*International Journal of Innovative Computing, Information and Control*
Volume **8**, Number **3(A)**, March **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 1821–1836

# DYNAMIC REMOTE ATTESTATION THROUGH BEHAVIOR MEASUREMENT AND VERIFICATION

Masoom Alam[1], Xinwen Zhang[2], Mohammad Nauman[1], Tamleek Ali[1]
Sanaullah Khan[1], Shahbaz Khan[1], Quratulain Alam[1], Sajid Anwar[1]
Arfan Jaffar[3], Amir Hayat[4], Muhammad Ali[1] and Awais Adnan[1]

[1]Security Engineering Research Group, Institute of Management Sciences
1-A, E-5, Phase VII, Hayatabad, Peshawar, Pakistan
{ masoom.alam; nauman; tamleek; sanaullah }@imsciences.edu.pk
{ shazalive; q.alam; sajid.anwar; m.ali; awais }@imsciences.edu.pk

[2]Huawei Research Center at Santa Clara, CA, USA
Xinwen.Zhang@huawei.com

[3]FAST-National University of Computer and Emerging Sciences, Islamabad, Pakistan
arfan.jaffar@nu.edu.pk

[4]School of Electrical Engineering and Computer Science, NUST, Pakistan
amir.hayat@seecs.nust.edu.pk

Received November 2010; revised March 2011

Abstract. *The interdisciplinary nature of information handled by the enterprise information systems and their interaction with domains beyond organizational boundaries necessitates the incorporation of comprehensive security mechanisms within such systems, enabling them to manage access control requirements in a flexible manner, making sure that the data assets of the organization and its customers are safe not only within the organization but also outside its boundaries. Therefore, state of the systems outside the direct control of the organization must be verified before granting access to sensitive data so as to make sure that such client systems are benign and that the resource will be used according to expectations. UCON provides control over access to information and its usage at a fine grained level which is not possible with traditional access control, while trusted computing technologies in general and remote attestation in particular can be used for the verification of the client systems along with the protection of the host systems from rootkits and other security attacks. Remote attestation is an important characteristic of trusted computing technology which provides reliable evidence that a trusted environment actually exists. Existing approaches for the realization of remote attestation measure the trustworthiness of a target platform from its binaries, configurations, properties or security policies. All these approaches are low-level attestation techniques only, and none of them define what a trusted behavior actually is and how to specify it. In this paper*[1]*, we present a novel approach to verify the trustworthiness of a platform whereby trustworthiness of the platform is associated with the behavior of a policy model. In our approach, the behavior of a policy model is attested rather than a software or hardware platform. Thus, the attestation feature is not tied to a specific software or hardware platform, or to a particular remote attestation technique, or to an individual type of security policy. We select usage control (UCON) as our target policy model in the context of a health care enterprise information system. We propose a framework to identify, specify and attest different behaviors of UCON. We discuss the prototype implementation for the realization of our approach.*

**Keywords:** Security, Trusted computing, Remote attestation, Behavioral attestation, Usage control

[1]A previous version of the paper was published in SACMAT 2008.

## 1. Introduction.

Security of enterprise information systems is of utmost importance in todays computing environment, as it is increasingly becoming difficult for an organiztion to effectively operate within the confines of a closed system. More often than not, an enterprise information system exchanges data with other domains mostly over a network, potentially exposing the system and the hosted information to malicious entities. Therefore, it is crucial for an enterprise information system to not only protect the incumbent data from unauthorize access but also make sure that the authorized entities use the data or information according to expectations. To tackle these challenges effectively enterprise information systems need to provide fine grain control access control over the data belonging to the organization or its customers. Ideally, an enterprise information system should verify the configuration of a platform before communicating any sensitive data in an open and potentially hostile environment. To counter these challenges there needs to be paradigm shift in the way enterprise information systems are designed, maintained and applied encorporating state of the art access control and computer security technologies within them, utilizing software as well as leveraging capabilities of the hardware based security.

Due to the inter-disciplinary nature of information handled by such systems, they need a diverse form of access control one which is capable of handling traditional access control as well as DRM while at the same time address the issues of trust managemet in an open computing environment. Recent developments in the field of access control have resulted in the emergence of usage control or UCON. Park and Sandhu [1] have coined the term usage control and proposed a model called *Usage CONtrol* (UCON), which enhances traditional access control models [2-4] in two respects: 1) continuity of an access decision and 2) mutability of attributes. *Continuity of an access decision* means that a decision to allow access to an object is made not only before access but also during access and may result in revocation of access permissions if policy conditions are no longer satisfied. *Mutability of attributes* means that attributes of subjects or objects may change[2] as side effects of access, which may also result in a change in ongoing or subsequent access decisions.

Research on computers and information security has exposed the weaknesses of software based solutions for computer security in general and access control in particular. To be truly effective, security of enterprise software must be rooted in hardware so as to be effective against sophisticated security attacks such ad rootkits. Trusted computing technology is an effort for enhancing the security of computer systems via hardware based mechanisms augmenting the softwares responsible for the security of such platforms so as to remove their inherent weaknesses.

The term "Trusted Computing" refers to a technology introduced by the *Trusted Computing Group* (TCG) [5], in which PCs, consumer electronic devices, PDA's and other mobile devices are equipped with a special hardware chip called *Trusted Platform Module* (TPM).

TCG defines trust as follows: "*trust is the expectation that a device will behave in a particular manner for a specific purpose*" [6]. The term *particular manner* is concerned with the question of *how* a task is expected to be performed; *specific purpose* refers to a particular task or scenario like usage of an object, web service access, or some computational activity. In other words, *"trust is directly associated with the expected behavior of a particular task"*.

Using trusted computing technologies, an enterprise information system can verify the state of a platform before releasing sensitive data assests to make sure that the client

[2]Only subject or object attributes mutability is allowed in UCON.

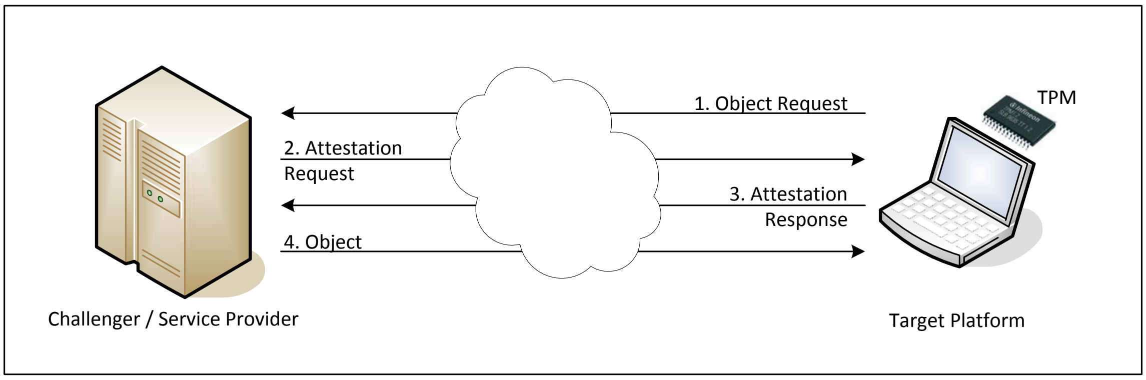


FIGURE 1. A typical remote attestation scenario

platform is capable of protecting the resource and that its usage will be according to expectations. Remote attestation is an essential characteristic of trusted computing which provides reliable evidence that a trusted environment actually exists on a specified platform. This feature enables a trusted computing platform to remotely certify to third parties in enciphered form the behavior of its running software and the status of its hardware and software components. In a typical remote attestation scenario (cf. Figure 1), a *challenger* verifies the trustworthiness of a *target* or *remote platform* before dispatching a resource, or before or during an access to an object. If the target platform has provable trusted environment, sensitive information can be released to it.

