John S. Erickson
Hewlett-Packard Laboratories
Norwich, Vermont, USA
john.erickson@hp.com

Revised September 2002

OpenDRM: A Standards Framework for Digital Rights Expression, Messaging and Enforcement

> *...Build a platform, or set of protocols, so that it can evolve in any number of ways; don't play God; don't hardwire any single path of development; don't build into it a middle that can meddle with its use. Keep the core simple and let the application (or end) develop the complexity...No one can control how the system will evolve. No single individual gets to set the path the system will follow. It might evolve to follow a path, but it will evolve by the collective choice of many...*
> Lawrence Lessig, *Open Code and Open Societies* (1999) [OpenCode]

> *...There were people who saw that there was a need for such software, but each one thought that they were going to lock everyone into their system. And pretty much there would be no progress. They wouldn't explain to people what they were doing. They would have people using their thing; they couldn't switch to another, and they couldn't get another person to do [it] for them....it would be better if it was available to everybody than if there were all kinds of things that people were keeping only on one machine...I was much more concerned with the idea that it should be usable by everybody...*
> Donald Knuth (1990)

Introduction

The lack of open, accessible, interoperable standards for digital rights management has often been cited by stakeholders as a leading cause for the slow adoption of DRM technologies. The fact that layered standards can contribute to interoperability should be obvious; that DRM standards developed in an open environment can contribute to the public interest is a more subtle, but equally important point.

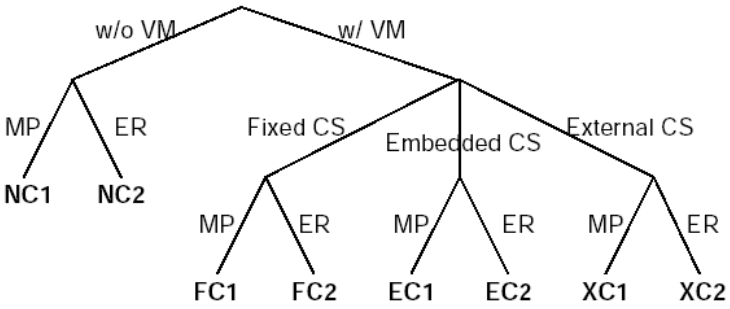
This document is a collection of thoughts that I have been developing and maintaining for several years on the notion of a multi-layered, open DRM standards architecture, which I think of as *OpenDRM*. Some aspects of this argument have been articulated in earlier works such as [PREP] and [FairUse]. These comments are not meant to reflect any specific project by my employer or of any standards group, but simply my current, personal view of the world.

DRM Overview and Reference Model

It is useful to begin with a brief, general, technical overview of DRM technology. We can set the context for the discussion by placing OpenDRM within a DRM *taxonomy,* and then by presenting a walk-through of a generic *DRM Reference Model* that applies to OpenDRM.

A DRM Taxonomy

In [Park], Park, Sandhu and Schifalacqua outline a taxonomy for the "controlled dissemination of information," which can be usefully applied to DRM architectures. As can be seen from tree diagram below, their categorization ranges from no control being applied, to very flexible mechanisms for maintaining originator control over specific uses of resources.



VM: Virtual Machine
MP: Message Push
ER: External Repository
CS: Control Set

NC1: No control architecture w/ MP
NC2: No control architecture w/ ER
FC1: Fixed control architecture w/ MP
FC2: Fixed control architecture w/ ER
EC1: Embedded control architecture w/ MP
EC2: Embedded control architecture w/ ER
XC1: External control architecture w/ MP
XC2: External control architecture w/ ER

That this taxonomy can be applied to DRM mechanisms in general is seen by considering the following mappings:

- A *virtual machine* is the system element that implements the desired control. Central to this would be the *DRM Client* or *Controller,* but the reach of this control must also typically extend into the user application (e.g. a rendering or editing tool).
- A *control set* specifies the usage rules that must be enforced. As shown in the diagram, in some implementations this control set may be either *fixed* or built-in to the virtual machine, or are *embedded* or otherwise attached to the resource when deployed. In both cases neither the originator nor the recipient can change the rules once either the controller or the resource are deployed. The most interesting is the third case, in which the control set is managed *externally* (separately, in terms of time and space) from both the DRM controller and the deployed content, best characterizes most modern DRM technologies. Both embedded and external control sets are typically expressed using some form of ***rights expression language***.

