Enforcing DRM Policies Across Applications

Srijith K. Nair, Andrew S. Tanenbaum
Dept. of Computer Science
Vrije Universiteit
1081 HV Amsterdam
The Netherlands
{srijith,ast}@cs.vu.nl

Gabriela Gheorghe, Bruno Crispo
Dept. of Information and Communication Technology
University of Trento
Italy
{gabriela.gheorghe,crispo}@disi.unitn.it

ABSTRACT

In this paper we present Trishul-UCON (T-UCON), a DRM system based on the UCON_{ABC} model. T-UCON is designed to be capable of enforcing not only application-specific policies, as any existing software-based DRM solution does, but also DRM policies across applications. This is achieved by binding the DRM policy only to the content it protects with no relations to the application(s) which will use this content. Furthermore, to guarantee that the policy is continuously enforced, we designed T-UCON as a JVM-based middleware that mediates the usage requests of any Java application to the protected content. Each request is granted or denied according to the content policy. We illustrate the unique features of T-UCON by using typical examples of DRM policies such as the *pay-per-use* and the *use only N times* scenarios. Preliminary results on the overhead of our solution are also provided.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*

General Terms

Security, Design

Keywords

Digital Rights Management, Policy Enforcement, Access Control, Usage Control

1. INTRODUCTION

After being *in fashion* for some years now, DRM is currently approaching a more mature phase, gradually attracting a steadier research community. This trend is partially reflected in the industry too. Despite lesser emphasis compared to early days, there are still many companies [2] highly interested in flexible, cheap and secure DRM technologies. This is motivated by the indisputable fact that an increasing amount of digital content is produced every day and some of it needs to be protected in terms of its distribution and consumption.

Broadly speaking, existing DRM solutions can be classified as hardware-based and software-based. Trying to determine which one is better in terms of security alone could be misleading since each of them serves different needs. In practice the main discriminant between the two is the business model of the distribution system rather than their actual security strength.

Hardware-based solutions (e.g., Zune, iPod, etc.) present closed systems consisting of *compliant* devices that are by construction made to conform to the DRM specifications. The security of these systems relies on the impossibility to *fake* a compliant device and on the admission control protocol that allows only compliant devices to interact with other compliant devices. An advantage of these solutions is the simplicity of the design, but the disadvantage is the cost. Building devices impossible or hard to fake or break is very expensive, thus in practice an acceptable compromise between manufacturing costs and estimated loss of revenue due to DRM failure is considered.

On the other hand, software-based solutions (e.g., iTunes Fairplay, Adobe DRM, etc.) are cheaper and more flexible, since they do not require special hardware and they can share a computer with other non-DRM applications. These solutions build a software-protected environment (e.g. player, reader, etc.) within which (and only within which) the protected content can be consumed. Other applications cannot access the protected content since it is typically encrypted with a decryption key embedded in the protected environment. Software-based solutions are secure, assuming the operating system is trustworthy. These software-based solutions are the best choice in all those scenarios where the content provider does not have control over the hardware used by the consumers, and furthermore it has its own interest at heart to make the content available to as many device types as possible. This paper is about software-based DRM solutions.

Motivation

Despite being more flexible than hardware-based systems, current software-based solutions still suffer from many drawbacks that limit the type of DRM policies they enforce. Due to their design, all existing solutions cannot, for example, implement cross-application DRM policies. Thus typical

DRM'08, October 27, 2008, Alexandria, Virginia, USA.

use only N time policies like '*play the song "Imagine" no more than 5 times*' cannot be enforced. What is now enforced are rather policies like '*play the song "Imagine" no more than 5 times using 'ThisPlayer''*, thus binding the policy to a specific application. Similarly, **pay-per-use** policies like '*the cost of playing the song "Imagine" is 50 cents the first 10 times, then 5 cents for the next 10 times and 1 cent for the next 100 times*' are impossible to implement if one tries to enforce them at the song level rather than for a specific application.