Several approaches have been proposed in the literature for the realization of remote attestation. For instance, configuration-based attestation requires that a target platform presents the trusted configurations of its platform to a challenger. Based on the configurations, the challenger determines the trustworthiness of a target platform. However, revealing all system configurations may give some insights into the target platform, thus making a security attack inevitable [7].

### 1.1. **Our approach.**

In this paper, we present a novel approach – Model-Based Behavioral Attestation (MBA) – in which the trustworthiness of a target platform is associated with the behavior of its *policy model*. As a policy model provides an abstract and formal representation of the security properties of a platform, we define trust as follows: "*trust is the expectation that a* policy model *will behave in a particular manner for a specific purpose*", where *"the behavior of a policy model refers to the aggregate of its observable actions or reactions in response to its environment"*. Thus, in MBA, a target platform is trustworthy for a challenger if each behavior of that policy model is trustworthy, which is followed by the target platform for the specific purpose of the challenger.

An application, be it a Java virtual machine, or a browser, implements its policy model through its reference monitor. A reference monitor is a module within an application responsible for executing and enforcing its access control policies. According to MBA, traditional chain of trust proposed by the TCG is taken further up to the reference monitor of an application. Dynamic behavior of an application is achieved by analyzing the enforcement of its access control policies. We present a case from health care domain; however, in practice the approach of MBA is realizable for a wide range of softwares such as virtual machines, browsers, etc.

Our MBA approach provides a high-level framework under which multiple low level attestation frameworks can be realized. This means that our target architecture realizes a high-level framework which uses 1) chain of trust mechanisms implemented by IMA, thus ensuring that underling operating environment of an application is trustworthy, 2) application specific isolation mechanism enabled by the PRIMA, thus allowing only defined applications to communicate with the target application and 3) property verification approach defined by the property based attestation in order to verify abstract properties of a platform rather than its hardware and software configurations. In fact, our MBA approach brings the chain of trust up to the corresponding application (in our case health care one) and afterwards, analyzes the reference monitor of the corresponding application which implements a policy model.

1.2. **Contributions.** Our contributions in this paper are the following: 1) We present MBA, which is a high-level framework that abstracts the details of low-level attestation techniques. It is not a new attestation technique. Rather, it provides a framework through which the existing low-level techniques can be selected based on different scenarios. 2) We generalize behavioral attestation [8] and associate behaviors with policy *models* instead of individual security policies. Thus, the attestation is more exact, simple and scalable. 3) We identify and specify the correct behavior of UCON as an example target policy model for demonstration of MBA. 4) Finally, we present a target architecture for the realization of our approach in the context of behavioral attestation of the UCON model encorporated in a healthcare enterprise information system.

UCON is a comprehensive usage control model, which covers all aspects of usage control. This selection is motivated by the fact that usage control scenarios are concerned with the enforcement of access control policies about an object, which does not necessarily remain within the domain of its stakeholder. For example, in health care scenarios, medical data can be accessed within private clinics, laboratories and/or other hospitals. Without guaranteeing the correct enforcement of usage control policies, it is impossible to impose constraints on object usage. For detail knowledge of the UCON model, we refer the reader to [1].

1.3. **Outline.** The identification of different behaviors associated with the UCON model is described in Section 2 and its specification elaborates the attestation of enforcement behaviors. Section 3 presents the target architecture for the realization of our approach. In Section 4, we conclude our contributions and present future directions for this research.

2. **Behavior Identification and Association.** In order to formally specify the behavior of a policy model, its components need to be identified. A policy model is built from its distinct components, the behavior of each of which effects the overall behavior of the policy model. Thus, in our viewpoint, *the behavior of a policy model is the aggregate of observable actions or reactions of its distinct components in response to their environment.* In this statement, the *environment* of a policy model is defined by its policy instance and *observable actions* refer to those actions which are performed by different components of a policy model during policy enforcement. Observability of actions means that there must be a way of communicating to the challenger that the required actions were performed.

Based on the components of UCON, we identify three types of behaviors which need to be associated with the UCON model. These are (1) active subject/object behaviors, (2) state transition behaviors and (3) attribute update behaviors.

1. *Active subject/object behaviors* capture the behaviors of all subjects and objects and their corresponding rights, which are active for a given system state. These behaviors correspond to the subjects, objects, and rights of UCON.
2. *State transition behaviors* capture the behaviors of state transitions when associated authorizations, obligations, and conditions are fulfilled. These behaviors correspond to authorizations, obligations, conditions, system states, and state transition actions

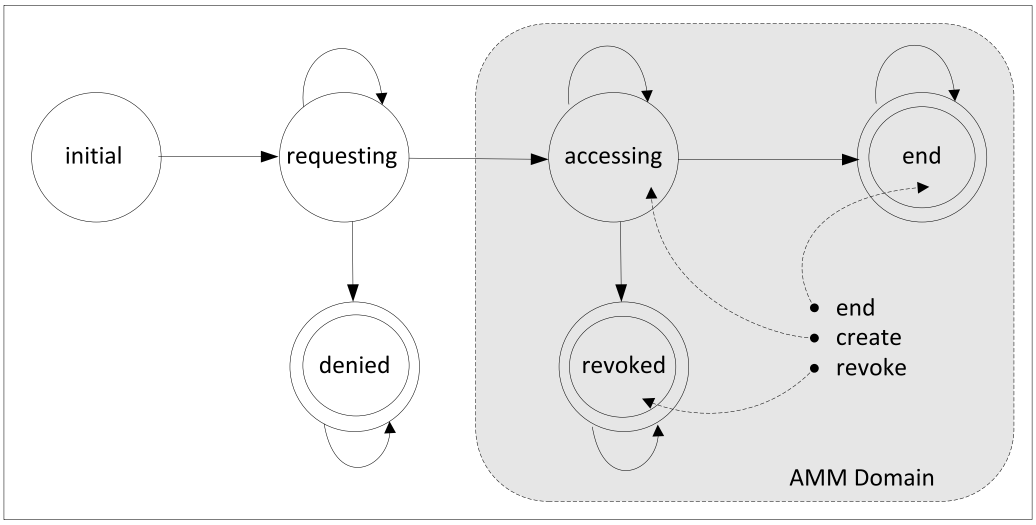


Figure 2. UCON AMM domain

of UCON. Moreover, these behaviors also capture the behavior of attributes involved in related predicates.

