PEI Models for Scalable, Usable and High-Assurance Information Sharing

Ram Krishnan
George Mason University
Fairfax, VA, USA
rkrishna@gmu.edu

Ravi Sandhu
George Mason University and
TriCipher Inc.
Fairfax, VA, USA
sandhu@gmu.edu

Kumar Ranganathan
Intel System Research Center
Bangalore, India
kumar.ranganathan@intel.com

ABSTRACT

Secure Information Sharing (SIS) or "share but protect" is a challenging and elusive problem both because of its broad scope and complexity ranging right from conception (objective and policy) to culmination (implementation). In this paper, we consider how to solve SIS challenges with three main and conflicting objectives: *scalability*, *usability* and *high-assurance*. In the context of SIS, high-assurance requires strong controls on the client. It is widely accepted that such controls cannot be entirely software-based. In this regard, we consider solutions based on commercially emerging hardware-rooted Trusted Computing Technology. For SIS, we argue super-distribution ("protect once and access wherever authorized") and off-line access are necessary to achieve scalability and usability. As we will see, although a Trusted Platform Module [1] (TPM) provides a range of powerful functionalities, it does not enable true super-distribution in any obvious manner. We therefore limit super-distribution to occur within a group. A group is an abstract set of TPM-enabled machines. For simplicity, we assume all content that are distributed to be read-only. Drilling down, we propose concrete Policy, Enforcement and Implementation (PEI) models for SIS within a group (group-based SIS or g-SIS). In the policy layer, we develop a framework for specifying subject and object group membership. In the enforcement layer, we explore ways to approximate instant and preemptive revocation of group members to support off-line access. We use the UCON [9] model to formally specify the policy and enforcement models. In the implementation layer, we outline protocols using Trusted Computing Technology [1] that would realize our policy model and thereby our objectives. We also demonstrate the value of this layered approach by showing how our enforcement and implementation models can easily accommodate enhancements in the policy model.

1. INTRODUCTION

Sharing information (objects) *while* protecting it is one of the earliest problems to be recognized in computer security, and yet remains a challenging problem to solve. Although SIS sounds like an oxymoron, its application scenarios are endless ranging from revenue centric retail Digital Rights Management (DRM) and sensitivity centric intellectual property to national security centric secret property. Classic access control models are either inherently weak or don't even address this problem domain. The Discretionary Access Control model or DAC (like the access matrix) as discussed in [5, 7, 2, 8] is fundamentally limited in that they control access only to original objects but not to copies. If objects could be read, one can read and create a copy of this object. Objects are protected up to the point when read access is granted to a subject. From then on, the owner has no control on his object. Mandatory Access Control models or MAC (like Bell-Lapadula) as discussed in [3, 4, 11] address information flow but are too rigid for fine-grained access control and falling back to DAC for fine-grained access control as the Orange Book suggests [2] is pointless.

In this paper, we consider how to solve the Secure Information Sharing (SIS) problem with three main and conflicting objectives: *scalability*, *usability* and *high-assurance*. In the context of SIS, high-assurance requires strong controls on the client. It is widely accepted that such controls cannot be entirely software-based. In this regard, we consider solutions based on commercially emerging hardware-rooted Trusted Computing Technology. For SIS, we argue super-distribution ("protect once and access wherever authorized") and off-line access are necessary to achieve scalability and usability. As we will see, although a Trusted Platform Module [1] (TPM) provides a range of powerful functionalities, it does not enable true super-distribution in any obvious manner. We therefore limit super-distribution to occur within a group. A group is an abstract set of TPM-enabled machines. For simplicity, we assume all content that are distributed to be read-only.

In this paper, we use the recently proposed UCON model [9] to formally specify the policy and enforcement frameworks for the secure information sharing problem. As discussed earlier, our conflicting objectives forces us to consider super-distribution within a group. We define group to be a set of entities that share a common property. This definition of a group is abstract and powerful enough to accommodate a wide range of real world scenarios. For instance, a group could be tightly-knit that is task oriented (E.g.: project groups) or loosely-knit with similar interests (E.g.: discussion forums) or an ecosystem (E.g.: a network of competi-

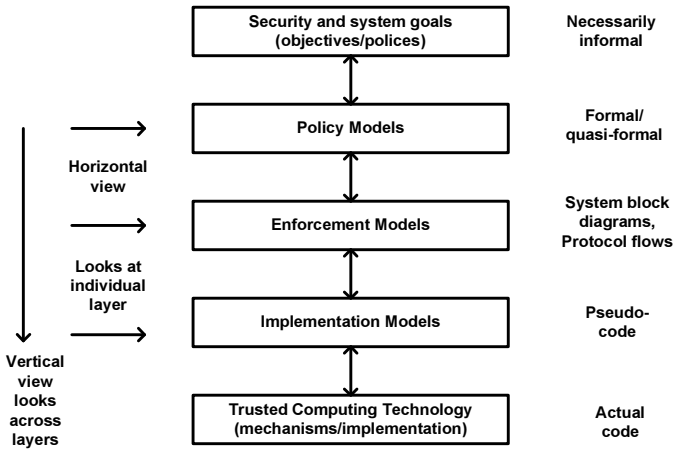

Figure 1: The PEI Models Framework.

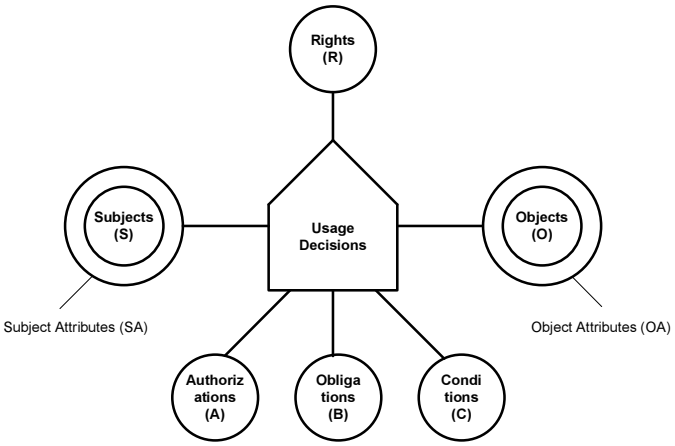

Figure 2: UCON Components.

tors, suppliers and customers working towards a common goal that benefits everyone). On the other hand, contextual properties like location, proximity, etc. could also form an ad-hoc group.

We explore policy, enforcement and implementation models for the group-based secure information sharing problem. To help separate and answer what we want to solve from the how we want to solve, many frameworks have been proposed in the past including the policy-mechanism separation principle, OM-AM framework [12], etc. We use the more recent PEI framework [14] that suggests additional separation between enforcement and implementation models which we believe is absolutely necessary to address a complex problem such as SIS.