Contributions

There have been previous work done on modeling DRM architectures and associated policy languages [2, 10]. However there are fewer implementations of such systems. In this paper we present the design and implementation of Trishul-UCON, an open and generic software-based architecture that enforces DRM policies. In particular, we will show how Trishul-UCON can be used to enforce DRM policies both application specific and more importantly across applications. After providing solutions for the two examples mentioned above we show that Trishul-UCON also enforce DRM policies that use *obligations*. Preliminary performance tests confirm the feasibility of our approach.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the UCON_{ABC} model [10], a usage control model, focusing on its relevance to DRM systems. In Section 3 we present the design and implementation of Trishul-UCON, while in Section 4 we analyze some examples of DRM policies that have been implemented using Trishul-UCON. Assumptions regarding trusted system are briefly mentioned in Section 5. Performance evaluation of our prototype implementation is presented in Section 6. We discuss related work in this area in Section 7 and present our conclusions in Section 8.

2. UCON_{ABC} MODEL AND DRM

In this section we briefly explain the UCON_{ABC} model and how it can be used to model various DRM scenarios.

2.1 The UCON_{ABC} Model

The UCON_{ABC} model [10] was introduced to extend traditional access control to consider the problem of authorization not only at the time of access to a resource but also during its usage.

The main components of the UCON_{ABC}[1] model, as shown in Fig. 1, are the *Subjects*, which wishes to assert various *Rights* over certain *Objects*. Subjects and objects are endowed with *Attributes* that capture the properties and/or capabilities of these components. The decision to determine the rights of the subject on the object are based on - *Authorization* (A) where attributes of subjects and objects are checked in order to make authorization decisions, *oBligation* (B) where checks are performed to ensure that certain actions are performed by the subject and *Conditions* (C) where environmental (system) attributes are checked as a part of the usage decision process.

The generality required by complex usage control scenarios is achieved by adding the notion of decision continuity and subject and object attribute mutability to the model.

[1]where ABC stands for Authorizations, oBligations and Conditions respectively

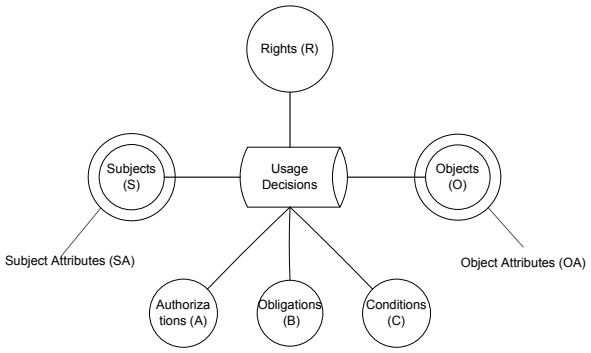


Figure 1: Components of the UCON_{ABC} model

The decisions could be made before the usage is permitted (*pre*) or during the usage session (*on*), while the attributes can be updated before (*pre*), during (*on*) or after (*post*) the usage has been granted. These notions of decision continuity, attribute mutability and the checks imposed by UCON_{ABC} enable it to meet the requirements of generic DRM models [5, 7].

2.2 Modeling DRM

As with the generic UCON_{ABC} model, a DRM system consists of two main entities: subjects and objects. Subjects are users or software agents (e.g. a multimedia player) being executed on behalf of the user. Objects are data files, such as music or video files, whose access and use are subject to various restrictions (when, where or how they can be employed).

Subject attributes are properties and capabilities associated with the user that allow him/her to exercise rights over objects. Attributes relevant to DRM systems include credit card number, prepaid credit balance and similar financial details of the user. Object attributes are properties associated with objects, like the cost of the media file, meta-data like artist name, bitrate of the MP3 file, etc. Properties like the remaining play count and age of the file are also considered as object attributes in our system. Conditions are used to express environment variables (e.g., date, time) used to evaluate DRM policies. Obligations express actions that must be executed to use the object (e.g., accept the license first).

In the rest of this section we show how to use the UCON_{ABC} model to express the examples of application-independent policies introduced earlier on.

2.2.1 Pay-per-use

Conceptually a pay-per-use service is one of the simplest DRM scenarios. An object has a value associated with it, which forms its attribute. The subject's credits form his attribute. The policy associated with the object states that every use (say view) of the object requires the value of the object to be decremented from the credit balance of the user. To implement such policies the authorization for use of the object is required before the usage is allowed and the mutable attributes are updated as a *pre* update process. Using UCON_{ABC}, a generic pay-per-use policy can be modeled, using the notation in [10], as:

M is a set of monetary amount
$credit : S \rightarrow M$
$value : O \times R \rightarrow M$
$ATT(S) : credit$
$ATT(O, R) : value$
$allowed(s, o, r) \Rightarrow credit(s) \geq value(o, r)$
$disallowed(s, o, r) \Rightarrow credit(s) < value(o, r)$
$preUpdate(credit(s)) : credit(s) = credit(s) - value(o, r)$

where O is the object, S is the subject, R the right and $ATT()$ denote attributes. In our specific example the O is the song "Imagine", value changes over time being 50 cents for the first 10 times, 10 cents for the next 10 and 1 for the next 100.