3. *Attribute update behaviors* capture the behaviors of attribute update actions such as *preupdate*, *onupdate*, and *postupdate*. These behaviors correspond to the update actions of subject and object attributes.

Note that although UCON considers conditions as decision making factors, it does not capture changes to conditional information (system attributes). Therefore, behaviors of system attribute updates are not included in MBA. In general, MBA trusts that system attributes are monitored and updated in a trusted way.

In general, the criteria for behavior analysis is complete, if it covers all the components of a given target policy model. Our behavior analysis of UCON is complete, since it covers all the components of the UCON model.

## 2.1. **Active subject/object behaviors.**

UCON is a session oriented model in which a state transition activates or deactivates a subject and/or an object. We use Access Monitoring Matrix (AMM) to capture the behaviors of subjects or objects added and removed as a result of different state transitions. We define UCON ACM as follows:

**Definition 2.1.** *The AMM of a UCON system state is defined as* $A : O \times R \to 2^S$ *where* $O$ *is a set of active objects,* $R$ *is a set of rights, and* $S$ *is a set of active subjects.*

Thus, if $s \in A(o, r)$, it means that subject $s$ is exercising right $r$ on object $o$ in some state. For simplicity, we assume that there is only one usage session for a single $(s, o, r)$ existing at one time. However, one subject can access multiple objects and one object can be accessed by multiple subjects at the same time. Figure 2 shows different UCON states and their associated *AMM actions*. The AMM actions mark the domain of active subjects and objects. For example, a subject or object is considered active when a state transition from *requesting* to *accessing* occurs. The AMM action *create* captures the corresponding behavior. Likewise, the AMM action *revoke* and *end* capture the behaviors when an access has been revoked and when an access is ended, respectively. These AMM actions are described below.

**AMM Action** *create***:** Whenever a subject is permitted to access an object, the AMM action *create* adds a new subject $s$ and object $o$ to the set of active subjects $S$ and the set of active objects $O$, respectively. The new set of active subjects and set of active objects are denoted by $S'$ and $O'$, respectively. Further, the creator $s$ is added to AMM for $(o, r)$ and the modified AMM $(A')$ is the new AMM. Formally:

$$\mathit{create}(s, o, r) = (O', S', R, A')$$

where $S' = S \cup \{s\}$, $O' = O \cup \{o\}$ and $A'(o, r) = A(o, r) \cup \{s\}$.

**AMM Action** *revoke***:** The AMM action *revoke* captures the behavior when a subject or object is removed from the set of active subjects and the set of active objects due to state transition from *accessing* to *revoked*. The *revoke* AMM action is defined as follows:

$$\mathit{revoke}(s, o, r) = (O', S', R, A')$$

where

$$\begin{aligned} S' &= S \ominus \{s\} =^{def} \begin{cases} S & |A^{-1}(\{s\})| \geq 2 \\ S - \{s\} & \text{otherwise} \end{cases}, \\ O' &= O \ominus \{o\} =^{def} \begin{cases} O & \sum_{i=1}^{n} |A(o, r_i)| \geq 2 \\ O - \{o\} & \text{otherwise} \end{cases} \\ A'(o, r) &= A(o, r) - \{s\} \end{aligned}$$

In order to obtain the current set of active subjects, subject $s$ is removed from the set of active subjects $(S \ominus \{s\})$. However, the removal places a restriction that if subject $s$ is accessing more than one objects on the AMM – verified through the inverse of $A$ – subject $s$ is not removed. Similarly, before removing object $o$ from the AMM, it is also verified that object $o$ is not being accessed by any subjects through any right. Further, the AMM is updated at $(o, r)$ by removing $s$ from the AMM entry $(o, r)$. The new set of active subjects and objects are denoted by $S'$ and $O'$, respectively.

**AMM Action** *end***:** Similar to *revoke*, the AMM action *end* conditionally removes the active subject and object from the AMM. Formally:

$$\mathit{end}(s, o, r) = (O', S', R, A')$$

The AMM action *revoke* is caused by a call from usage control system whereas, the AMM action *end* is caused by a subject itself. The effects of both the transitions are same, thus, both AMM actions result in the same behavior. Based on the AMM actions, we now define the trustworthiness of the UCON AMM as follows:

**Definition 2.2.** *The UCON AMM behavior is trustworthy if all of the following conditions hold:*

- $\mathit{create}(s, o, r) \to o \in O' \wedge s \in S' \wedge s \in A'(o, r)$
- $\mathit{revoke}(s, o, r) \to S' = S \ominus \{s\} \wedge O' = O \ominus \{o\} \wedge$
  $A'(o, r) = A(o, r) - \{s\}$
- $\mathit{end}(s, o, r) \to S' = S \ominus \{s\} \wedge O' = O \ominus \{o\} \wedge$
  $A'(o, r) = A(o, r) - \{s\}$

*Three symbols $\mathcal{CR}$, $\mathcal{EN}$, $\mathcal{RK}$ are defined to show that trustworthiness is expected for AMM actions create, end and revoke, respectively.*

The above definition states that the UCON AMM is trustworthy if following the AMM action $\mathit{create}(s, o, r)$, subject $s$ is added to the set of active subjects (represented by $S'$), object $o$ is added to the set of active objects (represented by $O'$) and the AMM itself contains an entry $(o, r)$ such that $s \in A(o, r)$. Similarly, based on the definitions of the AMM actions *end* and *revoke*, the trustworthiness of the AMM actions *end* and *revoke* is defined.

In general, for every identified behavior, the formal specification takes the form: first, the behavior and its trustworthiness are defined. Afterwards, a symbol is used to denote that trustworthiness is expected for the corresponding behavior.

2.2. **State transition behaviors.** In UCON, a state transition action is associated with a combination of authorization, obligation and condition statements. For simplicity, we assume that a logical expression represents all of them such that a state transition occurs only if the logical expression associated with the transition is *true*. We define a state transition behavior as follows:

**Definition 2.3.** *Given a state $t_i$ and a logical expression $e$, we write $t_{i-1} \rightarrow_e t_i$ if the transition from state $t_{i-1}$ to $t_i$ is allowed by the logical expression $e$ where $e$ is a conjunction of authorization, obligation, and condition statements.*

Logical expressions contain subject, object and/or environment attributes. These attributes play a key role in sorting out usage control decisions in UCON. Due to their significant importance, the correct enforcement of UCON policies requires that all attributes involved in a particular usage control scenario should be trusted. If attributes, or the procedures responsible for updating them are not trusted, the end result is an untrusted attributes mutability, or untrusted decision continuity. Consider an example, when attribute like subject location or number of times that an object can be accessed is updated in an untrusted way, or the behavior of the login shell representing a subject or the memory area used to load an object is not trusted, the state transition resulting from this attribute predicate is not a valid transition; i.e., the resulting system state is not trusted. Instead of relying on normal attribute mutability and decision continuity – the key UCON concepts, we propose trusted attribute mutability and trusted decision continuity.