The remainder of this paper is organized as follows. In section 2 we give a brief overview of the PEI framework, the UCON model and Trusted Computing Technology. In section 3, we specify a policy framework for the g-SIS problem and differentiate it from policies applicable to related problem domains. We provide motivating use-cases that require such a rich policy framework. In section 4, we formally specify policies for g-SIS using the UCON model, under the idealized assumption that instant and preemptive revocation can take place. In section 5, we identify two enforcement models to accommodate the policy requirements. These enforcement models necessarily have concrete approximations to the idealized instant and preemptive revocation assumptions of the policy model. We identify trade-offs and develop a usable model to reconcile these trade-offs. Again, we formally specify the usable model using UCON and demonstrate that a model as rich and powerful as UCON is required to model the secure information sharing problem. In section 6, we argue that we need the hardware rooted trust of trusted computing technology to solve this problem at the mechanism level with high-assurance. We outline protocols for our enforcement model. We conclude the paper in section 7.

2. BACKGROUND

We give a brief overview of the PEI framework, UCON model and Trusted Computing Technology. The overviews are necessarily short and we direct interested readers to the references for further details.

2.1 PEI Framework

Many approaches have been proposed in the past to help solve security problems including the traditional approach of separating policy from mechanism, OM-AM [12], etc. We use the more recent PEI framework [14] (Figure 1) to analyze our g-SIS problem and synthesize suitable solutions. At the highest level of the framework, we specify the overall goals or objectives of the security system. At this level the discussion and analysis is necessarily informal. In the Policy Models (P) level, we formally specify our objectives using an appropriate model. Many models exist including the traditional DAC and MAC, RBAC96 [13], and the more recent UCON. In the Enforcement Model (E) level, we specify system architecture to address the "how" question. In the Implementation Model (I) level, we propose concrete protocol flows and address specific issues left open in the Enforcement layer.The final layer makes system dependant decisions like technologies to use and produces actual code.

It is important to understand that in PEI the relationship between adjacent layers is many to many. A policy model could have multiple enforcement models and vice versa. Similarly, an enforcement model could have multiple implementation models and vice versa. Thus a horizontal view helps to explore various means to realize the objectives at a particular layer, and the vertical view helps to provide a specific solution.

2.2 UCON model

The UCON model [9] for usage control is a new foundation of access control which combines traditional authorization with obligations and conditions. It also provides mutability of attributes and continuity of decisions. Figures 2 and 3 give the high level overview of UCON. Usage decisions are based on subject and object attributes, authorizations, obligations and conditions. Authorizations are predicates based on subject and/or object attributes, such as role name, security classification or clearance, credit amount, etc. Obligations are actions that a subject needs to perform in order to allow access. For example, a subject should *agree* to a license

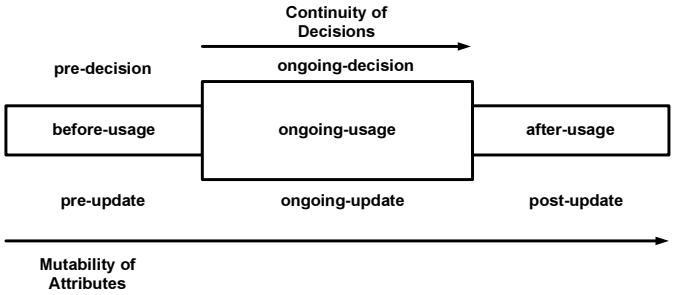

Figure 3: Continuity & Mutability Properties of UCON.

agreement before accessing an object. Conditions are used to capture system states and dependencies that are not dependent on subject and object attributes but rather on the state of the system. For example, a subject will be granted access to an object if the *server's load is less than a threshold value*. Access decisions are not only made at initial request but can also be continued throughout access time. Thus we could have pre, on-going and post access decisions, obligations and conditions. The presence of ongoing decisions is called the continuity of UCON. Another important property of UCON is attribute mutability. Mutability means that one or more subject or object attribute values can be updated as the results of an access. Along with the three phases, there are three kinds of updates: pre-updates, ongoing updates, and post-updates.

2.3 Trusted Computing

Trusted Computing Technology is an industry standard proposed by the Trusted Computing Group (TCG) [1]. It is widely accepted that software only mechanisms cannot provide high assurance. This motivated TCG to provide a root of trust at the hardware level through the Trusted Platform Module (TPM). The technology has evolved to a great degree now and we only provide a very brief overview here. The TPM mainly offers three novel features: Trusted Storage for keys, Trusted Capabilities and Platform Configuration Registers (PCR's). Trusted Storage for keys is provided by encrypting user's keys with chain of keys. The root key at the top of the chain is stored within the TPM and is not accessible outside the TPM. Trusted Capabilities are capabilities exposed by the TPM that are guaranteed to be trustworthy. Users cannot modify the behavior of these capabilities. PCRs are hardware registers in the TPM that are used to store integrity metrics (hash values) of software (e.g.: boot chain). Trusted Capabilities and PCRs provide some powerful functionalities. For example, Seal is a capability that encrypts and binds some data to a specific PCR value. This data can be accessed (Un-Sealed) by authorized entities only when PCR value at unseal time matches with the specified PCR value at seal time. Using this feature, one can make sure that a data blob such as a key would be available to authorized entities only when the platform (e.g.: every piece of software involved in the boot cycle up to the kernel) is in a trustworthy state (which is implied by the PCR value). There are many capabilities and features provided by TPM and abundant materials have been published by the TCG.

3. POLICY FRAMEWORK

From here on we use the word object to refer to information of any form (e.g. documents, voice data, etc.) that belongs to the group and inter-changeably use the words object and document. We use the word user to refer to any entity (machine, human, programs, etc.). We use the word member to refer to a user who is enrolled into a group. Before we delve into formally specifying the policies, we clearly state the objectives.

3.1 Objectives

We specify the following objectives for the g-SIS problem studied in this paper.

1. The objects are read-only. In the example of documents, we assume that documents can only be read and cannot be modified. If modified, the new document will not belong to the group unless explicitly added as a 'new' object again.
2. Objects are obtained via super-distribution. Super-distribution means that the contents of a protected (and therefore encrypted) object can be accessed anywhere by an authorized user. It implies that there is no difference between the encryption for one user versus another. In other words the encryption is done in a universal manner for all users rather than customized for each individual user.
3. We want to impose policies on these objects and hence need client side access control.
4. We want to enable off-line access. That is, the member need not be connected to a server while accessing the object.
5. We assume an admin who owns the group. The admin can add and remove members from the group. We do not care how an admin is appointed. The admin may or may not be a member of the group.

We now digress briefly to compare g-SIS with a related problem –broadcast/multicast encryption. Member management in g-SIS scenario sharply differs from Secure Internet Multicast. In multicast, as users join and leave a group, remaining members go through a re-key process thereby refreshing the group key [10]. However, for secure information sharing, such a requirement is extremely un-friendly because members need not be always connected to a server to access the objects. Thus, continuing our list of objectives, we have the following.