2.2.2 Use N times

Next, let us consider the often-discussed DRM policy of 'use only N times'. Of course, the semantic of 'use' is application dependent. In our example it is "play". To be even more realistic, many content providers let anybody play 50% of the song as a means of advertising the song. We support this feature also in our example. So a "play" action counts as such only if at least 50% of the song has been played. In UCON_{ABC} such a policy can be modeled as:

B is amount of bytes
N is an integer
$size : S \rightarrow B$
$played : S \rightarrow B$
$plays\_left : O \rightarrow N$
$allowed(s, o, r_1) \Rightarrow plays\_left > 0$
$disallowed(s, o, r_1) \Rightarrow plays\_left \leq 0$
$allowed(s, o, r_2) \Rightarrow true$
$postUpdate(play(s), r_2) : played(s) + read$
$postUpdate(plays\_left(o), r_2) : plays\_left(o) - 1$; if $play(s) > size(o)/2$

Where r_1 is the *open* right and r_2 the *read* right and in this example o is the song "Imagine", and N is 5.

2.2.3 Metered payment

A membership-based metered payment DRM system presents a slightly different scenario. In such a system, the subject needs to be a valid member possessing an expense account to access the object and the expense of usage is dependent on the usage time duration. Thus the membership is the object attribute while the subject attributes is the cost per unit time. The usage control can be modeled as follows:

M is set of monetary amount
ID_{mem} is a set of membership IDs
$Time$ is a current usage unit of time
$expense : S \rightarrow M$
$usage : S \rightarrow Time$
$member : S \rightarrow ID_{mem}$
$value_t : O \times R \rightarrow M$, cost of r per unit time
$ATT(S) : member, expense, usage$
$ATT(O, R) : value_t$
$allowed(s, o, r) \Rightarrow member(s) \neq \phi$
$disallowed(s, o, r) \Rightarrow member(s) = \phi$
$postUpdate(expense(s)) : expense(s) + value_t(o, r) \times usage(s)$

Thus we saw that UCON_{ABC} provides us with a sufficiently generic model to express DRM policies. However, as described in Section 7, the current state of the art does not present any actual system that implement DRM systems using the UCON_{ABC} framework. In the next section we present the design and implementation of Trishul-UCON (T-UCON), an UCON_{ABC}-based architecture for implementing DRM systems, which we believe is the first in this area.

3. TRISHUL-UCON ARCHITECTURE

Trishul-UCON[2] is designed and implemented by extending the Java Virtual Machine (JVM) and hence can work for all Java applications. T-UCON provides, among other features, the mechanism to intercept and analyze Java methods calls to decide on whether to allow them or not and a mechanism to convey this decision back to the JVM.

Unlike traditional DRM solutions that restrict policy enforcement to specific applications, T-UCON is capable of enforcing policies independent of and across applications. This is done by associating the DRM policies to objects, mediating any access to these policy-restricted objects using the T-UCON system and by capturing the state of the system across application runs in the object and subject attributes.

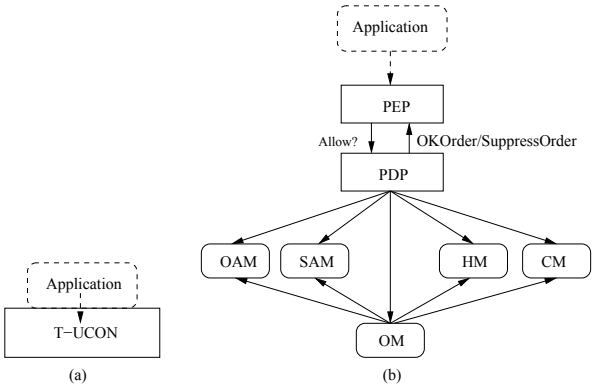


Figure 2: A schematic representation of (a) T-UCON intercepting application methods calls and (b) various components of the Trishul-UCON architecture