Trusted attribute mutability means that subject or object attributes can be updated, if and only if, the corresponding attribute behaviors are trusted and the procedures responsible for updating them are also trusted. Similarly, trusted decision continuity means that in addition to decision continuity features of UCON, all the attributes involved in a particular state transition should be in a trusted state.

Each subject $s$ or object $o$ is represented by a set of attributes in UCON. For trusted decision continuity, we define each attribute $attr_i$ to be a 3-tuple (*name*, *value*, *behavior*), where *name* and *value* have the same semantics as defined in UCON, and *behavior* represents the trusted status of the attribute. At model level, *behavior* is an abstract entity which can have different semantics for different attributes. For example, for memory, it can mean the status of memory protection against intrusions; for a binary file, it can mean the integrity of the file. The actual semantics are augmented in the behavior transformation step. The behavior of an attribute is either $TRUSTED$ or $UNTRUSTED$, where a $TRUSTED$ value corresponds to a Boolean value *true* and $UNTRUSTED$ corresponds to a Boolean value $false$. We define the trustworthiness of a state transition behavior as follows:

**Definition 2.4.** *Let $Attributes = \{attr_1, attr_2, \ldots, attr_n\}$ represent a finite set of subject, object and environment attributes defined in a logical expression $e$ such that $t_{i-1} \rightarrow_e t_i$. Then, the trustworthiness of a state transition behavior takes the form:*

$$\forall j.\ attr_j.behavior = TRUSTED \wedge e = TRUE$$

*where $j = 1, \ldots, n$.*

*We use $\rightarrow_e$ to denote that trustworthiness is expected for the corresponding state transition behavior.*

2.3. **Attribute update behaviors.** Attribute update actions such as *preupdate*, *onupdate* and *postupdate* enable attributes mutability. We define attribute update behavior as follows:

**Definition 2.5.** *Given an attribute $x_i$, and an update operation 'update', the attribute update behavior is defined as the application of the update operation on the attribute. The application of the operation yields a new value which is assigned to the attribute. Formally:*

$$update : x_i \to x_i'$$

The trustworthiness of attribute update behavior is defined as follows:

**Definition 2.6.** *An attribute update behavior is trustworthy iff the corresponding attribute is in a trusted state and the procedure updating it is also trusted. Formally:*

$$update(x_i) = \begin{cases} true & iff \;\; x_i.behavior = TRUSTED \wedge \\ & update.behavior = TRUSTED \\ false & otherwise \end{cases}$$

*where $i = 1, \ldots, n$ and $x_i$ represents the subject or object attribute that needs to be updated.*

*We write $\mathcal{A}U_x$ to denote that a trusted change is expected for attribute $x$, and $\mathcal{A}U$ for the set of all attributes that can be updated.*

The *update* represents the *preupdate*, *onupdate* and *postupdate* attribute update actions. Any attribute update action is an (*update, behavior*) pair, where *behavior* represents the trusted status of the corresponding update procedure.

## 3. **Target Architecture.**

In this section, we present our reference architecture for the realization of MBA approach using a case study from health care domain. Figure 3 is a high level abstraction our target architecture. The client side hosts a remote attestation agent that is capable of handling various remote attestation scenarios using different techniques. The server specifies the remote attestation requirements in the remote attestation challenge. The remote attestation challenge is a request which identifies the information that is to be reported by the remote attestation agent, e.g., the server may request the client system to perform binary attestation on specified applications or other system components alternatively other remote attestation techniques might be used such as property based attestation or any other remote attestation scheme if the client implements it. The remote attestation agent observes the request and reports the information accordingly.

In our case, the communication between different stakeholders is performed in the following steps (cf. Figure 4):

- The hospital site (server) has the object (medical record) that may be requested by a client medical application (client).
- Upon request for the medical data, The hospital evaluates its access control policy [2] in order to determine whether the client medical application is authorized to access the object and verifies the capabilities of the client platform to make sure that it is capable of handling the access control and usage control requirements associated with the object that has been requested.
- Afterwards, the object and its associated meta data are transfered to the client via a secure connection.
- The meta data consists of a UCON policy and a file which contains the attributes of the object referred in the UCON policy. These attributes are described as (name, value) pair in XML format.

