events for computing trust values. Hence we introduce the concept of *experience policy*. It is defined as follows.

Definition 16. *An* experience policy *specifies a totally ordered set of non-overlapping time intervals together with a set of non-negative weights corresponding to each element in the set of time intervals.*

Recent intervals in the experience policy are given more weight than those far back. The whole time period $[t_0, t_n]$ is divided in such intervals and the DLS keeps a log of events occurring in these intervals.

If e_k^i denote the k^{th} event in the i^{th} interval, then we denote the value associated with e_k^i as v_k^i. This value is assigned according to relative importance of the event e_k^i.

$v_k^i \in [-10,0)$ if $e_k^i \in Q$, $v_k^i \in (0,10]$ if $e_k^i \in P$ and $v_k^i = 0$ if $e_k^i \in N$ where, P = set of all trust-positive events, Q = set of all trust-negative events and N = set of all trust-neutral events. The system assigns different weights to different events on a 10-point scale depending on the seriousness or effect of the event. For example, providing a wrong telephone number by a user may not be as serious offense as forging a credit card number. So the system assign two different negative values for these two trust-negative events.

The *incidents* IN_j, corresponding to the j^{th} time interval is the normalized sum of the values of all the events, trust-positive, trust-negative, or neutral for the time interval. The normalization is done in such a way that $IN_j \in [-1,1]$. If n_j is the number of events that occurred in the j^{th} time interval, then

$$IN_j = \begin{cases} \perp \quad \text{, if } \nexists\, e_k \in [t_{j-1}, t_j] \text{ for any } k \\ \frac{\sum_{k=1}^{n_j} v_k^j}{\sum_{k=1}^{n_j} |v_k^j|} \quad \text{, otherwise} \end{cases}$$

The *experience* of DLS with regards to U in the context c is given by, ${}_{DLS}\mathrm{E}_U^c = \sum_{i=1}^{n} w_i IN_i$, where, $w_i \in [0,1]$ is a non-negative weight assigned to i^{th} interval.

Evaluation of recommendation In our modified trust model [12] recommendation is evaluated on the basis of a *recommendation value* returned by a recommender to the truster about the trustee. A truster will, most likely, have a trust relationship with the recommender, which is different from a trust relationship between truster and trustee and is formulated as specified by the trust model in [12]. The context of this trust relationship will be to act "reliably to provide a service (recommendation, in this case)" and it can be established parallelly or prior to the establishment of current trust relationship. This trust relationship will affect the score of the recommendation provided by the recommender. Therefore, *recommendation* of the DLS with regards to a user U for a context c is given by ${}_{\Psi}R_U^c = \frac{\sum_{j=1}^{n} (\mathbf{v}(DLS \xrightarrow{rec} j)_t^N) \cdot V_j}{\sum_{j=1}^{n} (\mathbf{v}(DLS \xrightarrow{rec} j)_t^N)}$, where Ψ is a group of n recommenders, $\mathbf{v}(DLS \xrightarrow{rec} j)_t^N)$ = trust-value of j^{th} recommender and $V_j = j^{th}$ recommender's recommendation value about the user U.

Recommendation plays a role in the evaluation of trust level of a user when the DLS is a member of a consortium of digital libraries. In such cases, a member of the consortium should be able to provide information about certifiable behavior at resource pool boundaries. Also recommendations play a role in the process of delegation. Delegations are task oriented relationships that recur within a community. A delegation is a set of privileges required to accomplish related task.

We next observe that given the same set of values for the factors that influence trust, two different DLS may come up with two different trust values for the same user. During evaluation of a trust value, one DLS may assign different weights to the different factors that influence trust. For example, the DLS may choose to emphasize more on its experience about the user than some knowledge about the user. Which particular component of the trust vector needs to be emphasized more than other is a matter of the *normalization policy* of the DLS.

Definition 17. *The* normalization policy *for a trust relationship* $(DLS \xrightarrow{c} U)_t$ *is a vector of same dimension as of* $(DLS \xrightarrow{c} U)_t$*; the components are weights in the range* $[0,1]$ *with their sum being equal to 1 and assigned to experience, knowledge, and recommendation components of* $(DLS \xrightarrow{c} U)_t$.