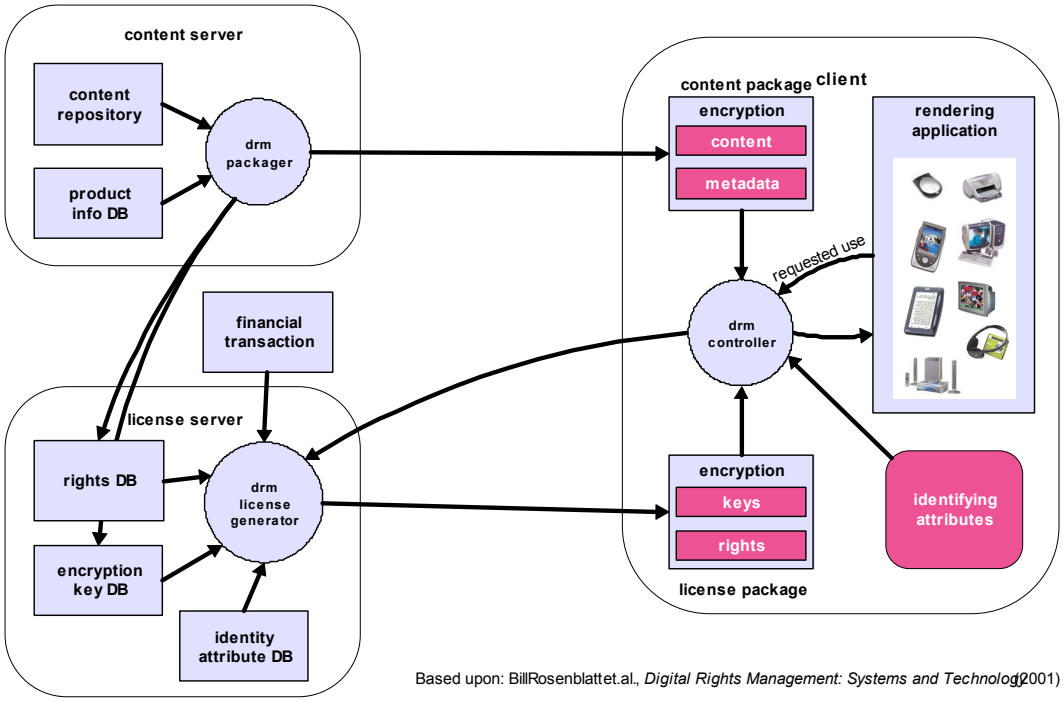
  Although not specifically addressed by the authors, some combination of the embedded and external models are possible, for example when certain "default" or generic privileges have been attached to the deployed resource but the recipient wishes to "upgrade". The external control set can be made to either supplement or revoke the embedded rules [DAGS95].

- Mechanisms can vary in how they *individualize* the deployed content to the recipient, based upon whether the content is retrieved from an *external repository* or through simple file transfer (e.g. peer-2-peer), which the authors term *message push.*

The *DRM Reference Model* which follows primarily applies to the *external control set* sub-branches of this taxonomy.

The DRM Reference Model
The diagram that follows provides an overview of a generalized, comprehensive DRM system in which users are granted specific use rights to information based upon originator control. Since this

model accurately describes most commercially-viable, *second-generation* DRM solutions, we will refer to it as our *DRM Reference Model*. Note that this model assumes the availability of standardized or proprietary "infrastructures" for identification, metadata, authentication and cryptography.



Based upon: BillRosenblattet.al., *Digital Rights Management: Systems and Technology* (2001)

The following outlines the process flow depicted in the DRM Reference Model:

1. *User obtains content:* The user might receive it through file-transfer or streaming protocols, by way of a direct request to a file server or through p2p file sharing, email, or direct media transfer (i.e. on removable media).
2. *User attempts to use the content in some way:* The DRM client determines, through policies bound to the package and/or implicit in the packaging format, that the requested use requires *authorization*.
3. *DRM Client makes Rights Request:* If the *license package* containing the necessary authorization credentials cannot be found on the user's machine or has expired, attributes of the user's request, including the *usage context*, are packaged and sent to a license server.
4. The license server verifies the submitted client identification or attributes credentials against an identity or attribute database.
5. The license server looks up rights specifications (rules) for this content item.
6. A financial transaction is launched, if none has been recorded and the rules require it.
7. The contents of the license package are assembled: the rights specification, various identifiers or attributes, revocation information, cryptographic keys to the content, all specific to the content and context of use.
8. The license is securely packaged (including authentication information) and transferred to client.
9. The DRM client uses the license to open the content for the particular requested use.
10. The content is rendered or otherwise, as requested.