6. When a user joins or leaves the group, remaining members should not be affected. In other words, join and leave operations should be completely oblivious to other members.
7. Objective 6 requires that current members should not be forced to be online and go through a re-key process.[1]

[1]Members should not be asked to re-key or contact a server to get a new key. Note that re-keying is not an efficient solution in SIS as the member needs to keep track of which docu-

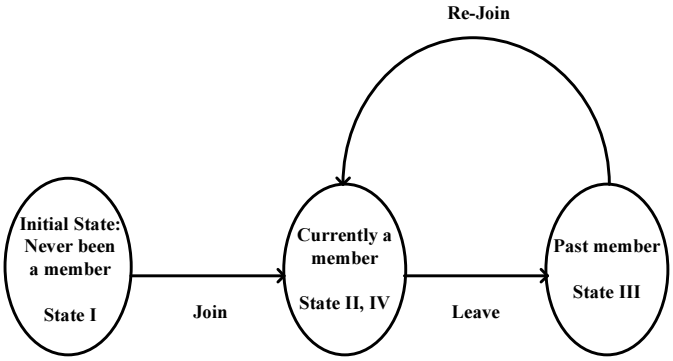

Figure 4: Various states of a subject in a group.

8. In secure multicast, we only worry about forward and backward secrecy [10]. Forward-secrecy means that if a user leaves a group, he should not be able to read group data that are created in the future. Backward-secrecy means that when a new user joins a group, he should not be able to read data created in the past. However, SIS is not limited to forward and backward secrecy of information. Flexible membership policies are to be enforced. When a new user joins the group, whether he can access any group documents created prior to his membership is policy-dependant. Any group document created after he joins the group are accessible. When a member leaves the group, whether he can continue to access all documents that he possesses is policy-dependant. However, he cannot access any documents exchanged in the future.

9. We use Trusted Computing Technology for high-assurance of protection of information and enforcement of policies within a group.

3.2 Member States

Figure 4 shows various states of a member in a group. We identify 4 states. In the initial state (state I), the user has never been a member of the group. In state II, he is a current member of the group. In state III, he left the group and is a past member. Note that a past member in state III can reach state II again by re-joining the group (state IV). Having classified the membership scope, we can define access policies for each state.

In State I, the access policy is straight-forward: the user has no access to any group documents.

In State II (current member), access could be allowed only to current documents or both current and past documents (documents that were created before a member joined a group). For each of these two access policy in state II, we can have rate-limited access (number of accesses allowed per unit time) or usage-limited access. These usage restrictions can be either used for refreshing membership status or for access throttling purposes.

In State III, many kinds of policies could be applied: 1. A past member may lose access to all documents. 2. A past member may access any document created during his membership time period. 3. A past member may be allowed access to the documents he accessed during his membership (the ones local to his machine). On multiple join and leave, accesses could accumulate. 4. A past member can only access the documents he accessed during his membership (the ones local to his machine). But on multiple join and leave, access is allowed only to documents acquired during his latest membership.

For State IV (member rejoin), access polices for members could get complicated. For example, when a member first left the group, he might have been denied access to all (both past and current) documents (policy state III. 1). But when he re-joins, if one allows access to all past documents (policy state II.2), in a sense it contradicts with the policy during his initial membership. These kinds of issues could get complicated over multiple re-join operations and needs to be stated with extra-care. To keep it simple, we suggest having two re-join policies to choose from: no rejoin allowed or when a past member re-joins he will join the group as a new member.

3.3 Motivating Use-cases

We discuss a few interesting use-cases for the rich set of access policies that we have identified so far. While there are a multitude of scenarios that can be imagined, it is not our goal to capture each of those. Rather we provide a few use-cases to justify some of the policies discussed earlier. For simplicity, we assume that no re-join is allowed in the following discussion. We consider 4 cases and corresponding use cases for each of these below.

Case 1. Suppose current members (in state II) can access only current documents and past members (in state III) lose access to all documents. We have the following use case.

Memory-less Collaboration: Many universities and corporations allow access to their content and share as long as one is within their network. Once the user leaves the network, the user loses access to the content.

Case 2. Suppose current members (in State I) can access only current documents and past members (in State III) can access documents created during his membership period. We have the following use case.

Collaborative Computing: A financial institution could recruit a software-consulting firm to provide software solutions. This forms a short-lived group. The incoming group members (from the software firm) cannot access any older documents. When they finish the project and leave the group, they can continue to have access to the documents exchanged during their membership in order to add to their profile. This is dependant on financial institution's policy.

Case 3. Suppose current members can access current documents and also the documents created before their member-

ment was encrypted with which key. As users join and leave a group, the remaining members will need to go through a re-key process resulting in encrypting documents with different keys along the time line. One cannot discard the old key (as done in multicast) as disseminated documents encrypted with the old key continue to persist.

ship period and past members lose access to all documents. We have the following use cases.

Employee management: An employee joining a company can access all the current documents. When an employee quits, he loses access to all the sensitive documents he had access to.

Government projects/contracts: DoD contracts have a multi-tiered structure. A member of a contracting company may be authorized to access certain set of documents only for the duration of the project –once the project is over, the contractor's right to use the document is automatically voided.

Supply Chain: In a supply chain situation, there are lots of partners and lots of suppliers who will send quotes for a given proposal. They need to have access to the proposal and related content. But once the quote/response is submitted, their membership context for that particular or group of proposals ceases and they shouldn't have access to any of the older content that they had access to.

Case 4. Suppose current members can access current documents and also the documents created before their membership period and also past members can continue to access documents created before his leave time. We have the following use cases.

Intra/Inter corporate discussion groups: Documents (program code, emails, reports, etc.) are exchanged amongst group members formed within a company (or between companies). When a new member joins a discussion group, he should be able to access all the earlier discussions and can continue to access them even after leaving the group (E.g. Email). However this is mostly dependant on the kind of documents exchanged and the company's policy.

Collaborative product development: In the case of several automobile models, there are product twins –models from the same company that resemble each other, except for the division's brand name and price tag. It's less expensive for auto manufacturers to produce parts in bulk and share them than to build separate components for their various brand names. Ford and Mercury, for example, are under the same corporate umbrella, and their Taurus and Sable four-doors are among the company's twins. In such instances, there could be either a loose collaboration (e.g. shared design team, parts ordering/manufacturing but different factories) or a tight collaboration (e.g. joint manufacturing of two different models). In either case, the members from different parties join hands and share documents actively. They will need access to both old documents and current documents. Even after the collaboration period, they will need access to the old documents for further refinement and production.

3.4 Object States

Symmetric to states that a member goes through, we have a notion of membership of objects or documents. Figure 5 shows various states of a document in a group. We now identify policies for document membership.

For state I, members have no access to documents that are not part of the group. For state II (current document), current documents can be accessed by current members. For State III (past document) we have the following possibilities: 1. No one can access past document. 2. Any member can access. 3. Any one including non-members can access. For State IV (readded documents), there are several possibilities: 1. Documents cannot be re-added. 2. When a document is re-added, it will be treated as a new document in the group. 3. When a document is re-added, it will treated as an old document (its history will be preserved).