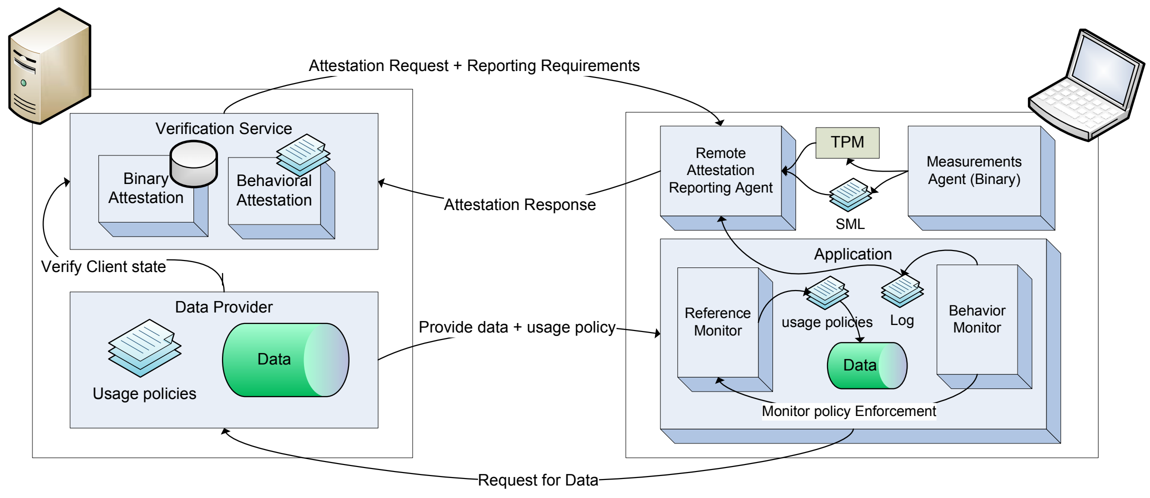
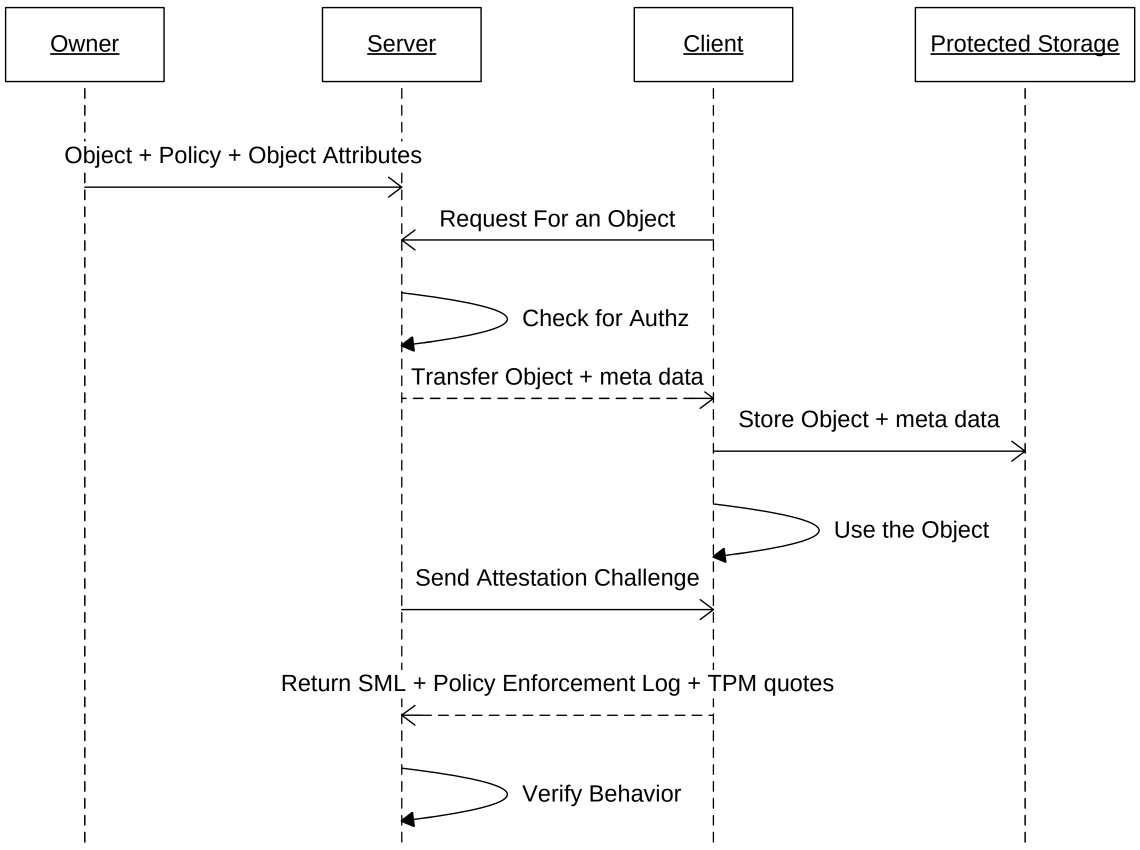


FIGURE 3. Target architecture

- The protected medical object and its meta data are stored in protected repository on the client platform. During the usage of the medical data by the corresponding medical application, the associated UCON policies are enforced.
- The hospital can later perform an attestation of the runtime behavior of the health care application regarding the correct enforcement of its UCON policy.



Owner
Server
Client
Protected Storage
Object + Policy + Object Attributes
Request For an Object
Check for Authz
Transfer Object + meta data
Store Object + meta data
Use the Object
Send Attestation Challenge
Return SML + Policy Enforcement Log + TPM quotes
Verify Behavior

FIGURE 4. Interaction between server and the client during behavior attestation

In addition to the versatile remote attestation agent, an enterprise healthcare application has been deployed on the client platform which enforces the usage control requirements associated with the objects originating from the server. The application has two major components which enable policy enforcement and remote attestation. These components are known as the *reference monitor* and the *behavior monitor*. The reference monitor enforces the UCON policies that are received from the server along with the protected medical object. The behavior monitor measures the policy enforcement behavior of the reference monitor to enable the server (or challenger) to attest the runtime behavior of the target application. The behavior monitor uses *trusted logging* mechanism to store and protect the behavior measurement logs. A trusted computing base is the set of all hardware and software components that are responsible for enforcing security policies on a platform [9] and whose untrusted behavior can not be detected. The behavior monitor is part of our trusted computing base.

The object and its associated meta data are transferred to the client medical application in an secure manner encrypted with the AIK public key so as to prevent the object and its associated policy from being tampered in transit from the server to the client. The server attests the core system components that are detrimental to the correct enforcement of the UCON policy including the target application and its reference monitor. The secret of the key with which the object and its meta data are encrypted by the server is known only to the reference monitor of the target application. Consequently, the object can only be decrypted and accessed by the reference monitor of the target application and ensures the protection of the supplied object from untrusted applications. After the provisioning of the object to the client platform the server can perform behavioural attestation on the client platform specifying the attestation requirements.

The following sections describe different essential components of our target architecture in greater detail.

3.1. **Reference monitor.** The reference monitor in the target application on the client platform is responsible for controlling usage of the object by enforcing the UCON policies. Figure 5 shows how various components of the reference monitor interact during the enforcement of the UCON policy. When an object is received by the application on the client platform, it is registered with the reference monitor and is added to the protected repository along with its associated meta data. The object can only be accessed by subjects or users after it is registered with the reference monitor.

We bind the protected repository with the TPM in order to secure it. We have used Trusted Java libraries [10] to bind the protected repository with the platform. Trusted Java provides a high-level API to Java programs for communicating with the TPM. When some data is bound to a specific TPM, it can only be decrypted on the hardware platform to which it is bound. By binding the data with a specific binding key, it can be ensured that only the application which has the private portion of the key and which is operating on the specific TPM's platform can decrypt the data. Thus, using this approach, we can ensure that the protected object can only be decrypted 1) by the authorized reference monitor, i.e., one which has access to the specific binding key, and 2) on the same platform for which the object was released.

Access to the protected repository is maintained by a component known as *access manager* which is part of the reference monitor. The protected repository contains the objects and their associated meta data in an encrypted form (cf. Figure 6). Subjects can only request for exercising rights on objects after they have been successfully authenticated by various means such as login password, biometrics, etc. An authentication mechanism creates meta data consisting of the attributes of the subject when a subject has been