In the reference model above we assume that the interactions between the *DRM Client* and the *DRM License Generator* are carried out using a ***rights messaging*** (or *transaction*) ***protocol;*** the

"payload" of the messages that make up that protocol are composed using the vocabulary defined by the *rights expression language*. We therefore see that the ability to fully express both *rights requests* and *rights grants* (or permissions) must be included in the scope of any acceptable rights expression language. *The exchange of rights assertions and rights requests is symmetric in this architecture*.

Note that various specializations on this model are possible, but generally most robust, commercial systems will be found to be consistent with it. Also, the traditional, *first generation* approach to DRM can be characterized by “throwing” cleared content “over the wall” to a rendering application following decryption by the DRM client. This is universally regarded as weak security, and is therefore the motivation for *next-generation DRM approaches* that are based upon authenticated code and trusted execution and which provider kernel-level support for handling unencrypted content.

Architectural Principles

At its core, OpenDRM should provide a framework that defines *open interfaces* between at least three architectural levels of abstraction: *rights expression languages, rights messaging protocols* and *mechanisms for policy enforcement and compliance*.

Rights Expression Languages

As we saw in the DRM taxonomy discussion earlier, so-called *rights expression languages* provide the basis for expressing rights information and usage control policies, and as such can supply the payload vocabularies for a variety of rights messaging applications including:

- Intellectual property rights (IPR) information discovery
- Simple policy expression, including constraints on access to resources
- Rights negotiation and trading, including rights requests and/or claims by information users
- The expression of rights agreements and electronic contracts (e-Contracts)

Minimally, a rights language should provide *vocabulary and syntax* for the declarative expression of rights and rights restrictions. In order to guarantee interoperability and evolvability, we would expect a rights language to be inherently extensible: it must provide an open-ended way to express rights not anticipated by the language "core." Such extensions might accommodate new *operations on content*, including uses that are specific to particular media domains, or new *contextual constraints*.

We believe that rights languages will prove to be important enablers of interoperability for systems that mediate access to resources. Although the principles underlying IPR-specific metadata have been around since at least 1993 [Perritt], the importance of *standardized* vocabularies and formats for expressing IPR policies has only recently begun to be appreciated at the application level [XRML], [ODRL]. More recently, at least one alternative rights metadata schema has been developed for articulating a rightsholder's desire to turn over the public domain specific rights to works [Commons].

> **Note:** *At this writing the OASIS Rights Language TC is using XrML 2.1 as a starting point for its standardization work. They have adopted an "extension" model similar to that described above. XrML has also been adopted as the rights expression language of choice for the MPEG-21 standard, while the Open Mobile Alliance* [OMA] *has adopted the Open Digital Rights Language* [ODRL]*, a rights language standard proposed by Renato Ianella. One analysis of XrML and ODRL appears at* [XrMLODRL].

It should be assumed that IPR policies must be expressed at different levels of abstraction, and therefore different vocabularies will be appropriate. To avoid chaos and to facilitate interoperability, we believe that some sort of *rights language ontology* will be required: a set of reusable terms for the basic rights concepts that can be mapped to different syntaxes. This is precisely the work that was begun by the <indecs> project in 1998 [indecs], and is a fundamental basis of ODRL and, more completely, the <indecs>2-RDD *Rights Data Dictionary* project. [indecs2]

> **Note:** *The <indecs>2-RDD has also been adopted as the rights data dictionary model for the MPEG-21 standard.*

This concept of *interoperability through shared ontology* is similar to what the ebXML working group has been trying to achieve with their *core components* approach to building interoperable business objects. [ebXML] Following that model, existing or future IPR expression languages could interoperate through translation via this shared semantic layer, rather than necessarily forcing applications and services to use a single common language.