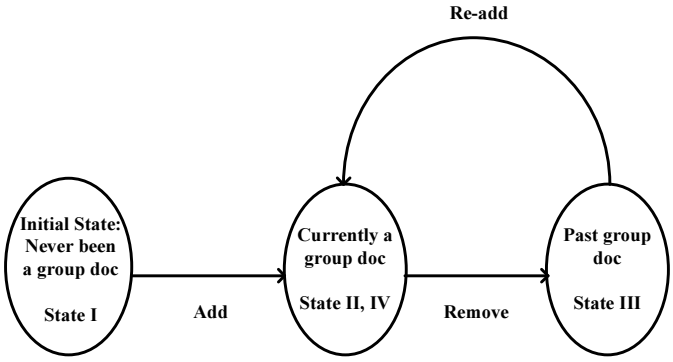

Figure 5: Various states of an object in a group.

It is clear that specifying membership policies for members and documents could get complicated. But we believe we have motivated the reader to appreciate the richness of the policy framework for the g-SIS problem, and therefore the necessity for a flexible enforcement and implementation framework to accommodate these variations and switch between them as necessary.

4. IDEAL POLICIES

In the earlier section, we provided an overview of the policy framework for group membership. The policies identified there define the characteristic of a group and we call them as group-level policies or meta-policies. In order to formally state these policies, we arbitrarily pick the following meta-policy from each state of the member and specify them under the UCON model:

For members, we choose the group-level policy: *During membership, members cannot access documents created before join time and can only access current documents. After leaving the group, members can continue to access documents created during his membership. Members cannot re-join.*

For group documents, we choose the group-level policy: *Documents currently in group can be accessed by current group members only. Past documents cannot be accessed by any member. Documents cannot be re-added into the group.*

For clarity, in this paper we only consider single group membership. In addition to meta-policies, we have some key policies to consider. *Ideally*, we would want the following policies for g-SIS:

1. A Team Representative (TR, the group admin) adds and removes member and documents into/from the group.

2. Instant and preemptive revocation of a member from a group by the TR takes effect. That is if the member is removed from the group he immediately moves into state III (instant revocation) and the policy with respect to documents currently being accessed by this member are immediately and preemptively adjusted into the state III policy (preemptive revocation).[2]

3. Instant and preemptive revocation of a document from a group by the TR takes effect, similar to the instant and preemptive revocation of a member from a group above.

4.1 UCON model conventions and notations

We specify a few conventions that are used throughout this model.

1. The Subjects and Objects in our system have the same meaning as that in the HRU model [6]. Briefly, Subjects (S) have an entry in both the row and column in the access matrix and Objects (O) have only a column entry. Thus, $S \cap O = \phi$.

2. The UCON attribute functions (discussed later) are either partial or total. In a partial function, attributes may not be defined for every subject and/or object in the system. In a total function, attributes are defined for every subject and/or object in the system.

3. NULL is a special value indicating 'undefined' and it is not a part of the domain of any UCON attribute.

4. Every subject and object in the system belongs to a certain *type* which is assumed to be pre-defined.

4.2 UCON Attributes

In this section, we define the UCON attributes that are used throughout the model. TS is the set of all time-stamps. N is a set of all integers.

$ty : S \cup O \rightarrow \{doc, member, TR\}$ This is an attribute function that maps the subjects and objects to one of the following types:

- *doc* group document type
- *member* a regular group member
- TR the Team Representative (group admin) of the group
- CC the Control Center (discussed later)

This is a total function and is immutable for both subject and objects.

$memberJoin : S \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the subject (member) joined the group according to the TR.

$memberLeave : S \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the subject (member) left the group according to the TR.

[2]In practice immediate and preemptive revocation is not feasible, hence some approximation will be necessary. In applying the PEI framework we choose to defer consideration of this approximation to the enforcement model.

$docAdd : O \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the object (document) was added to the group according to the TR.

$docRemove : O \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the object (document) was removed from the group according to the TR.

$authorization : S \cup O \rightarrow \{true, false\} \cup \{NULL\}$ is a partial function that denotes whether the subject/object is authorized to be a group member as per the TR.

4.3 UCON Specification for Ideal Policies

4.3.1 Document read by a member

In this operation, the subject is the member and object is the document.

$Att(S) \supseteq \{memberJoin, memberLeave, ty\}$
$Att(O) \supseteq \{docAdd, docRemove, ty\}$

$allowed(S, O, read) \Rightarrow$
$typeCheck(S, O) \wedge membershipStatusCheck(S, O) \wedge$
$memberGroupPolicyCheck(S, O) \wedge docGroupPolicyCheck(O)$

where,
$typeCheck(S, O) : ty(S) = member \wedge ty(O) = doc$

$membershipStatusCheck(S, O) :$
$memberJoin(s) \neq NULL \wedge docAdd(O) \neq NULL$

$memberGroupPolicyCheck(S, O) :$
$(memberLeave(S) = NULL \wedge docAdd(O) \geq memberJoin(S))$
$\vee\ (memberLeave(S) \neq NULL \wedge$
$memberJoin(S) \leq docAdd(O) \leq memberLeave(S))$

$docGroupPolicyCheck(O) : (docRemove(O) = NULL)$

$stopped(S, O, R) \Leftarrow$
$(docRemove(O) \neq NULL) \vee (memberLeave(S) \neq NULL \wedge$
$docAdd(O) \geq memberLeave(S))$

Note that the *allowed* predicate specifies the predicates necessary for a subject to access an object with respect to the meta-policy specified at the beginning of section 4. It also checks if the members and documents are current. The *stopped* predicate specifies the instant and preemptive revocation requirement. These are given in the style of UCON as defined in [9]. Immediate and preemptive revocation requires the ongoing authorization capability of UCON also called continuity.

4.3.2 Enrolling a member into a group

This is a subject to subject operation between the TR (S1) and the joining Member (S2). It is the TR's discretion to allow new members to join the group.

$Att(S) \supseteq \{ty, authorization, memberJoin, memberLeave\}$

Here S1 and S2 are both subjects:

$allowed(S1, S2, join) \Rightarrow$
$typeCheck(S1, S2) \wedge newMemberCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = TR \wedge ty(S2) = member$

$newMemberCheck(S2)$:
$\neg authorization(S2) = NULL \wedge authorization(S2) = NULL$

$preUpdate(authorization) : authorization'(S2) = true$
$preUpdate(memberJoin)$:
$memberJoin'(S2) = TS1$, where $TS1 \in TS$
$preUpdate(memberLeave) : memberLeave'(S2) = NULL$

In this operation the effect of enrollment is immediate and the pre-update capability of UCON is used to capture the change in membership state.

4.3.3 Removing a member from a group

This is a subject to subject operation between the TR (S1) and the Member (S2).

$Att(S) \supseteq \{ty, authorization, memberLeave\}$

Here S1 and S2 are both subjects.