We use the notation $(DLS \xrightarrow{c} U)_t^N$, called *normalized* trust relationship to specify a trust relationship between the DLS and the user U. This relationship is obtained from the simple trust relationship after combining the former with the normalizing policy. It is derived as, $(DLS \xrightarrow{c} U)_t^N = \mathbf{W} \odot (DLS \xrightarrow{c} U)_t$. The $\odot$ operator represents the normalization operator. Let $(DLS \xrightarrow{c} U)_t = [{}_{DLS}E_U^c,\ {}_{DLS}K_U^c,\ \Psi R_U^c]$ be a trust vector such that ${}_{DLS}E_U^c,\ {}_{DLS}K_U^c,\ \Psi R_U^c \in\ [-1,1] \cup \{\perp\}$. Let also $\mathbf{W} = [W_E, W_K, W_R]$ be the corresponding trust policy vector such that $W_E + W_K + W_R = 1$ and $W_E, W_K, W_R \in [0,1]$. The $\odot$ operator generates the normalized trust relationship as $(DLS \xrightarrow{c} U)_t^N = \mathbf{W} \odot (DLS \xrightarrow{c} U)_t = [W_E,\ W_K,\ W_R] \odot [{}_{LS}E_U^c,\ {}_{DLS}K_U^c,\ \Psi R_U^c] = [W_E \cdot {}_{DLS}E_U^c,\ W_K \cdot {}_{DLS}K_U^c,\ W_R \cdot \Psi R_U^c] = [{}_{DLS}\hat{E}_U^c,\ {}_{DLS}\hat{K}_U^c,\ \Psi \hat{R}_U^c]$.

We next introduce a concept called the *value* of a trust relationship. This is denoted by the expression $\mathbf{v}(DLS \xrightarrow{c} U)_t^N$ and is a number in $[-1,1] \cup \{\perp\}$ that is associated with the normalized trust relationship $(DLS \xrightarrow{c} U)_t^N$. It is defined as $\mathbf{v}(DLS \xrightarrow{c} B)_t^N = {}_{DLS}\hat{E}_U^c + {}_{DLS}\hat{K}_U^c + \Psi \hat{R}_U^c$.

Trust (and distrust) changes over time. We claim that even if the underlying parameters do not change between times t_i and t_n at which a trust relationship is being evaluated, the trust relationship will change. To model this *trust dynamics* (i.e., the change of trust over time) we observe that the general tendency is to forget about past happenings. This leads us to argue that trust (and distrust) tends toward neutrality as time increases. Initially, the value does not change much; after a certain period the change is more rapid; finally the change becomes more stable as the value approaches the neutral (value = 0) level. The idea is captured by the equation $\mathbf{v}(T_{t_n}) = \mathbf{v}(T_{t_i}) e^{-(\mathbf{v}(T_{t_i})\Delta t)^{2k}}$ where, $\mathbf{v}(T_{t_i})$, be the value of a trust relationship, T_{t_i}, at time t_i and $\mathbf{v}(T_{t_n})$ be the decayed value of the same at time t_n. The effect of time is captured by the parameter k which is determined by the truster's *dynamic policy* regarding the trustee in context c.

The trust model also has a method to obtain a vector of same dimension as of $(DLS \xrightarrow{c} U)_t^N$ from this value $\mathbf{v}(T_{t_n})$. The current normalized vector together with this time-affected vector are combined according to their relative importance. Relative importance is determined by the DLS's *history_weight policy* which specifies two values α and β in $[0,1]$ (where, $\alpha + \beta = 1$) as weights to current vector and the vector obtained from previous trust value. The new vector thus obtained gives the actual normalized trust vector at time t for the trust relationship between the DLS and a user U in context c. This is represented by the following equation.

$$(DLS \xrightarrow{c} U)_{t_n}^N = \begin{cases} [{}_{DLS}\hat{E}_U^c, {}_{DLS}\hat{K}_U^c, \Psi \hat{R}_U^c] & \text{if } t_n = 0 \\ [\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}] & \text{if } t_n \neq 0 \text{ and } {}_{DLS}\hat{E}_U^c = {}_{DLS}\hat{K}_U^c = \Psi \hat{R}_U^c = \perp \\ \alpha \cdot [{}_{DLS}\hat{E}_U^c, {}_{DLS}\hat{K}_U^c, \Psi \hat{R}_U^c] + \beta \cdot [\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}] & \\ \qquad \text{if } t_n \neq 0 \text{ and at least one of } {}_{DLS}\hat{E}_U^c, {}_{DLS}\hat{K}_U^c, \Psi \hat{R}_U^c \neq \perp \end{cases} \tag{3}$$

where $[\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}]$ is the time-effected vector and $\mathbf{v}(\hat{T}) = \mathbf{v}(T_{t_n})$.