Finally, we caution that rights languages should not be thought of as e-Contract languages; rights vouchers are not e-Contracts! It is clear that one possible use of a rights language is to express the parties, terms, rights and obligations typically enumerated in e-Contracts. [c.f. FIRM] However, e-Contracts go beyond the immediate scope of OpenDRM since they require complex, dynamic behavior, not just declarative expression of state.

> **Note:** *The* rights specification *created by a rights expression language and contained within in a license package can be thought of as a* ***manifestation*** *of an e-Contract, held and maintained by some* license server *and generated based upon a specific context of use. Such e-Contracts are reifications of usage licenses.*

Rights Messaging Protocol
OpenDRM should define a *rights messaging or transaction protocol* (RMP) that would provide the means for inquiring about and disseminating IPR information and policies; it may also include a means for determining an agent's capabilities for enforcement or compliance with particular IPR policies. The compliance aspect of an RMP would permit intermediaries or agents to describe any IPR tracking mechanisms they may use, and the types and strengths of enforcement mechanisms they are able or to offer. [c.f. CC/PP] Some elements of this IPR information, supplied independently of the primary content stream, would allow consumers (or agents operating on their behalf) to decide whether to enter an agreement; other elements would enable publishers or intermediaries to determine whether to pass along information items on the basis of that information.

[RightsTalk] proposed an *Open Rights Management Interoperability Framework*, which was suggested as a way to exchange rights messages between peer applications, primarily by way of web-published services. The RightsTalk concept was not meant to target a particular sector; one could see it working as well for business-to-business messaging (e.g. sub-rights transactions) as it could for business-to-consumer (e.g. IPR policy expression) or consumer-to-business (e.g. IPR policy discovery, authorization queries). The important aspect of the RightsTalk concept was that interoperability between services and applications could be established by defining an extensible set of standardized rights messages and a standard way for handling those messages, including their sequencing and routing to applications.

It is clear that the ongoing standards efforts directed at delivering XML-based web services can provide the basis for a standardized rights messaging protocol. Mechanisms ranging from XMP-RPC and SOAP to the OASIS SAML and XACML efforts can provide a strong, standardized foundation upon which to build an ecosystem of interoperating rights management services that achieve a variety of goals.

> **Note:** *A standardize rights messaging protocol will be a critical element in achieving advanced rights management applications involving third parties, such as the* fair use infrastructure *described by Burk and Cohen* [Cohen] *or federated rights management systems that accommodate institutional users* [Martin]. *The case for this is made to a much greater extent in* [SLTPPC].

Mechanisms for Policy Enforcement (DRM)
OpenDRM should not specify a particular IPR policy enforcement mechanism, but rather should provide a set of *open APIs* that will enable a competitive market in the provision of enforcement methods. OpenDRM should provide a way to register sets of bindings that tie the leaves (or certain expressive elements) of rights expression languages into commercial transactions or other enforcement or compliance mechanisms. This ability to associate expressive vocabulary with

technical mechanisms provides operational semantics to the declarative platform for rights expression.

OpenDRM's policy-enforcement architecture must eventually accommodate the strict enforcement of policies in situations other than end use; examples are prevalent within the domain of b2b rights trading, where a publisher-to-distributor deal might stipulate that subsequent deals, involving rights assigned by the primary deal, are not authorized without (for example) a particular crypto-based token. Policy-compliant *deal engines* will understand this policy expression and will enforce the policy appropriately. Affordances in the OpenDRM architecture for policy-enforcement do not need to specify the nature of such tokens, only accommodate them.