$allowed(S1, S2, leave) \Rightarrow$
$typeCheck(S1, S2) \wedge currentMemberCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = TR \wedge ty(S2) = member$

$currentMemberCheck(S2) : authorization(S2)$

$preUpdate(authorization) : authorization(S2) = false$
$preUpdate(memberLeave)$:
$memberLeave'(S2) = TS2$, where $TS2 \in TS$

This operation is the converse operation to 4.3.2. The following two operations are similar operations with respect to adding and removing documents and also make use of the mutable attribute capability of UCON.

4.3.4 Adding a document into the group

Here subject is the member and object is the document. Any current member can add a document to the group.[3]

$Att(S) \supseteq \{ty, authorization, memberJoin, memberLeave\}$
$Att(O) \supseteq \{ty, docAdd, docRemove\}$

$allowed(S, O, add) \Rightarrow$
$typeCheck(S) \wedge membershipCheck(S) \wedge newDocCheck(O)$

where,
$typeCheck(S) : ty(S) = member$

$membershipCheck(S) : authorization(S)$

$newDocCheck(O)$:
$docAdd(O) = NULL \wedge docRemove(O) = NULL$

$preUpdate(docAdd) : docAdd'(O) = TS3$, where $TS3 \in TS$
$preUpdate(docRemove) : docRemove'(O) = NULL$

4.3.5 Removing a document from the group

Only a TR can remove a document from the group. Here subject is the TR and object is the document.

$Att(S) \supseteq \{ty\}$
$Att(O) \supseteq \{ty, docAdd, docRemove\}$

$allowed(S, O, remove) \Rightarrow$
$typeCheck(S, O) \wedge currentDocCheck(O)$

where,
$typeCheck(S, O) : ty(S) = TR \wedge ty(O) = doc$

$currentDocCheck(O)$:
$docAdd(O) \neq NULL \wedge docRemove(O) = NULL$

$preUpdate(docRemove)$:
$docRemove'(O) = TS4$, where $TS4 \in TS$

5. ENFORCEMENT MODELS

In this section, we discuss enforcement models for the specified policy framework. We propose an Ideal Enforcement Model that attempts to precisely cover our ideal policy model. We argue that in practical scenarios even the most ideal enforcement model is only an approximation of our policies. We therefore propose Approximate Enforcement Models. We formally specify an approximate enforcement model using UCON and demonstrate that the UCON model could be used to formally specify both Policy *and* Enforcement models.

5.1 UCON Ideal Enforcement Model

Recall that our Ideal Policy Model had an important requirement: instant and preemptive revocation of members and documents from a group. Instant revocation means that members and documents should be immediately removed from the group the moment the TR decides to cancel their membership. Preemptive revocation means that should membership status of either the member or the document change during access, access should be stopped. An Ideal Enforcement Model would then be a client-server model. A member should be permitted to access documents only after verifying membership with a server. However, even such a model of "refresh on every access" may not be totally ideal. Any distributed system has an inherent latency and membership verification made at the time of read request may not be valid at the time of actual read by the member. Thus even in an ideal enforcement model the membership validity is approximated to a window of latency time. Moreover, this enforcement model would directly contradict one of our main objectives: *off-line access*. Hence in order to realize our objectives, we discuss ways to approximate our ideal policy.

[3]Note that modeling adding and removing documents from a group could be more complex than what is shown here. In particular, we have plain-text document and protected document that is added to the group. These two documents are of different types and make the document type mutable. This opens up many issues. For instance, how to prevent the same plain-text document taking multiple forms of protected document? This has implications on document remove. For now, we abstract away from this issue and others in order to focus on our main theme.

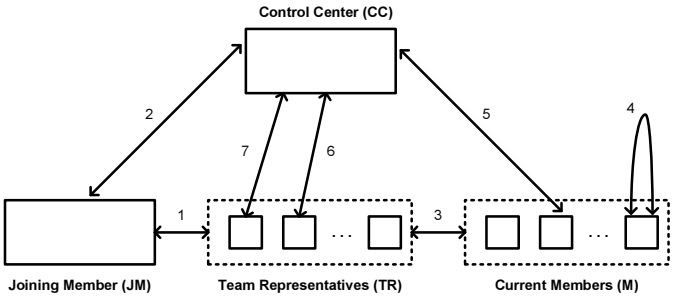

Figure 6: Enforcement Architecture.

5.2 Approximate Enforcement Models

Figure 6 shows our enforcement architecture. In order to facilitate various operations, we introduce a Control Center (CC). The CC acts as a single point of contact for both members and the TR. The TR would update membership status of both members and documents at the CC. The CC would then make access decisions based on current membership. Note that a group could have multiple TRs and we assume that members, TRs and the CC are configured for a group.

Steps 1 and 2 are member join operations. In step 1, the TR authorizes the Joining Member (JM) to join the group. In step 2, the CC officially enrolls the JM into the group by verifying the TR's authorization. In step 3, a member receives approval from the TR to add a document into the group. Step 4 and step 5 differentiates ideal and approximate enforcement models. In an ideal enforcement model, step 4 and 5 will become a single step of verifying membership with the CC before a document is read by the member. We can approximate this step based on usage count or time. A usage count based approximation would allow the member to access documents a specific number of times without having to contact the CC, thus enabling off-line access with limited usage count. A time based approximation would allow the member to access documents for a specific period of time without having to contact the CC, to provide off-line access with a time limit. Thus in approximate models, step 4 involves off-line document read by a member and step 5 would involve refreshing group membership if either the usage count or time expires. The window provided by the usage count or time is a degree of approximation of currentness of group membership. Steps 6 and 7 involve removal of members and documents from the group by the TR.

A time based approximate enforcement model would require an off-line trusted source of time. We are not aware of any TPMs that provide such a feature. This forces us to elect the usage-count based model as our approximate enforcement model. In the following section, we formally specify this model in UCON.

Password based, machine based and credential based enforcements models were proposed in a previous work [14]. However, we will show later that this architecture accommodates all of these models.

Our architecture is different from Microsoft Windows^{TM}[4] Rights Management Services (RMS) in that our motivation here is document access *without* "pre-planning". Preplanning is planning in advance which documents one would need to access off-line in the future and "checking-out" those documents by obtaining licenses to access them from a server. Further, in RMS when a client receives a document from some channel, it needs to contact the server and obtain a use license to open the document. Our architecture incorporates the notion of group level policies which not only controls access during membership but also after members leave the group. Our implementation models as discussed in section 6 uses trusted computing based mechanisms to provide strong client-side policy enforcement.

5.3 UCON Specification for Approximate Enforcement Model

In this section we formally specify an approximate enforcement model for membership renewal based on usage count. The idea is that members will be allowed to access documents for a number of times restricted by an usage count. Once the usage count limit is reached, members need to renew their membership with the CC. The CC would then update the usage count if the membership is still valid. Clearly, there are two sets of attributes in this case. There are regular attributes that are maintained by the members. These attributes may not always reflect the current accurate values (because members refresh attributes only after the usage count is down to zero and thereby have a possible lag with respect to the correct value). Then we also have the *Authoritative attributes* counterpart that are up to date and maintained by the CC. During a refresh cycle, the CC would update the regular attributes maintained by the members with the authoritative attributes maintained by the CC. We define additional attributes below.