Figure 2 provides a high level overview of the various components of T-UCON and their relationship with each other. The Policy Enforcement Point (PEP) intercepts Java method calls made by the applications that are of interest to the DRM system. Once the relevant calls are intercepted, the control is passed to the Policy Decision Point (PDP). It is the responsibility of the PDP to decide whether the application call can be allowed to proceed or not. To make this decision, the PDP consults the policy associated with the object by calling the Object Attribute Module (OAM). The OAM also provides an interface to query and update object attributes. The Subject Attribute Module (SAM) provides an interface for querying and updating the subject attributes, while the Condition Module (CM) provides a similar functionality for the system attributes. Once a decision has been made by the PDP on whether to allow the

[2]The name 'Trishul-UCON' is derived from 'Trishul' [9], an existing extension to the JVM to implement enforcement of information flow control.

action, it is communicated to the PEP. The PEP forwards this decision to the JVM, which then halts the action by throwing an exception, if needed.

In the rest of this section, we look at each of these components in detail.

The Policy Enforcement Point (PEP)

When T-UCON is launched, the PEP registers all the Java method calls (actions) that are of interest to the DRM system. For generic DRM scenarios, these include the file open and read method calls which need to be mediated and any network based calls which are denied by default.

When an application tries to execute any of the restricted methods, the PEP intercepts it and passes control to the PDP, sending along all the available information regarding the method call. Once the PDP makes a decision on whether to allow the call or not, the decision is passed to the PEP, which informs the application of the steps to take.

The Policy Decision Point (PDP)

Once an action is intercepted by the PEP, it is passed on to the PDP. The PDP is essentially responsible for ensuring that the required authorizations, obligations and conditions are met for the method call to proceed and if they are not, to disallow the action. A decision on whether to allow or deny an action is conveyed back to the PEP.

PDP and OM need the capability to query the attributes and update them according to the policy. To this end, the PDP calls the corresponding functions provided by the Object Attribute Module and the Subject Attribute Module.

The Obligation Module (OM)

While most of the authorization and condition requirements are sufficiently straightforward for the PDP to check and enforce by itself, the complicated nature of obligations warrants a dedicated entity - the Obligation Module (OM). When the PDP encounters an obligation requirement (e.g. user needs to accept the license agreement) in the object policy, it passes the obligation check to the OM for handling. The OM implements all the logic associated with the obligations requirement. Such a modular architecture allows all the logic required to interpret, check and enforce the obligations to be completely contained within the OM.

If the policy specifies an ongoing obligation, the OM registers a timer at the PDP, associating it with a unique identifier to identify the specific obligation. When the timer fires, the PDP passes the control to the OM to check for the obligation compliance. Figure 3 provides an overview of the obligation enforcement process.

The History Module (HM)

Many DRM policies requires history based decisions, since they typically span across several usage sessions of the object. We implement the history by associating to each object a state that is global with respect to the applications and the time.

The History Module provides two distinct functionalities - a convenient mechanism to log events/actions that have occurred and an efficient mechanism to query these logged events. As seen from Figure 2, the HM is called from the PDP as well as the OM to log and query actions, associated decisions as well as any other relevant checks performed prior to making the decision.

In order to provide an efficient service, the current implementation of the HM logs only the essential details including a timestamp, the method's name, the identity of the logging entity (PDP/OM), the decision returned (if PDP is the one logging), the state of obligation requirement (if OM is logging) and the identity of the object.

The Object Attribute Module (OAM)

The OAM provides an interface for the PDP and the OM to query and update object attributes. This modular design allows the OAM to rely on the PDP and the OM to initiate the pre/ongoing/post updates to the attributes while freeing them from having to interpret the syntax of the attribute specification.

Since the policy associated with the object is similar in nature to an object attribute, the OAM is also designed to query this information. It should be noted however that the OAM itself is not responsible for interpreting the policy nor the state of the object as stored in the attributes. These are still the responsibility of the PDP.

The Subject Attribute Module (SAM)

The SAM provides a similar mechanism to query and update the subject attributes. These are again invoked by the PDP and the OM when they need to perform pre/ongoing/post attribute updates.

The Condition Module (CM)

System attributes are queried using the CM. These include the date, time and other system variables that can be considered as *conditions* in the UCON_{ABC} model. Since system attributes are in general immutable in nature, the CM does not provide mechanisms to update them.