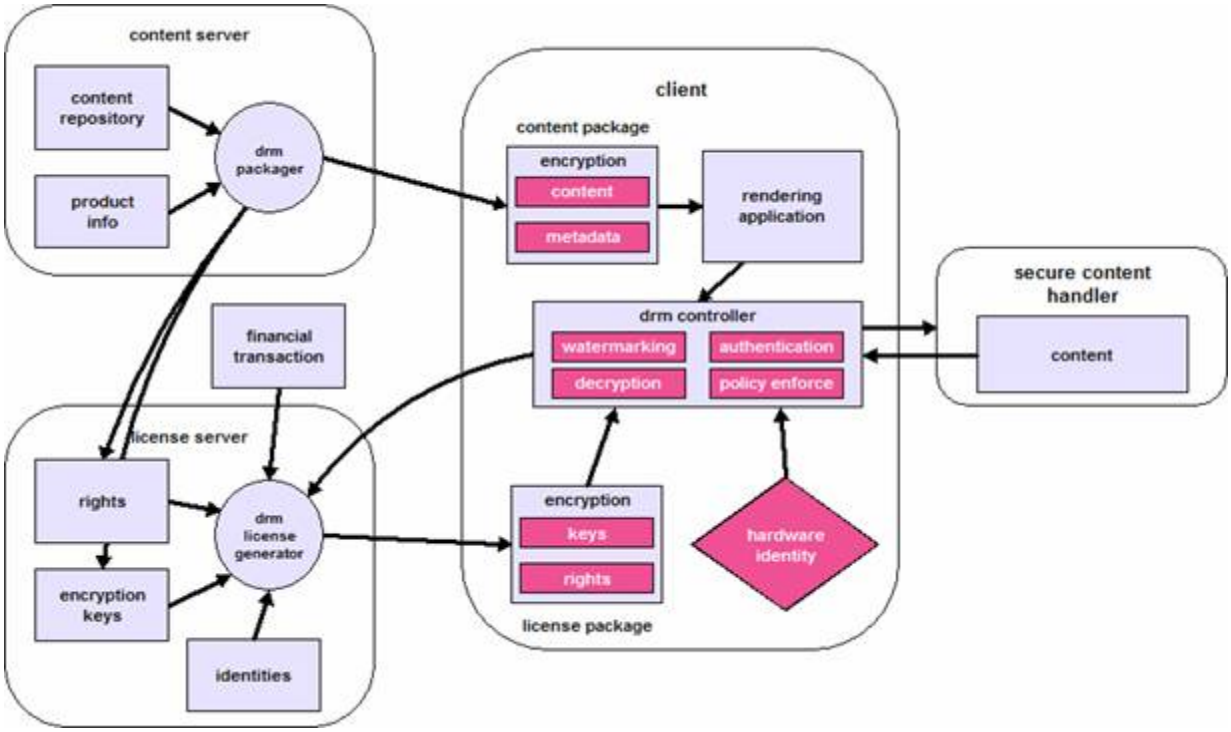
It is not clear what (if any) existing standards forum has in scope the establishment of technical enforcement API standards. One possible home might be the IPMP activity within MPEG-21. A possible approach to the problem might be a generalized secure container that would facilitate enforcement at the client end. This could be a common, open content presentation format, providing in essence a DRM-enabled meta-container that would enable vendors to co-exist with differentiated solutions, but yet would provide application developers and users with a uniform, extensible interface.

Such a meta-container or policy enforcement envelope would unambiguously guide a standard envelope handler to call the appropriate DRM mechanism. From the envelope processor's perspective, all DRM mechanisms would be equally accessible. For DRM vendors, this would seem to be ideal: each would be "inside" any application that would adopt the format.

DRM and Trusted Platforms

An important role of the so-called *trusted* or *authenticated execution environment* is to provide *containment* for code execution. In particular, trusted execution prevents other applications from accessing the particular executable's memory space, except through a well-defined, controlled interface [England]. A trusted environment may also provide for *authenticated loading* of executable modules, validating code against signatures and checking signed modules against policies for acceptance or revocation. This process of *code certification and authentication* ensures that loaded modules provide a particular, assured behavior, and prevents the loading and execution of rogue code.

These trusted execution environments increase the reach of DRM solutions. They dramatically increase the assurance that the various code modules responsible for handling and "unlocking" the content, including DRM clients, system constructs such as local content repositories, and downstream rendering applications, will each *do the right thing* and act according to expressed policies. Important examples of DRM based upon these concepts are the Microsoft "DRM-OS" and the Intertrust "RM-OS"; some of these concepts have been deployed in the *Secure Audio Path* in Windows Me and XP.



The fundamental improvement of DRM based on trusted execution over previous models is that in the trusted model, unencrypted content does not simply get tossed “over the wall” to the rendering application. Rather, the content handler maintains control of the content, and applications must request content services through the controlled interface. The content handler is therefore the intermediary responsible for enforcing the use policies. This is illustrated in the modified *DRM Reference Model* diagram which appears above.