$authoritativeMemberJoin : S \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the subject (member) joined the group according to the TR.

$authoritativeMemberLeave : S \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the subject (member) left the group according to the TR.

$authoritativeDocAdd : O \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the object (document) was approved to be added into the group according to the TR.

$authoritativeDocRemove : O \rightarrow TS \cup \{NULL\}$ This is a partial function that denotes the time at which the object (document) was removed from the group according to the TR.

$usageCount : S \rightarrow N$ **where** N is a natural number. This attribute is a partial function that denotes the num-

[4]Microsoft Windows is a registered trademark of Microsoft Corporation in the United States of America and/or in other countries. Other product names mentioned in this article may be trademarks or registered trademarks of their respective companies and are the sole property of their respective manufacturers.

ber of times a member can access documents without refreshing his membership status.

$safeUsageCount : S \rightarrow N$ **where** N is a natural number. This attribute is a partial function that denotes the threshold value below which a member may opportunistically refresh his usageCount.

$docRenewed : S \rightarrow \{true, false\}$ is a partial function that denotes whether the subject has renewed the attributes of all the documents in the subject's machine.

Also we use UCON pre and on-going conditions to mandate internet connectivity during operations as follows:

$getPreCON(S, O, R) = \{(connectivity = true)\}$
$getOnCON(S, O, R) = \{(connectivity = true)\}$

connectivity is the status of internet connectivity to the CC/TR. This requires the use of UCON conditions in this model.

5.3.1 Document read by a member

Here subject is the member and object is the document.

$OBS = CC$
$OBO = S, O$
$OB = renew$

$Att(S) \supseteq \{ty, memberJoin, memberLeave, usageCount, safeUsageCount\}$
$Att(O) \supseteq \{ty, docAdd\}$

If $usageCount(S) \leq 0$,
$getPreOBL(S, O, R) = (CC, S, renew) \wedge (CC, O, renew)$
$otherwise$,
$getPreOBL(S, O, R) = \phi$

$allowed(S, O, R) \Rightarrow$
$preFulfilled(getPreOBL(S, O, R)) \wedge usageCount(S) > 0 \wedge$
$typeCheck(S, O) \wedge membershipCheck(S, O) \wedge$
$memberPolicyCheck(S, O) \wedge docPolicyCheck(O)$

where,
$typeCheck(S, O) : ty(S) = member \wedge ty(O) = doc$
$membershipCheck(S, O)$:
$memberJoin(S) \neq NULL \wedge docAdd(O) \neq NULL$
$memberPolicyCheck(S, O)$:
$(memberLeave(S) = NULL \wedge docAdd(O) \geq memberJoin(S)) \vee$
$(memberLeave(S) \neq NULL \wedge$
$memberJoin(S) \leq docAdd(O) \leq memberLeave(S))$
$docPolicyCheck(O) : docRemove(O) = NULL$

$preUpdate(usageCount)$:
$usageCount'(S) = usageCount(S) - 1$

Comparing with the corresponding operation in section 4.3.1 for ideal policy, there are some similarities. However, we also observe many differences. We have new obligation requirement. Before a read can be allowed, the member is obligated to renew his attributes if his usage count has reached zero. The *preUpdate* predicate decreases the usage count every time the member opens a document. Once the usage count reaches zero, a refresh will be forced to continue document access. Also, the *stopped* predicate is not required anymore since we do not have preemptive revocation here.

5.3.2 Usage Count renewal

This is a subject to subject operation between the CC and the member.

$Att(S) \supseteq \{ty, memberJoin, memberLeave, usageCount, safeUsageCount, authoritativeMemberJoin, authoritativeMemberLeave, docRenewed\}$

$allowed(S1, S2, refresh) \Rightarrow$
$preConChecked(getPreCON(S1, S2, refresh)) \wedge$
$typeCheck(S1, S2) \wedge membershipCheck(S2) \wedge usageCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = CC \wedge ty(S2) = member$
$membershipCheck(S2)$:
$memberJoin(S2) \neq NULL \wedge memberLeave(S2) = NULL$
$usageCheck(S2)$:
$usageCount(S2) < safeUsageCount(S2) \wedge docRenewed(S2)$

$stopped(S1, S2, refresh) \Leftarrow$
$\neg onConChecked(getonCON(S1, S2, refresh)$

$preUpdate(usageCount)$:
$usageCount'(S2) = N1, where N1 \in N$
$preUpdate(memberLeave)$:
$memberLeave'(S2) = authoritativeMemberLeave(S2)$
$preUpdate(memberJoin)$:
$memberJoin'(S2) = authoritativeMemberJoin(S2)$
$preUpdate(docRenewed) : docRenewed'(S2) = false$

This is a new operation (as compared to the policy model) that facilitates renewal of member's attributes including usage count. When the usage count reaches zero, the member will be forced to obtain updated membership attributes from the CC. The CC updates the member's attributes with its authoritative attributes. If the the TR had instructed the CC to remove a member, corresponding attributes will be updated during this step. Note that we have a new UCON condition: *preConChecked* is the condition that there is a connectivity to CC during this operation. If the condition fails, the updates will be stopped. Note that this operation will succeed only if *docRenewed* is true and this forces members to renew the membership status of all the documents (in the following step) they possess.

5.3.3 Document membership renewal

Here we update the membership status of all the documents in the system. The TR could remove a document from the group and inform the CC. When the members come for renewal, this operation forces all documents on member's machine to be renewed as part of the usage count renewal in the previous operation. Again, this is new operation compared to the policy model. Here the subject is the Member and objects are the documents in the member's system. Note that *preConChecked* checks whether connectivity to CC is available during this operation.

$Att(S) \supseteq \{ty, docRenewed\}$
$Att(O) \supseteq \{ty, docAdd, docRemove, authoritativeDocRemove\}$

$allowed(S, O_i, refresh) \Rightarrow$
$preConChecked(getPreCON(S, O_i, refresh)) \wedge$
$typeCheck(S, O_i)$

where,
$typeCheck(S, O_i)$:
$ty(S) = CC \wedge ty(O_i) = doc$,
for i=1 to Z, where Z is the total number of documents the subject has.

$stopped(S, O_i, refresh) \Leftarrow$
$\neg onConChecked(getOnCON(S, O_i, refresh)$

$preUpdate(docRemove)$:
$docRemove(O_i) = authoritativeDocRemove(O_i)$
$\forall authoritativeDocRemove(O_i) \neq NULL$,
where i=1 to Z, Z is the total number of documents the subject has.
$preUpdate(docRenewed) : docRenewed(O) = true$

Exactly how this document renewal is efficiently done is an implementation issue and we abstract away from such questions at the enforcement layer.

5.3.4 Enrolling a member into a group

Compared to the corresponding operation in section 4.3.2 in the ideal policy model, observe that member join is now a two step process. In step 1, the TR authorizes the joining member to join the group. In step 2, the CC officially enrolls the joining member into the group.