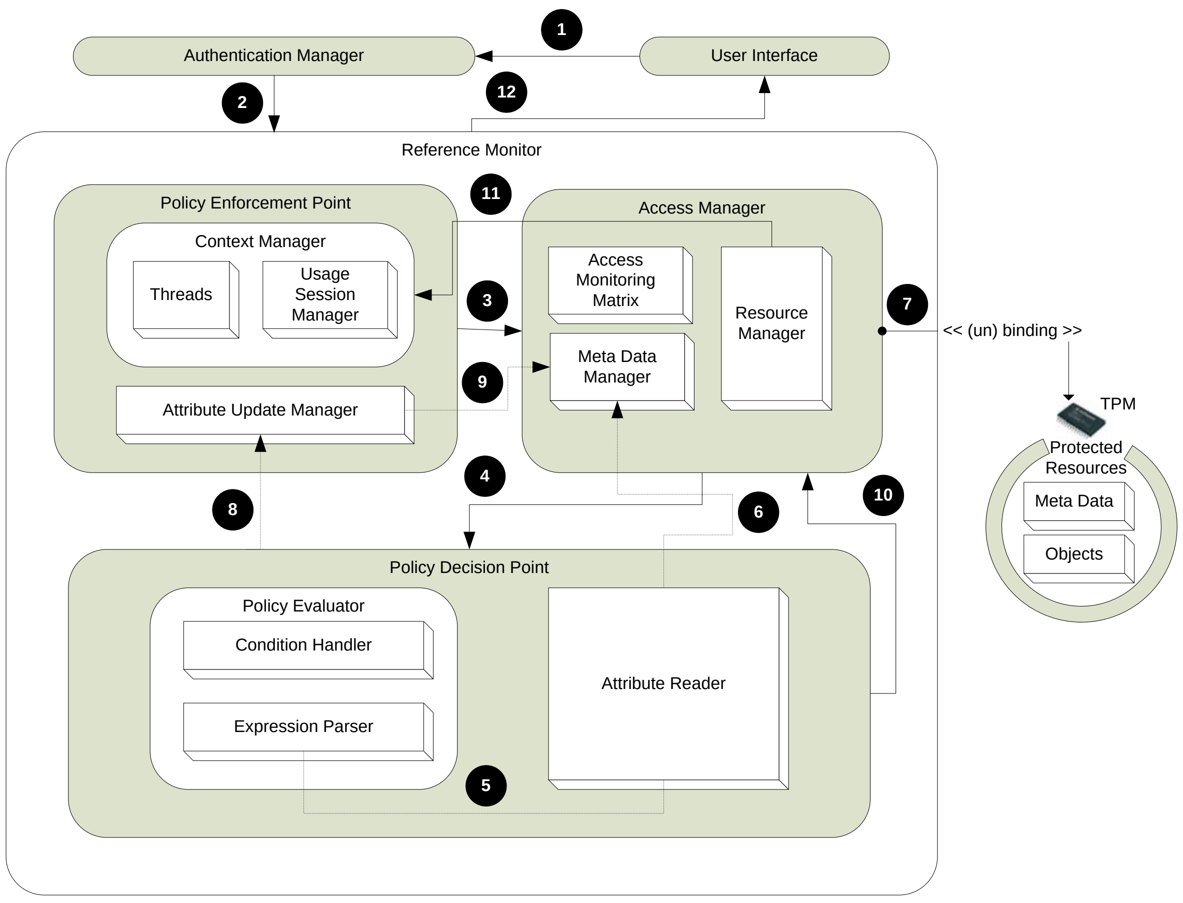


Figure 5. Details of the reference monitor

successfully authenticated. The reference monitor gets the meta data of the subject which contains the attributes of the subject, from the authentication mechanism (cf. Figure 6), and adds it to the protected repository by the access manager, this procedure is depicted in the first four steps shown in Figure 6. The attributes of subjects are similar to those of objects and are also stored in XML files. The attributes of subjects must be supported by a certificate which certifies that the attributes are trustworthy.

The reference monitor is virtually non by-passable, as only the reference monitor has the capability to decrypt the objects stored in the protected repository. Thus, every subject that needs to access objects must request the reference monitor to grant access to the object. The reference monitor then parses and enforces the UCON policies for the use of the object before granting access to the subject on the object.

The reference monitor evaluates and enforces UCON policies using policy decision point (PDP) – which decides about the usage of an object by a subject, and policy enforcement point (PEP) – which enforces the decisions of the policy decision point.

The policy enforcement point manages all the usage sessions that are active within the application at any one instance of time, and keeps track of the states transition taking place within each usage session. A usage session consists of various UCON states. The policy enforcement point is responsible for starting and ending usage sessions, allowing or denying a state transition to occur, and for performing attribute updates specified in the UCON policy.

The policy enforcement point handles requests of subjects for exercising rights on objects, but is dependent on the access manager for gaining access to the object. When a

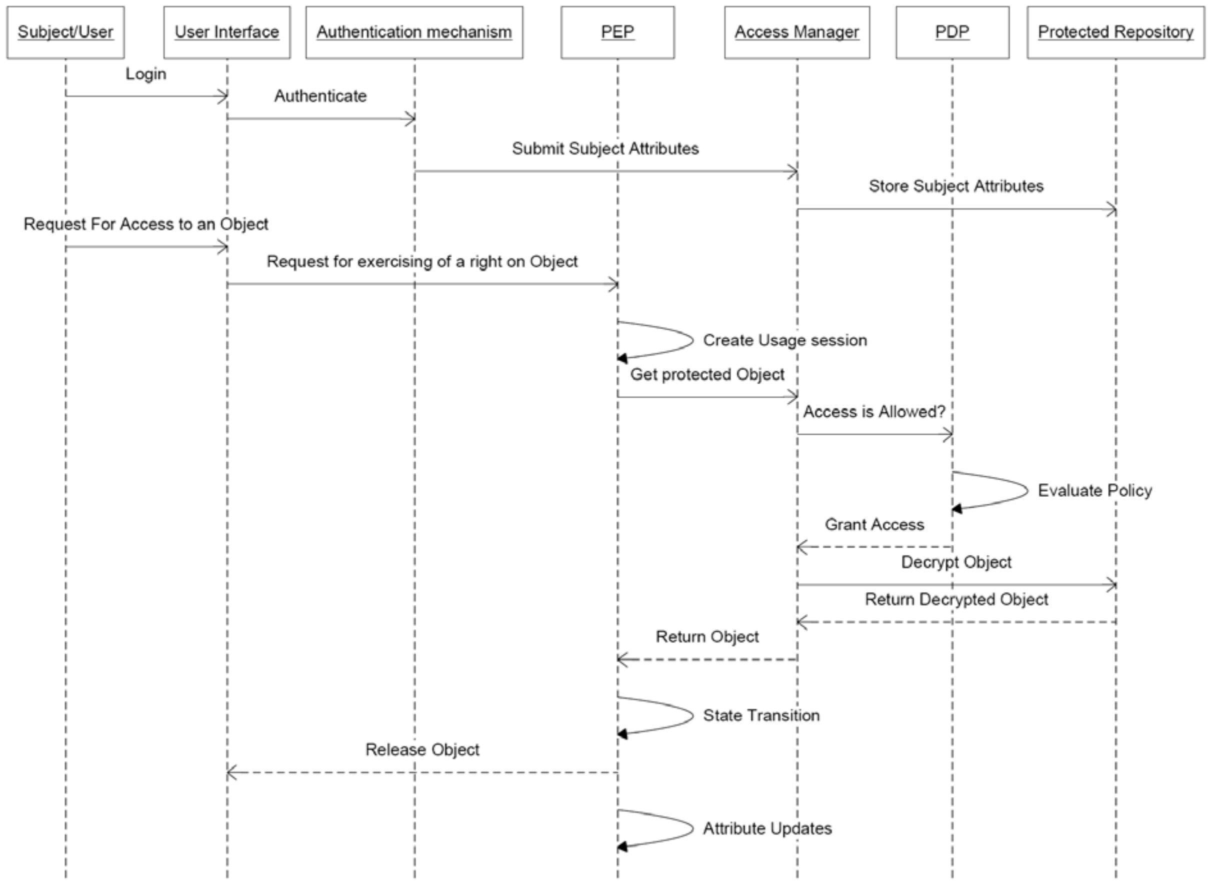


FIGURE 6. Enforcement of UCON policy by the reference monitor

subject tries to access an object the policy enforcement point creates a usage session with the subject in the requesting state and asks the access manager to grant access to the object.