A special category of trusted platform is the *dedicated system* built on closed hardware and software. Examples include certain eBooks, music players and other dedicated rendering appliances. Closed, proprietary systems represent the ultimate manifestation of the controlled boot/execution model.

Another source for information on *kernel-level enforcement* is [Elisar]

Appendix: Further Thoughts on Next-Generation DRM

> **Note:** *This discussion was written several months before Microsoft's Palladium architecture went public, and anticipates a generalized approach to the same end:* ***trusted execution****.*

The purpose of this section is to help ensure that OpenDRM is in a position to contribute to the development of industry-leading DRM technologies, solutions and standards in the future. Please note that these thoughts are not meant to be exceptionally new or radical. Their motivation is merely to put things in order and to organize our thinking in this specific area. Some of this thinking — perhaps most of it —mirrors “trusted infrastructure” solutions that Microsoft, Intertrust and other major DRM players have envisioned, in some cases for almost a decade. It is our hope that by factoring this thinking into the *OpenDRM Architectural Framework*, this effort will contribute to the industry by fostering a more open dialog while moving the technologies forward.

- Uppermost on my mind is the emerging role of *trusted systems*. Next-generation trusted platforms will have certified, named configurations, which will typically be characterized by the equivalent of a cryptographically signed registry. External systems wishing to access and interact with remote targets will ask them to produce certificates that can be used to demonstrate the authenticity of their configurations.

- A trusted system must undergo a process of *authenticated boot*, whereby only authenticated components that are part of the certified profile are loaded by an authenticating boot loader. Such components will have been previously tested and signed; any component that is a candidate for loading will be required to match the signature that has been stored within the profile prior to loading. The profile therefore acts like a "signature" for the configuration.
- Within trusted systems, components will check the authenticity of other components in a similar fashion. Although the procedures for signature-based code authentication might be carried out by many systems, the *decision* of whether to accept a specific certification is an individual one that can only be based upon the needs and wishes of the particular systems integrator or administrator. In the future, components will not interoperate with components that they do not trust. The key is to realize that "trust" is inherently *relative*.
- The expansion and contraction of a subsystems' umbrella of trust is likely to be highly dynamic, with *revocation* and *component exclusion* being integral parts of interactions between components in the infrastructure. An important design concept is therefore the ability of the infrastructure to "heal" itself, always working to mitigate the damage that may potentially be caused by the global propagation of rogue components.
- Trusted execution environments like those described above will provide the sort of containment necessary to realize *kernel-level content protection mechanisms* that actually have some hope of being robustly secure. Examples of systems that can leverage trusted component infrastructures include policy enforcement "engines" that act based upon declarative policy expressions, and secure media pathways that only propagate cleartext media streams to trusted media-handling modules. An early example of this is the Microsoft Secure Audio Path (SAP), first deployed on Windows Me and Windows XP.

I see a variety of challenges that will hinder the adoption of this sort of trusted component infrastructure:

- *The scalability of certification facilities and their procedures*. So far the only code-signing facility for trusted Windows components (esp. Windows DRM components, Secure Audio Path-certified components, etc) is the Windows Hardware Quality Lab (WHQL). This is problematic not only because of Microsoft's proprietary control over WHQL, but raises practical questions over whether WHQL will have the necessary "bandwidth" to support reliable testing for all candidate components.
- The same issues are valid for at least one other example of proprietary code testing and certification: all components developed using Intertrust's SDK's are subjected to a similar certification process.
- The nature of the actual code reviews and tests performed by proprietary code certification labs poses a different sort of danger (or concern). To be of value, code certification must entail more than the simple "rubber-stamping" of code; one should expect a certifying organization to run through a battery of code and interface (or specification) reviews to reduce the vulnerability of trusting applications to exploits of a component. So it is clear that *any* signing organization, proprietary or otherwise, must *explicitly declare* what tests it performs — in other words, *exactly what its certification means*.

  Hardware and software component certification might well emerge as a type of value-added service, but it will be an absolute requirement for providers of end-to-end trusted solutions. For example, a trusted printer driver might in turns be signed by *Vendor M* and *Vendor H*; for some applications the M-level certification might be sufficient, but for a particular *Vendor H* trusted printing application the additional level of signing might be required.