4. USING T-UCON TO ENFORCE DRM POLICIES

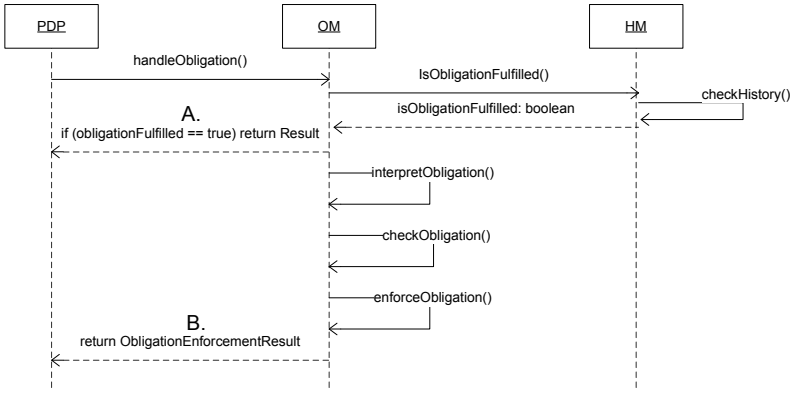
Now that the T-UCON architecture has been explained, in this section we revisit the DRM examples and explain how the architecture is used to enforce them.

4.1 Pay-per-use

When the application, on behalf of the user, tries to perform the read operation, it is intercepted and the control is given to the PEP to enforce the policy. The intercepted action is then forwarded to the PDP. The PDP queries the OAM to check the exact policy associated with the object. After interpreting the policy, the PDP queries the OAM again to read object attribute *value* and the SAM for the subject attribute (*credit*). It then decides on the authorization based on the the value of $credit(s)$ and $value(o, r)$. Figure 4 shows the details of the step involved in the process.

4.2 Use N times

On interpreting the policy, the PDP queries the OAM for the objects' use_left attribute. The action is allowed if this value if greater than 0. This implementation has the limitation however that once the use_left has reached 0, the object can never be opened, even for providing a preview of the content. If such an access is to be allowed, the following logic is used instead:



A. prior-to-usage obligation checking
B. prior-to-usage attribute update added to case A., by calling attribute modules

Figure 3: Working of the Obligation Module of T-UCON

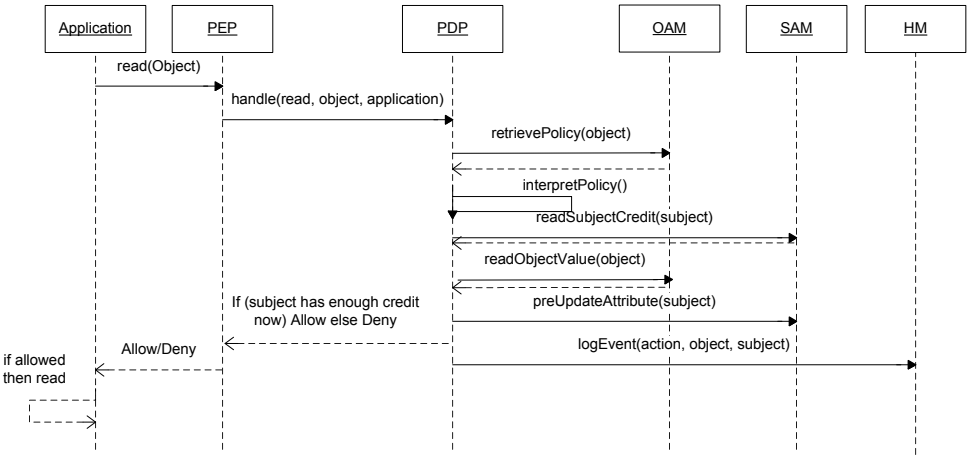


Figure 4: Implementing pay-per-use policy using T-UCON

$allow(s, o, r_2) \Rightarrow read(s) < size(o)/2 \vee use\_left > 0$
$disallow(s, o, r_2) \Rightarrow use\_left \leq 0 \wedge read(s) > size(o)/2$

The steps involved in implementing this modified policy using T-UCON is shown in Figure 5. As these steps show, since T-UCON performs an update of $plays\_left$ object attribute irrespective of which application is playing the file, the object policy is enforced across different applications.

4.3 Metered payment

As the formalization presented before shows, the key subject attributes are its membership ID, the total expense and the current usage time while the object attribute is its usage value in unit time. What exactly constitutes the action/rights (R) is scenario dependent. For example, in our implementation, a simple read of the object is used as R.