The access manager as the name suggests manages access to the encrypted repository that contains objects and their meta data. It is the only component which is capable of encrypting and decrypting the objects stored within the protected repository, consequently any change to the object or meta data (e.g., attribute updates) is carried out through the access manager. When the policy enforcement point requests the access manager to get access to an object, the access manager queries the policy decision point in order to get a decision on whether access should be granted or denied. If access is allowed then access manager decrypts the object and provides it to the policy enforcement point which grants access to the user.

When a state transition into a given state occurs the policy enforcement point performs the attribute updates that are specified in the policy associated with the corresponding state. The policy enforcement point also keeps track of all subjects and objects, that are active within the application at any one instance of time.

The policy decision point evaluates the UCON policy associated with an object in order to determine whether access has to be granted or not. The policy decision point is also responsible for parsing and evaluating UCON policies. When the conditions specified in the UCON policy are satisfied, the policy decision point directs the access manager to grant access to the object.

```
 1 <PolicyEnforcementLog>
 2   <UsageSession>
 3     <SubjectID>Doctor</SubjectID>
 4     <ObjectID>PatientRecord</ObjectID>
 5     <Right>Read</Right>
 6     <StateTransition>
 7       <TimeStamp>06-13-2009 09:46:43 AM</TimeStamp>
 8       <PreviousState>Initial</PreviousState>
 9       <Policy>Requesting</Policy>
10       <CurrentState>Requesting</CurrentState>
11       <AttributeUpdateActions>
12         <TimeStamp>06-13-2009 09:47:34 AM</TimeStamp>
13         <TargetAttribute>
14           <ObjectID>PatientRecord</ObjectID>
15           <AttributeName>NoOfTimesRequested</AttributeName>
16         </TargetAttribute>
17         <InitialValue>0</InitialValue>
18         <InitialState>Trusted</InitialState>
19         <UpdatedValue>1</UpdatedValue>
20         <UpdatedValueEvaluation>
21           <Expression>
22             <FunctionID>
23               edu.serg.mbar.algorithms.functions.Add
24             </FunctionID>
25             <Inputs>
26               <Input Type="ObjectAttribute" State="Trusted">0</Input>
27               <Input Type="Constant" State="Trusted">1</Input>
28             </Inputs>
29             <Outputs>
30              <Output>1</Output>
31             </Outputs>
32           </Expression>
33         </UpdatedValueEvaluation>
34       </AttributeUpdateActions>
35     </StateTransition>
36     <AccMatrixAction Type="Create" TimeStamp="06-13-2009 09:47:53">
37      ...
```

FIGURE 7. A policy enforcement log created by the behavior monitor

The following section gives an overview of how the enforcement behavior of the reference monitor is measured using the behavior monitor. Afterwards, how measured behavior is verified at the challenger end, is detailed.

3.2. **Behavior monitor.** Behavior monitor is responsible for recording and logging the enforcement behavior of the reference monitor during the evaluation of a UCON Policy. The behavior monitor is part of our trusted computing base and thus, it is trusted by the challenging party.

All enforcement behaviors are recorded in XML format called the behavior enforcement log. This log contains fine grained details about the enforcement of the UCON policy. The behavior monitor records the policy enforcement behavior with the help of hooks inserted into the procedures which are responsible for enforcing the UCON policy and carry out state transitions and attribute updates. In order to monitor the enforcement of UCON policies, the behavior monitor logs *state transition behavior, attribute update behavior* and *active subject/object behavior*.

These behaviors are helpful in determining whether the enforcement of UCON policies was carried out in a trustworthy manner. Figure 7 shows an example of the enforcement log created by the behavior monitor. The policy enforcement log begins with the `<PolicyEnforcementLog>` markup tag (line 1). The policy enforcement log contains the details of all the usage sessions that were initiated by the application on the client platform. These usage sessions show how UCON policy was evaluated and enforced on the client platform. The policy enforcement log contains various `<UsageSession>` tags which contains the details of a usage session (line 2). A usage session is initiated whenever a subject requests a right on an object (cf. Figure 7 – lines 3, 4 and 5). The usage session tag contains various `<StateTransition>` tags (line 6) these contain the details of the states and state transitions that took place within a usage session during the policy enforcement process. The following subsections describe in detail the contents of these logs.

3.2.1. *Active subject/object behavior.* The behavior monitor records the behavior all of the subjects and objects that are active within the application at any one instance of time. When subjects and the objects become active, i.e., a right is granted to the subject on an object then they are added to the Access Monitoring Matrix (AMM). Therefore the addition and removal of the subject, the object and right from the AMM is recorded by the behavior monitor.

A subject $s$, an object $o$, or a right $r$ is added to the AMM when a state transition to *accessing* state is made by the subject $s$ while exercising right $r$ on the object $o$. Line 36 of Figure 7 is an example of how changes to the AMM are recorded. There are two types of actions that are performed on the AMM 1) *Create* where a subject, an object or a right is added into the AMM and 2) *Remove* where a subject, an object, or a right is removed from the AMM when a subject ends access to an object or when access is revoked from the subject.

3.2.2. *State transition behavior.* State transitions are allowed only after the policy associated with the given state is evaluated successfully and the conditions specified in the policy are satisfied. Therefore, to monitor state transition behavior, the behavior monitor collects information about the evaluation of the conditions in the policies associated with the specified state. The following information is logged in order to monitor state transitions; 1) $Time$ – of the state transition (cf. Figure 7 – line 7), 2) $PreviousState$ – the state before the state transition took place (cf. Figure 7 – line 8), 3) $CurrentState$ – the state after the state transition or the new state (line 10), 4) *Policy* – the type of policy that was evaluated after the successful evaluation of which the state transition took place, for example *Requesting*, *PermitAccess*, *DenyAccess*, *RevokeAccess*, or *EndAcces* (line 9) and 5) $ConditionEvaluation$ – that provides information about the functions that were used to evaluate the expressions that make up the conditions in the policy, the inputs of those functions and their outputs.