- PGP-inspired, peer-level code signing models are conceivable, but are they *practical*? The "peers" in this case would typically not be individual users, but rather partnering and *inter-working* organizations. All of the arguments of the previous point hold, but with special emphasis placed on the *certification criteria*, the nature of tests and the competency and bandwidth of the testers — in other words, what the certifications mean in the context of the business relationship.

- *How realistic is the notion of an interoperable trust model?* Consider this scenario: prior to interacting with some remote service, let's assume that a particular service wants to authenticate a target platform — in the future world of trusted infrastructures, it will not be sufficient for a service simply to provide a compatible interface; the implementation, including its execution platform, will be required to demonstrate its authenticity.

  Our expectation is that the target will be required to produce some signature that characterizes its current configuration, which will then be checked against some signature of record for platforms of that type, certified at a particular level of trust. T/Linux, T/Windows, T/PPC, T/SonicBlue, T/iDEC, T/eBook will have inherently *different mechanisms* for maintaining their configurations — different manifestations of a "signed registry"— and will therefore respond to challenges differently; yet the larger certification/authentication principles must be the same. How will this ecosystem work?

  Most promising for the future may be a *common, open, secure content object storage model and interface*, something like an abstracted Kahn/Wilensky *repository access protocol* [RAP]. This could be thought of as a generalization on Microsoft's UnifiedDRM model, which provides a uniform content access model that centrally controls access to and handling of content and may be secured by "individualizing" the interface to a specific client platform [UnifiedDRM]. The apparently intent is for this secured content-handling architecture to be broadly applied, so that it will work for any application and any content type, not simply those controlled by Microsoft.

  In order for a *generalized* secure digital object model to be universally adopted it must be *transportable to a wide selection of platforms*; the data model of deployed objects must be the same, regardless of whether the target platform is T/Windows or T/Linux. The controlled, programmatic "container" interface through which applications request access to the content must be presented in a way that is accessible to a wide variety of applications on that platform. Finally, each of the other platform-dependant aspects of the trust infrastructure must be replicated — including the certification of the target platform's code, especially those applications that will request to access and process cleartext versions of the secured content.

It is vitally important for the IT industry to commit itself to delivering open solutions to the content value chain that provide content owners with *trusted, end-to-end infrastructures*. Achieving these goals with today's methods, platforms and infrastructure seems impossible; even those few companies that have developed trusted solutions are a long way from the sort of ubiquity, transparency and ease-of-use that content owners want. The OpenDRM contribution will be to identify, understand and eliminate the barriers that have prevented previous industry players from being successful, either because they have been blind to the keys to unlocking the opportunity — such as the role of open standards in fostering innovation — or have simply lacked the scale and resources to break through the barriers.

References and Further Reading

[PREP] John S. Erickson et.al, Principles for Standardization and Interoperability in Web-based Digital Rights Management: A Position Paper for the W3C Workshop on Digital Rights Management (January 2001). See http://www.w3.org/2000/12/drm-ws/pp/hp-erickson.html

[Commons] Creative Commons, "Creative Commons: Cultivating the Public Domain." White Paper (2002). See http://www.creativecommons.org/concepts/cultivating.

[Park] Jaehong Park, Ravi Sandhu, and James Schifalacqua, "Security Architecture for Controlled Digital information Dissemination." The Proc. of Annual Computer Security Applications Conference (ACSAC), New Orleans, Louisiana, Dec. 2000. See http://dlib.computer.org/conferen/acsac/0859/pdf/08590224.pdf

[DAGS95] John S. Erickson. "A Copyright Management System for Networked Interactive Multimedia," Proceedings of the Dartmouth Institute for Advanced Graduate Studies: Electronic Publishing and the Information Superhighway, Boston, MA (June 1995).

Also John S. Erickson, "US Patent #5765152: System and method for managing copyrighted electronic media" (July 8, 1998). See http://www.delphion.com/details?&pn=US05765152__

[England] Paul England and Marcus Peinado, "Authenticated Operation of Open Computing Devices," ACISP 2002, published in LNCS 2384 (Springer-Verlag), pp. 346-361 (July 2002).