Note, for DLS, it may not be reasonable to decrease (increase) the trust (distrust) level of a user at a faster rate. Because that will result in reduction (enhancement) in her access privileges with duration of time. For example, let a user with trust value, say 0.4 stop interacting with the DLS. At this point she is cleared to say, cc_i. After a long time, the user again interacts with DLS and finds her trust level goes down to, say 0.25 and she can not access all of cc_i anymore and is restricted to a content class, say cc_j where $cc_i \succeq cc_j$. This issue can be solved in one or both of the following ways: (i) Choose the value for k in the *dynamic policy* to ensure a very slow decay in trust values, or (ii) Assign a very small value for β in *history-weight-policy* thereby putting very less importance on the time-affected vector.

Sometimes it may not be possible to obtain a non null value for any of the trust parameters. In such cases the DLS system tries to determine if it is aware of a trust relationship for the same user in a related context that *covers* the current context. Recall from section 3 that if such a trust relationship exist it is *useful* in the given context. In such cases, the trust level established for the related context is used by the DLS system to determine access.

5 Architecture of the DLS access control module

The high level system architecture of the DLS access control module consists of the components as shown in figure 3. The two main components are *authorization controller* and *trust engine*. The authorization controller interacts with the *content-server* and the trust engine.

Access specification module This module defines the classification of resources into content classes and objects. That is, the module defines CC and $\mathscr{P}_o$s for each object. It also defines the content class hierarchy CCH. Types of access privileges that are to be tied to each content class or object is also specified here. This module is also responsible for specifying any special constraint (other than trust level) or an exception that has to be satisfied to allow access to a content class or to an object. In other words, the module is responsible for definitionning the functions $OC, CCSP, CCSP^{-1}, OAP$, and OAP^{-1}.

Access control module This module is responsible to classify trust levels into different sub-intervals i.e., defines the set $\mathscr{I}$. It also defines the *association function* A.

Access analysis module This module has a user database. It receives the user's information and user's request through a *Service module*. It passes user information to trust engine and receives trust related result from it. Consulting with the access specification module and access policy module, it takes the decision about the specific request of the user and pass it to the service module. It also verifies user information and checks for special constraints and exceptions.

Service module The service module is an independent module outside the authorization controller as well as trust engine. Its job is to interact with the user through an interface. It collects user input and sends it to access analysis module of authorization controller. According to the decision it receives from access analysis module

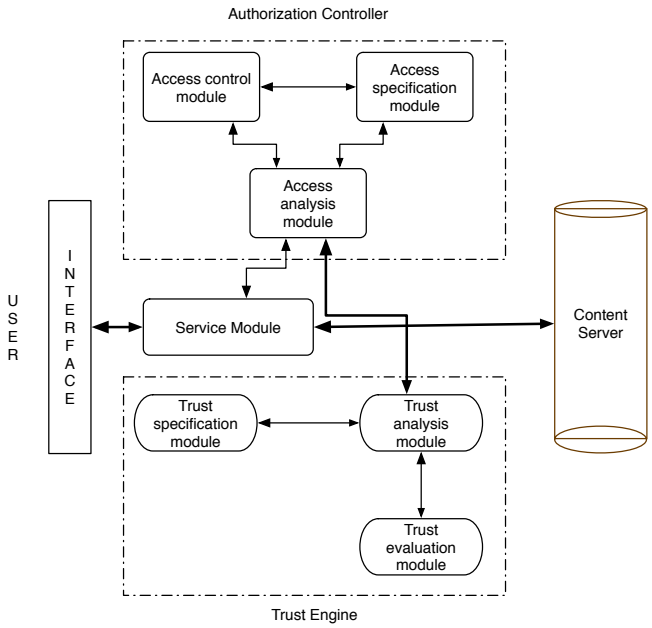

Fig. 3. Architecture of DLS digital library system

about the request it interacts with the content-server and provides the requested service to the user.