Step 1: This is a subject to subject operation between TR and the joining member. In the policy model, we did not have a CC and the TR was responsible for setting the join and leave time-stamps (TS). With our two step join architecture, we move this operation of setting time-stamps to the CC. The TR authorizes join and the CC officially enrolls the member by setting the time-stamps.

$Att(S) \supseteq \{ty, authorization\}$

$allowed(S1, S2, join) \Rightarrow$
$preConChecked(getPreCON(S1, S2, join)) \wedge$
$typeCheck(S1, S2) \wedge authorizationCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = TR \wedge ty(S2) = member$
$authorizationCheck(S2) : authorization(S2) = NULL$

$stopped(S1, S2, join) \Leftarrow$
$\neg onConChecked(getOnCON(S1, S2, join))$

$preUpdate(authorization) : authorization'(S2) = true$

Step 2: This is a subject to subject operation between the CC and the joining member. Here *preConChecked* is the condition that there is connectivity to the CC.

$Att(S) \supseteq \{ty, authorization, usageCount, safeUsageCount,$
$authoritativeUsageCount, authoritativeSafeUsageCount,$
$memberJoin, memberLeave,$
$authoritativeMemberLeave, authoritativeMemberJoin\}$

$allowed(S1, S2, add) \Rightarrow$
$preConChecked(getPreCON(S1, S2, add)) \wedge$
$typeCheck(S1, S2) \wedge authorizationCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = CC \wedge ty(S2) = member$
$authorizationCheck(S2)$:
$authorization(S2) = true \wedge$
$authoritativeMemberJoin(S2) \neq NULL$

$stopped(S1, S2, add) \Leftarrow$
$\neg onConChecked(getOnCON(S1, S2, add))$

$preUpdate(authoritativeMemberJoin)$:
$authoritativeMemberJoin'(S2) = TS1, whereTS1 \in TS$
$preUpdate(memberJoin)$:
$memberJoin'(S2) = authoritativememberJoin(S2)$
$preUpdate(authoritativeMemberLeave)$:
$authoritativeMemberLeave'(S2) = NULL$
$preUpdate(memberLeave)$:
$memberLeave'(S2) = authoritativeMemberLeave(S2)$
$preUpdate(authoritativeUsageCount)$:
$authoritativeUsageCount'(S2) = N1, whereN1 \in N$
$preUpdate(usageCount)$:
$usageCount'(S2) = authoritativeUsageCount(S2)$
$preUpdate(authoritativeSafeUsageCount(S2))$:
$authoritativeSafeUsageCount'(S2) = N2, whereN2 \in N$
$preUpdate(safeUsageCount)$:
$safeUsageCount'(S2) = authoritativeSafeUsageCount(S2)$

5.3.5 Removing a member from the group

This is a two step process that happens between the TR and the CC and is different from the corresponding operation in section 4.3.3 of the policy model. In step 1, the TR removes the authorization of the member at the CC. In step 2, the CC updates the member's authoritative attributes. These attributes will be pushed into the member's system when the member comes to the CC for renewal.

Step 1: This is a subject to subject operation between the TR and the member. The member attributes here actually represents the member's attributes on the CC. Here *preConChecked* means that there is connectivity to the CC.

$Att(S) \supseteq \{ty, authorization, authoritativeUsageCount,$
$authoritativeSafeUsageCount, memberJoin, memberLeave\}$

$allowed(S1, S2, remove) \Rightarrow$
$preConChecked(getPreCON(S1, S2, remove)) \wedge$
$typeCheck(S1, S2) \wedge authorizationCheck(S2)$

where,
$typeCheck(S1, S2) : (ty(S1) = TR \wedge ty(S2) = member)$
$authorizationCheck(S2) : authorization(S2)$

$stopped(S1, S2, remove) \Leftarrow$
$\neg onConChecked(getOnCON(S1, S2, remove))$

$preUpdate(authorization(S2)) : authorization'(S2) = false$

Step 2: This is a subject to subject operation between the CC and the member. The member attributes here represents the member's attributes on the CC.

$Att(S) \supseteq \{authorization, authoritativeUsageCount, authoritativeSafeUsageCount, memberJoin, memberLeave\ authoritativeMemberJoin, authoritativeMemberLeave\}$

$allowed(S1, S2, remove) \Rightarrow$
$typeCheck(S1, S2) \wedge authorizationCheck(S2)$

where,
$typeCheck(S1, S2) : ty(S1) = CC \wedge ty(S2) = member$
$authorizationCheck(S2)$:
$\neg authorization(S2) \wedge$
$authoritativeMemberJoin(S2) \neq NULL \wedge$
$authoritativeMemberLeave(S2) = NULL)$

$preUpdate(authoritativeMemberLeave)$:
$authoritativeMemberLeave'(S2) = TS2, whereTS2 \in TS$
$preUpdate(authoritativeUsageCount)$:
$authoritativeUsageCount'(S2) = 0$
$preUpdate(authoritativeSafeUsageCount)$:
$authoritativeSafeUsageCount'(S2) = 0$

This operation is the converse operation to 5.3.4. The following two operations are similar operations with respect to adding and removing documents and also make use of the mutable attribute capability of UCON.

5.3.6 Adding a document to the group

Here the subject is the member and object is the document. *preConChecked* is the condition that there is connectivity to CC. For brevity, we do not consider the TR approval for document addition (note that this is a policy level decision). The *allowed* predicate makes sure that only current members can add documents to the group.

$Att(S) \supseteq \{ty, memberJoin, memberLeave\}$
$Att(O) \supseteq \{ty, docAdd, docRemove\}$

$allowed(S, O, add) \Rightarrow$
$preConChecked(getPreCON(S, O, add) \wedge$
$typeCheck(S, O) \wedge membershipCheck(S, O)$

where,
$typeCheck(S, O) : ty(S) = member \wedge ty(O) = doc$
$membershipCheck(S, O)$:
$memberJoin(S) \neq NULL \wedge$
$memberLeave(S) = NULL \wedge docAdd(O) = NULL)$

$stopped(S, O, add) \Leftarrow$
$\neg onConChecked(getOnCON(S, O, add))$

$preUpdate(docAdd) : docAdd'(O) = TS3, whereTS3 \in TS$
$preUpdate(docRemove) : docRemove'(O) = NULL$

5.3.7 Removing a document from the group

Here subject is the TR and object is the document. *preConChecked* is the condition that there is connectivity to CC. The *allowed* predicate makes sure that only the TR can remove a document.