[CC/PP] Mikael Nilsson et.al, Composite Capabilities/Preference Profiles (CC/PP): Requirements and Architecture. See http://www.w3.org/TR/2000/WD-CCPP-ra-20000721/

[Cohen] Dan Burk & Julie Cohen, Fair Use Infrastructure for Copyright Management Systems, 11 Harv. J. Law & Tech. (forthcoming 2002)
See http://www.isc.umn.edu/research/papers/Tprc.pdf

[Martin] Mairéad Martin et.al., "Federated Digital Rights Management: A Proposed DRM Solution for Research and Education," D-Lib Magazine, Volume 8 Number 7/8 (July/August 2002). See http://www.dlib.org/dlib/july02/martin/07martin.html

[SLTPPC] Mulligan, D., Burstein, A., and Erickson, J. "Supporting Limits on Copyright Exclusivity in a Rights Expression Language Standard. A requirements submission to the OASIS Rights Language Technical Committee." On behalf of The Samuelson Law, Technology & Public Policy Clinic, and The Electronic Privacy Information Center (August 13, 2002).
See http://xml.coverpages.org/OASIS-SLTPPC-EPIC-8-13-02.pdf

[XrMLODRL] Bill Rosenblatt, "ODRL 1.1. Review," GiantSteps Media Technology Strategies (August 2002).
See http://www.giantstepsmts.com/DRM%20Watch/odrl11.htm

[Digest] H. Franks et.al, HTTP Authentication: Basic and Digest Access Authentication, IETF Internet-Draft (September 1998). See http://hopf.math.northwestern.edu/digestauth/draft.rfc

[DublinCore] The Dublin Core Metadata Initiative. See
http://www.oclc.org/oclc/research/projects/core/index.htm

[EBX] The EBX Specification. See http://www.ebxwg.org/pdfs/spec.pdf

[ebXML] ebXML technical Architecture Specification, v0.9 (October 2000). See
http://www.ebxml.org/specdrafts/ebXML_TA_v0.9.pdf

[Elisar] Elisar Software Corporation, An Overview of Digital Rights Enforcement and MediaRights Technology (April 5, 2001). See http://www.elisar.com/news/MRoverview.pdf

[UnifiedDRM] John Manferdelli, "New Challenges in Embedded Security: Digital Rights Management," Consortium for Efficient Embedded Security (CEES), Boston (July 2001). See
http://www.ceesstandards.org/downloads/Microsoft_John_M.ppt

[RAP] Sandra Payette, Christophe Blanchi and Naomi Dushay, "Repository Access Protocol (RAP) IDL Version 1.3,"
See http://www.cs.cornell.edu/cdlrg/fedora/IDL/

[FIRM] R. Martin Roscheisen, A Network-Centric Design For Relationship-Based Rights Management, Ph.D. Dissertation (Stanford University, 1997).
http://pcd.stanford.edu/rmr/commpacts.html

[RightsTalk] John S. Erickson, Toward an Open Rights Management Interoperability Framework (June 1999). See http://www.oasis-open.org/cover/ericksonRT19990624.pdf

[KhareReagle] Rohit Khare & Joseph Reagle, Panel 3: Rights Management, Copy Detection, and Access Control, NRC/CSTB/Information Systems Trustworthiness Project. See
http://www.w3.org/IPR/work/NRC-v1.htm

[indecs] Godfrey Rust and Mark Bide, The <indecs> Metadata Framework (June 2000). See
http://www.indecs.org/pdf/framework.pdf

[ODRL] Renato Iannella, Open Digital Rights Language Specification v1.0. See
http://odrl.net/1.0/ODRL-10.pdf

[OeB] Open eBook Publication Structure. See http://www.openebook.org/specification.htm

[P3P] Platform for Privacy Preferences (P3P) Project. See http://www.w3.org/P3P/

[P3PDataXfer] P3P Working Group, Removing Data Transfer from P3P. See http://www.w3.mag.keio.ac.jp/P3P/data-transfer.html

[P3P1.0] The Platform for Privacy Preferences 1.0 Specification. See http://www.w3.org/TR/P3P/

[Perritt] Henry W. Perritt, Jr. Knowbots, Permissions Headers and Contract Law, Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. See http://www.ifla.org/documents/infopol/copyright/perh2.txt

[TCPA] Trusted Computing Platform Alliance (TCPA) Specification v0.9. See http://www.trustedpc.org/home/Specification.htm

[W3C7Points] W3C, The W3C in 7 Points.

[XRML] ContentGuard, Inc., XrML White Paper. See http://www.xrml.org/white_paper.htm