On parsing the policy and noting the metered payment restriction, the PDP asks the OAM to lookup the object attribute. At the same time, the subject attributes are queried using the SAM. The PDP then perform the necessary authorization checks based on which the right is allowed or disallowed. Once the method call has been executed, the PDP then performs the post update on the subject attribute *expense* by calling the SAM, as shown in Figure 6.

5. TRUSTED SYSTEM CONSIDERATIONS

The T-UCON architecture proposed here, being a pure software based solution, does not rely on any hardware functionality to perform policy enforcement. However, in order to ensure the integrity of the architecture, it is imperative to leverage on some trusted hardware technologies.

We assume that T-UCON is run on a trusted operating system, which in turn is run on a Trusted Platform Module (TPM) [3] supported hardware.

One way of assuring integrity of the T-UCON architecture is through the deployment of a trusted subsystem similar to the one proposed by Zhang et al. in [13]. Using such a system, a TPM's sealing and trust chain based attestation functionality are used to ensure that the objects can be accessed only by applications running inside T-UCON.

The actual mechanism involved in implementing this functionality is not considered as core part of our architectural framework, mainly because the power of the TPM devices are improving at a rapid pace and using a very specific mechanism would prevent us from using the latest available technology. For example, the use of dynamic root of trust (instead of the older static one as used previously in [13]) of the TPM, by exploiting the recently added Secure Virtual Machine extensions [8], tremendously increase the power of the TPM to verify and protect the integrity of the platform.

6. PERFORMANCE

The T-UCON architecture presented in the paper is designed with the expressed aim of being modular and generic enough to implement a wide range of DRM policies. Such a design however has the downside of introducing performance overhead. In this section we report the results of performance measurements conducted using our prototype implementation of T-UCON. The measurements were performed on a Intel Core 2 Duo 2GHz machine with 2GB RAM running Ubuntu 7.10 with a 2.6.22 Linux kernel.

We consider the specific example of pay-per-use policy discussed earlier. The read access to an MP3 file is allowed only if the credit remaining for the user is more than the per-use value of the file. Table 1 summarizes the results of the measurements.

Table 1: Performance comparison of T-UCON prototype in pay-per-view microbenchmark

Environment	time (ms)
Unmodified JVM, no policy	1
T-UCON, no policy	10
T-UCON, pay-per-use (XML)	1780
T-UCON, pay-per-use (txt)	240

Table 2: Performance comparison of T-UCON prototype in pay-per-view application run

Environment	time (s)
Unmodified JVM, no policy	503
Unmodified JVM, SM policy	503
T-UCON, no policy	504
T-UCON, pay-per-use (XML)	511
T-UCON, pay-per-use (txt)	508

When an unmodified JVM is used, with no policy enforced, the application was able to start reading from the MP3 file in just over 1 ms. When the application was run with T-UCON intercepting the method calls, an overhead of 9 ms was introduced. In this case again, policies were neither consulted nor enforced. Thus the overhead measured is due to the method call intercepting and control passing mechanisms involved in the architecture.

In the next experiment, the pay-per-use policy was enforced using T-UCON. The subject and object attributes were first represented using XML, to simulate the use of an XACML [1]-like language, resulting in an observable overhead of 1780 ms. Closer analysis revealed that more than 75% of this overhead is introduced by the process of reading, parsing and updating the XML files. In order to estimate a less biased overhead, simple text files were used next to represent the attributes of the subject and the object. The overhead reduced to a much lower value of 240 ms.

The low overhead observed in the second and last set of experiments show that the system overhead are dependent on the complexity of the checks involved and on the way the policy and attributes are represented and as such cannot be attributed solely to the architecture itself. Furthermore once the action is allowed and the necessary updates have been made, T-UCON does not perform any more mediation and hence the overhead is small when the application run is considered in its entirety.

To verify this, we next considered the case of playing a 5.6MB, 4.56 minute long MP3 file, again using the pay-per-use policy. Table 2 shows the time required to play the file for various test cases, averaged over 5 runs. The second row denotes the time taken for the unmodified JVM to play the file when a simple `grant read permission` is specified by the Java Security Manager, while the rest of the rows are similar to those in Table 1. Do note that the current Security Manager supports only simple access control policies and not usage control restrictions. The low overhead figures observed when using T-UCON supports our claim regarding the practicality of using our architecture.