$Att(S) \supseteq \{memberJoin\}$
$Att(O) \supseteq \{docAdd, authoritativeDocRemove\}$

$allowed(S, O, remove) \Rightarrow$
$preConChecked(getPreCON(S, O, remove)) \wedge$
$typeCheck(S, O) \wedge membershipCheck(O)$

where,
$typeCheck(S, O) : ty(S) = TR \wedge ty(O) = doc)$
$membershipCheck(O)$:
$docAdd(O) \neq NULL \wedge authoritativeDocRemove(O) = NULL$

$stopped(S, O, remove) \Leftarrow$
$\neg onConChecked(getOnCON(S, O, remove))$

$preUpdate(authoritativeDocRemove)$:
$authoritativeDocRemove'(O) = TS4, whereTS4 \in TS$

6. IMPLEMENTATION MODELS

In this section, we give a brief overview of our implementation model based on Trusted Computing Technology [1]. As mentioned earlier, we need trust at the hardware level as provided by the TPM for the high-assurance requirements of the SIS problem. The implementation model involves a Trusted Reference Monitor [15] (TRM) module on every group member's machine. The TRM is a trust-worthy reference monitor that enforces group policies on the client. We also have a Trusted Viewer (TV) module that is used for viewing documents on member's machines. For our discussion we assume that the TRM and TV are somehow provisioned on the member's machines in a trust-worthy manner. We can use the mechanisms provided by the TPM [1] to protect the integrity of the TRM and the TV. It is beyond the scope of this paper to go over the protocols for each of the steps in figure 6. We only outline the protocol with main steps here.

In step 1, the Team Representative (TR) provides a signed credential to the Joining Member (JM) and authorizes joining the group. In step 2, the JM uses this credential to enroll into the group by contacting the Control Center (CC). The CC verifies the credential and installs a ticket on the JM in such a manner that only the TRM can access this ticket. This is achieved using TPM functionalities like seal, unseal, integrity measurements, etc. as mentioned in section 2. The format of this ticket would be: $signCC\{\{meta-policy||K||refreshMonotonicCount||TSJoin||TSLeave||DRL\}\}$. The meta-policy specifies the group level policies that needs to be enforced. K is the group key. The refresh frequency is defined by $refreshMonotonicCount$. It is the number of times the group key can be used before a membership status refresh is forced. Monotonic Counter is a hardware counter provided by the TPM. We use this counter (or a virtual monotonic counter) to keep track of refresh counts and prevent replay of older tickets. Note that only the TRM should be authorized to update this counter. $TSJoin$ is the time-stamp of member join time as seen by the CC and $TSLeave$ is the time-stamp of leave. DRL is the Document Revocation List that lists the documents that have been removed from the group. This whole ticket is signed by the CC and installed in such a fashion that only the TRM can access it.

The TRM would only let the TV open a document. Documents are always stored on the disk in protected form and is of the format: $signTR\{doc||K\{docKey\}||hash(doc)||TSAdd\}$.

doc represents the encrypted version of the document. The document is encrypted with a *docKey*, the document key. The *docKey* is encrypted with the group key K. $TSAdd$ is the time-stamp of the time at which the document was approved to added into the group by the TR. $hash(doc)$ is the hash value of the document. The TR's signature ($signTR$) shows the approval of the TR for this document to be added to the group.

In the following subsection we discuss how the TRM enforces policies for document read based on the ticket and the document format we discussed.

6.1 Enforcing Document Read

The TV requests the TRM to open the document. The TRM initially verifies if the TV is not tampered using the integrity measurement mechanisms of the TPM. Then the TRM would unseal the ticket. This step would succeed only if the TRM is in the same integral state as it was when the ticket was sealed (this could be guaranteed by TPM mechanisms). First the TRM would check the hardware monotonic counter (or the virtual monotonic counter) with the $refreshMonotonicCount$ and make sure it has not expired. Next it would check that the document ($hash(doc)$) is not part of the DRL. Next the TRM checks that the meta-policy permits opening the document based on $TSAdd$ of the document and the $TSJoin$ and $TSLeave$ of the member (as discussed in the section 5.3.1). If these steps succeed, the TRM would decrypt the document for the TV using K and the $docKey$. Note that if the $refreshMonotonicCount$ has expired, the ticket is no more valid and the member is forced to obtain a new ticket from the CC. The new ticket would contain updated counter value and updated time-stamps if the membership status has changed.

7. CONCLUSIONS AND FUTURE WORK

In this paper we investigated the Secure Information Sharing problem with three main objectives: scalability, usability and high-assurance. For scalability we proposed super-distribution within a group, for usability we proposed off-line access and for high-assurance we proposed trusted computing based mechanisms. We identified a rich policy framework for the information sharing space and formalized this framework using the UCON model. We also demonstrated that a model as rich as UCON (with the full machinery of authorization, obligations and conditions) is required for the SIS problem. We explored enforcement architectures and formally stated a usage count based approximate enforcement architecture. We outlined the steps for implementation models using Trusted Computing Technology.

Our approach to solving this problem using the PEI framework has many advantages. The enforcement and implementation models can be easily enhanced to accommodate interesting scenarios. For example, documents could be password protected by simply adding the password hash to the document and modifying the TRM to enforce password protection. Further passwords could be added on a per-document basis or on the whole group basis. For enforcing a password throughout the group, the password hash could be added to the ticket. The group level policies could be modified to member-level policies by constructing tickets with a different meta-policy for different members. These could be generalized to a credential based enforcement model where the required credential would be mentioned as part of the ticket. The TRM would then mandate such credentials from group members. Credentials could be role certificate, attribute certificate, etc.

A range of future work is underway. First a prototype needs to be built as a proof-of-concept for the group-based SIS problem. We have started investigating components required to build such a prototype. We need to explore generality of the policy framework to application scenarios other than document sharing. Restricting information flow across groups needs to be investigated. Support for document querying to obtain specific sections of documents instead of the whole document has many interesting usage scenarios. To support a true secure collaboration, document write should is an important requirement and hence should be studied.

8. REFERENCES

[1] Tcg specification architecture overview. *http://www.trustedcomputinggroup.org*.

[2] Dod trusted computer systems eval. criteria. Dec 1985.

[3] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. *Technical Report, The Mitre Corp.*, March 1975.

[4] D. Denning. A lattice model of secure information flow. *Comm. of the ACM*, pages 236–243, 1976.

[5] G. Graham and P. Denning. Protection principles and practice. *AFIPS Joint Computer Conference*, 1972.

[6] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Comm. of the ACM*, pages 461–471, August 1976.

[7] B. Lampson. Protection. *In fifth Princeton Symposium on Information Science and Systems*, 40:437443, 1971.

[8] S. Osborn, R. Sandhu, and Q. Munawer. Configuring rbac to enforce mandatory and discretionary access control policies. *ACM TISSEC*, 3(2), May 2000.

[9] J. Park and R. Sandhu. The ucon abc usage control model. *ACM Transactions on Information and System Security*, 7(1):128–174, Feburary 2004.

[10] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, pages 309–329, September 2003.

[11] R. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[12] R. Sandhu. Engineering authority and trust in cyberspace: the om-am and rbac way. *Proc. of the fifth ACM workshop on RBAC*, pages 111–119, 2000.

[13] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Feburary 1996.

[14] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and pei models. *Proc. of ASIACCS 2006*, pages 2–12, 2006.

[15] R. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. *Proceedings of the 10th ACM symposium on Access control models and technologies*, pages 147–158, 2005.