3.2.3. *Attribute update behavior.* The attribute updates are also specified in the policy associated with a state. To monitor the enforcement of policies, the behavior monitor records how the attribute updates are performed in various states during a usage session. Line 11 in Figure 7 shows how attribute updates are recorded by the behavior monitor. The information recorded by the behavior monitor pertaining to attribute updates includes 1) $Time$ – the time when the attribute update was performed (cf. Figure 7 – line 12), 2) $TargetAttribute$ – the attribute which was updated (cf. Figure 7 – line 13), 3) $InitialValue$ – the value before the attribute Update was performed (cf. Figure 7 – line 17), 4) $InitialState$ – e.g., trusted or untrusted (cf. Figure 7 – line 18), 5) $UpdatedValue$ – the updated value of the attribute after the attribute update action is performed (cf. Figure 7 – line 19), 6) $EvaluationoftheUpdatedvalue$ – how the updated value was evaluated, this information is helpful in analyzing how the expression stipulated in the UCON policy was evaluated. Thus, this section of the log contains the functions used to evaluate the expression, the inputs supplied to these functions and the output generated by these functions are logged (cf. Figure 7 – lines 20, 25, 29).

The following section provides an overview of how the enforcement behavior is verified by the challenger in order to determine whether the UCON policy was enforced in a trustworthy manner.

3.3. **Behavior verification.** The verification process consists of two major steps. The first step consists of the binary attestation of the application, so as to verify that the binary of the application is trusted, and has not been manipulated or modified maliciously. In order to attest the binary of the medical application, IMA is used on the client platform to measure the executable of the application. The Behavior Monitor on the client platform measures the binary of the application by writing a request to the securityfs, i.e., `/sys/security/measure`. The second step consists of the analysis of the policy enforcement log, in order to verify that the behavior of the application is trustworthy.

The semantics of each of individual entries in the log are verified in order to determine whether the UCON policy has been enforced correctly or not. The entries in the enforcement log describe the attribute update actions, state transition and the changes made to the AMM. Each of these entries contain different kinds of information. Therefore, each of these entries are verified differently.

The state transitions in the enforcement log are verified by analyzing the UCON policy of the concerned object, i.e., the object which has been used in the usage session. A state transition is said to be trusted when the right type (i.e., *Requesting, PermitAccess, EndAccess, RevokeAccess* and *DenyAccess*) of policy has been evaluated before the state transition took place, and also that the successful evaluation of the policy resulted in the state transition. Furthermore, the functions used in evaluating the policy are also verified along with their inputs and outputs so as to make sure that the functions used for evaluation were correct and that they behaved in a trustworthy manner.

Each state transition entry contains various attribute updates. Attribute updates in the policy enforcement log are verified against the *Attribute Update* predicates of the UCON policy. The Values that are verified in this regard is the nature of the update, i.e., *Preupdate*, *Postupdate*, *Outdate* to make sure that the right kind of updates were performed. After looking at the type of attribute update the initial value of the attribute is verified. In order to determine the trustworthiness of the attribute updates, the functions used to evaluate the updated value are analyzed along with their corresponding inputs and outputs. Attribute updates are trusted only if the right kind of attribute updates were performed and that the correct functions were used to evaluate the expressions and that the inputs given to the functions were trustworthy. The output from the functions is also verified in order to make sure that the functions behaved in a trustworthy manner.

Changes to the AMM are also verified during the behavior verification mechanism. If a subject made a transition to the accessing state then such a state transition must be followed by an AMM action `Create` which creates the record for the concerned subject, object and right in the matrix. On the other hand if a subject ends access or access is revoked from the subject then such state transitions must be followed by an AMM action `Remove` which removes the subject, object, or the right from the matrix. Active subject object behavior is said to be trusted if all of the actions performed on the AMM are correct according to the state transitions that occurred during the various usage sessions on the client platform.

4. **Conclusion.** In this paper, we have presented a high-level behavior specification and attestation framework, which can be built on top of various low-level attestation techniques and realized in various enterprise information systems to satisfy modern information security requirements. In our approach, the behavior of a model is attested rather

than a hardware or software platform. We selected UCON as an example target model as it captures continuous access control over objects in a distributed and dynamic computing environment. We have described the implementation issues involved in the realization of a dynamic attestation system based on our MBA approach.

The product of our work presented here is a high-level framework which can utilize different low-level techniques in a supporting manner for the purpose of attesting a remote platform. Behavior is associated with different components of a policy model. The behavior of each of these components can be attested separately at runtime and the aggregate of these behaviors can be used to measure the trustworthiness of the remote platform. Trust is thus associated with the dynamic behavior of a policy model instead of static measures such as hardware or software configurations or properties of the remote platform.

Although we would like to vouch for the integrity of the attestation information by extending them into the PCR of the TPM and use these aggregates to verify the integrity of the information reported during behavioural attestation, however the continuous nature of the usage of an object mandates requires these aggregates to persist over several system boots as compared to being volatile.

**Acknowledgment.** This work has been supported by the Institute of Management Sciences research grant number RG-CS-07001 and National ICTR&D Fund research Grant No. ICTRDF/TR&D/2008/45 to Security Engineering Research Group, Peshawar, Pakistan.

## REFERENCES

[1] J. Park and R. Sandhu, Towards usage control models: Beyond traditional access control, *Proc. of the 7th ACM Symposium on Access Control Models and Technologies*, New York, NY, USA, pp.57-64, 2002,

[2] S. Osborn, Mandatory access control and role-based access control revisited, *Proc. of the 2nd ACM Workshop on Role-based Access Control*, New York, NY, USA, pp.31-40, 1997.

[3] J. Joshi, E. Bertino, U. Latif and A. Ghafoor, A generalized temporal role-based access control model, *IEEE Transactions on Knowledge and Data Engineering*, vol.17, no.1, pp.4-23, 2005.

[4] R. Sandhu, Rationale for the RBAC96 family of access control models, *Proc. of the 1st ACM Workshop on Role-based Access Control*, New York, NY, USA, 1996.

[5] *Trusted Computing Group*, http://www.trustedcomputinggroup.org/, 2000.

[6] S. Pearson, *Trusted Computing Platforms: TCPA Technology in Context*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[7] T. Jaeger, R. Sailer and U. Shankar, PRIMA: Policy-reduced integrity measurement architecture, *Proc. of the 11th ACM Symposium on Access Control Models and Technologies*, New York, NY, USA, pp.19-28, 2006.

[8] X.-Y. Li, C. X. Shen and X.-D. Zuo, An efficient attestation for trustworthiness of computing platform, *IIH-MSP*, pp.625-630, 2006.

[9] P. Kanerva, *Anonymous Authorization in Networked Systems: An Implementation of Physical Access Control System*, Master Thesis, Helsinki University of Technology, 2001.

[10] *Trusted-Java, Jsr321: Trusted Computing Api for Java(tm)*, http://jcp.org/en/jsr/detail?id=321, 2009.