Trust specification module It is responsible for definitionning and managing trust relationships. It creates database entries corresponding to a specific user when a new trust relationship is established. It codifies general trust evaluation policies (for example policy for trust dynamics). The specification module conveys this information to the analysis module and the evaluation module as and when needed.

Trust analysis module The analysis module processes trust queries from access analysis module of authorization controller. It obtains trust vectors from the evaluation module.

Trust Evaluation module This module retrieves information about experience, knowledge, and recommendation from the database and also other pertinent information from the trust specification module to compute trust vector according to the theory specified in this paper. It also stores back resulting values in the database kept in trust specification module.

6 Conclusion and future work

In this work we develop a flexible access control framework for digital library systems. The framework is based on the vector trust model that we had proposed earlier. We show how a digital library system can specify access control policies by associating

a set of objects and access privileges with a set of trust levels. The underlying trust model evaluates a user's trust level with respect to the system using knowledge about the user. The system also considers its experience with the user to evaluate trust. This is a major contribution of the scheme where history of user's behavior is used to control her access clearance. A lot of work, however, still remains to be done. The scheme is proposed with a server-side approach. Extending the underlying trust model to a mutual trust negotiation model, we plan to design a two-way scheme to include client-side access control. Designing such a scheme would help to solve the issues like disclosure of policies, especially privacy protection policies, in online transactions. We also plan to develop efficient methods of interaction between an authorization controller and a trust engine.

References

1. Bertino, E., Ferrari, E., Perego, A.: Max: An access control system for digital libraries and the web. In: Proceedings of the 26th IEEE International Computer Software and Applications Conference, Oxford, UK (2002)
2. H.M.Gladney: Access Control for Large Collections. ACM Transactions on Information Systems **15**(2) (1997) 154–194
3. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA (1996)
4. Blaze, M., Feigenbaum, J., Ioannidia, J.: The KeyNote Trust Management System Version 2. Internet Society, Network Working Group. RFC 2704 (1999)
5. Li, N., Mitchell, J.: Datalog with Constraints: A Foundation for Trust-management Languages. In: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages, New Orleans, Louisiana (2003)
6. Winslett, M., Ching, N., Jones, V., Slepchin, I.: Assuring security and privacy for digital library transactions on the Web: client and server security policies. In: Proceedings of the IEEE international forum on Research and Technology Advances in Digital Libraries, Washington, DC, USA (1997) 140–151
7. Skogsrud, H., Benatallah, B., , Casati, F.: A Trust Negotiation System for Digital Library Web Services. Journal of Digital Libraries, Special Issue on Security **4**(3) (2004)
8. Ryutov, T., Zhou, L., Neuman, C., Leithead, T., Seamons, K.: Adaptive Trust Negotiation and Access Control. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden (2005)
9. Adam, N.R., Atluri, V., Bertino, E., Ferrari, E.: A Content-Based Authorization Model for Digital Libraries. IEEE Transactions on Knowledge and Data Engineering **14**(2) (2002) 296–315
10. Bonatti, P., Samarati, P.: Regulating Service Access and Information Release on the Web. In: Proceedings of the 7th ACM COnference on Computer and Communication Security, Athens, Greece, ACM Press (2000) 134–143
11. Ray, I., Chakraborty, S.: A Vector Model of Trust for Developing Trustworthy Systems. In: Proceedings of the 9th European Symposium of Research in Computer Security (ESORICS 2004). Volume 3193 of Lecture Notes in Computer Science., Sophia Antipolis, France, Springer-Verlag (2004) 260–275
12. Ray, I., Chakraborty, S., Ray, I.: VTrust: A Trust Management System Based on a Vector Model of Trust. In Jajodia, S., Mazumdar, C., eds.: Proceedings of 1st International Conference on Information Systems Security (ICISS 2005). Volume 3803 of Lecture Notes in Computer Science., Kolkata, India, Springer-Verlag GmbH (2005) 91–105