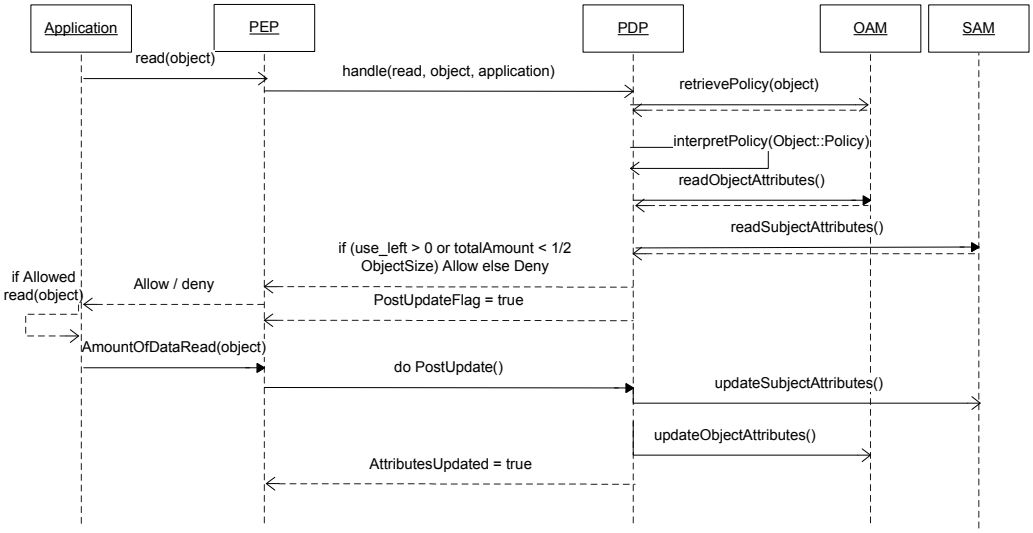


Figure 5: Implementing the 'play N times' policy using T-UCON

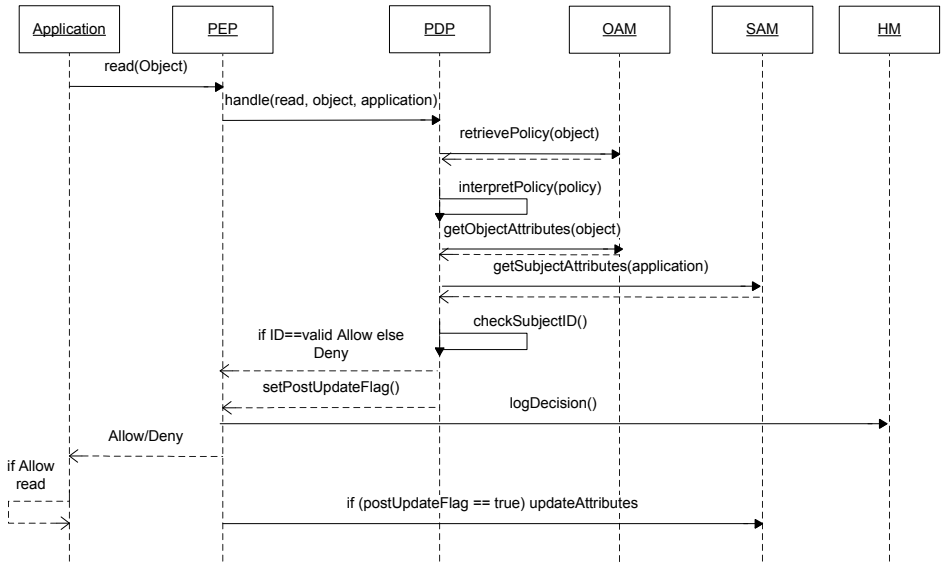


Figure 6: Implementing metered payment policy using T-UCON

7. RELATED WORK

Recent years have seen an increased interest in the area of enforcing usage control policies in distributed systems. Considering this area as a super set of the DRM enforcement studies, here we take a look at the existing proposals and highlight their differences compared to our approach.

Pretschner et al. [4], tackle the enforcement of usage control requirements in Service Oriented Architectures. The paper suggests a client-side architecture which is able to support domain separation and policy enforcement for various Java services or objects. The granularity of usage decisions supported by their architecture is at the level of applications, while T-UCON provides a very fine-grained control at the level of Java method calls. Furthermore in T-UCON, applications are not assumed to be trusted.

Aimed at mobile computing scenarios, the client-side enforcement approach proposed by Schaefer [11] considers the use of a reference monitor to enforce usage policies on the objects of interest. Although the architecture is similar to T-UCON, in their approach the monitor on the client side needs to be continuously updated with the information on a usage control server. Moreover, the work is purely theoretical in nature, while in this paper we present the design and implementation of a practical enforcement architecture.

A slightly different approach is taken by Zhang et al. [12] where they propose an authorization enforcement architecture for collaborative systems. While their focus is on the general collaborative systems, ours concerns enforcement on the consumer-side. Although typical UCON_{ABC} mechanisms like attribute updates and obligation checks are dealt with in detail, the proposal commits to the study of authorizations rather than usage control and hence among others, ongoing obligations are not considered. Trusted computing in usage control is approached in [13], and while the suggested architecture is similar to [12], the focus is on the integrity of inner security modules and details of specific enforcement scenarios are not presented.

Katt et al. [6] extends the original UCON_{ABC} model by adding the notion of *post-obligations*. Focusing on obligations from the point of view of subject, object and fulfillment time, the paper stresses an enforcement framework incorporating these aspects. However, in their architecture the PEP is embedded with the target application. This prevents policies from being enforced across applications and assumes a trusted application or the existence of a mechanism to safely direct application actions, neither of which are explained in detail in the paper. T-UCON, on the other hand, is firmly based on the premises of controlling untrusted applications and allows the policies to be associated with objects and enforced across applications.

8. CONCLUSION

In this paper we have presented the design and implementation of T-UCON, a generic software-based Digital Rights Management architecture. Unlike other DRM solutions, T-UCON is able to enforce policies associated with objects across multiple applications. Performance results show that the framework in itself adds little overhead though the checks associated with the DRM logic have the potential to introduce larger overheads.

9. ACKNOWLEDGMENTS

We would like to thank Sreejith K. S. and Arun P. Mohan for their help with the implementation of T-UCON. This work has been supported by NWO project ACCOUNT 612.060.319 and partially by the EU FP6 project GridTrust contract 0033817 and EU FP7-ICT-2007-1 project MASTER contract 216917.

10. REFERENCES

[1] Oasis extensible access control markup language. Website, 2008. `http://www.oasis-open.org/committees/xacml/`.

[2] Open mobile alliance digital rights management working group. Website, 2008. `http://www.openmobilealliance.org/Technical/DRM.aspx`.

[3] TCG TPM specification version 1.2. Website, 2008. `https://www.trustedcomputinggroup.org/specs/TPM/`.

[4] A. Berthold, M. Alam, R. Breu, M. Hafner, A. Pretschner, J.-P. Seifert, and X. Zhang. A technical architecture for enforcing usage control requirements in service-oriented architectures. In *SWS '07: Proceedings of the 2007 ACM workshop on Secure web services*, pages 18–25, New York, NY, USA, 2007. ACM.

[5] J. S. Erickson. Fair use, DRM, and trusted computing. *Commun. ACM*, 46(4):34–39, 2003.

[6] B. Katt, X. Zhang, R. Breu, M. Hafner, and J.-P. Seifert. Beyond UCON core models with general obligation model and continuity-enhanced policy enforcement engine. *ACM Symposium on Access Control Models and Technologies*, 2008.

[7] B. A. LaMacchia. Key challenges in DRM: An industry perspective. In *Digital Rights Management Workshop*, pages 51–60, 2002.

[8] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for TCB minimization. *SIGOPS Oper. Syst. Rev.*, 42(4):315–328, 2008.

[9] S. K. Nair, P. N. D. Simpson, B. Crispo, and A. S. Tanenbaum. A virtual machine based information flow control system for policy enforcement. *Electronic Notes in Theoretical Computer Science*, 197(1):3–16, 2008.

[10] J. Park and R. Sandhu. The UCON_{ABC} usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

[11] C. Schaefer. Usage control reference monitor architecture. *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*, July.

[12] X. Zhang, M. Nakae, M. J. Covington, and R. Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–36, 2008.

[13] X. Zhang, J.-P. Seifert, and R. Sandhu. Security enforcement model for distributed usage control. *sutc*, 0:10–18, 2008.