UNIVERSITY
OF WOLLONGONG
AUSTRALIA

University of Wollongong
Research Online

Faculty of Informatics - Papers (Archive)

Faculty of Engineering and Information Sciences

2007

# On implementing MPEG-21 intellectual property management and protection

N. P. Sheppard
*University of Wollongong*, nps@uow.edu.au

## Publication Details

This conference paper was originally published as Shephard, NP, On implementing MPEG-21 intellectual property management and protection, in Proceedings of the ACM Workshop on Digital Rights Management 2007, 29 October 2007, Alexandria, Virginia, USA, 10-22.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

# On implementing MPEG-21 intellectual property management and protection

**Abstract**
The MPEG-21 Intellectual Property Management and Protection (&lquot;IPMP&rquot;) Components specify a framework for inter-operable and renewable digital rights management based on IPMP tools that implement proprietary digital rights management features. MPEG-21 defines the mechanism by which protected multimedia objects are associated with proprietary IPMP tools, but does not specify the interface through which IPMP tools and MPEG-21 terminals communicate. This paper describes an implementation of the IPMP components including an interface to IPMP tools based on the MPEG Rights Expression Language; dynamic construction of authorisation proofs that permit a principal to carry out an action; and a cryptographic architecture bound to the existence of authorisation proofs. This implementation has been applied to scenarios in copyright protection, privacy protection and corporate document protection, suggesting that &lquot;IPMP&rquot; may be useful in applications other than intellectual property.

**Keywords**
digital rights management, MPEG-21 IPMP, MPEG REL, implementation

**Disciplines**
Physical Sciences and Mathematics

**Publication Details**
This conference paper was originally published as Shephard, NP, On implementing MPEG-21 intellectual property management and protection, in Proceedings of the ACM Workshop on Digital Rights Management 2007, 29 October 2007, Alexandria, Virginia, USA, 10-22.

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/580

# On Implementing MPEG-21 Intellectual Property Management and Protection*

Nicholas Paul Sheppard
School of Computer Science and Software Engineering
The University of Wollongong, NSW, 2522
Australia
nps@uow.edu.au

## Abstract

The MPEG-21 Intellectual Property Management and Protection ("IPMP") Components specify a framework for inter-operable and renewable digital rights management based on *IPMP tools* that implement proprietary digital rights management features. MPEG-21 defines the mechanism by which protected multimedia objects are associated with proprietary IPMP tools, but does not specify the interface through which IPMP tools and MPEG-21 terminals communicate.

This paper describes an implementation of the IPMP components including an interface to IPMP tools based on the MPEG Rights Expression Language; dynamic construction of authorisation proofs that permit a principal to carry out an action; and a cryptographic architecture bound to the existence of authorisation proofs. This implementation has been applied to scenarios in copyright protection, privacy protection and corporate document protection, suggesting that "IPMP" may be useful in applications other than intellectual property.

## 1 Introduction

The increasing availability of network technologies has made electronic distribution an attractive mode of distribution for valuable multimedia works, private data and sensitive corporate information. The owners of valuable and important information, however, rarely want this information to be distributed and used without limit.

Digital rights management allows information owners to control and monitor the distribution of files through electronic channels. Digital rights management technology has become well-known for its role in copyright protection for Internet music and video services, but is also becoming important in the protection of sensitive corporate information [1] and is emerging as a technology for protecting individuals' private information [22].

The MPEG-21 Multimedia Framework [14] seeks to define a generic framework for storing, distributing and using multimedia presentations. It is intended to provide standards for structuring collections of multimedia works, and managing and reporting on uses of these collections. We will give an overview of the MPEG-21 Multimedia Framework in Section 3.

Rights management is a core feature of MPEG-21. Part 4 of the specification proposes a framework for interoperable digital rights management centred on the notion of *IPMP tools* that are provided by the vendors of proprietary digital rights management systems, and Part 5 of the specification proposes a *rights expression language* to be used for writing licences that govern the use of multimedia works. Part 6 also defines a *rights data dictionary* for describing rights-managed scenarios, which is not addressed in the present paper.

In this paper, we describe our experiences in implementing a digital rights management system based on

*©ACM, 2007. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the ACM Workshop on Digital Rights Management 2007, http://doi.acm.org/10.1145/1314276.1314280.

the MPEG-21 IPMP Components. Our system, known as Smart Internet Technology Digital Rights Management ("SITDRM"), was developed at the Co-operative Research Centre for Smart Internet Technology in Australia, and has been applied to applications in copyright protection, privacy protection and enterprise DRM.

MPEG-21 leaves the form and scope of an IPMP tool largely undefined, stating simply that an IPMP tool "can be a single protection module, for example, a single decryption tool, and can also be a collection of tools, i.e. a complete IPMP system." Section 4 of the present paper describes the IPMP engine implemented by our digital rights management system, including an interface for IPMP tools based on the structure of the MPEG Rights Expression Language.

Section 5 describes our approach to distributing and locating licences and cryptographic keys required to access resources that have been protected using IPMP tools. In common with other digital rights management systems, our approach binds the possession of cryptographic keys to the possession of an *authorisation proof* (set of licences) that permits an action that requires those keys. In our system, terminals are able to build an authorisation proof dynamically at the time an authorisation request is made rather than require an information owner to supply a fixed authorisation proof at the time protected resources are created.

Section 6 outlines the applications to which we have applied SITDRM, and describes the particular security architecture and IPMP tools that we used in our applications.

Finally, we discuss some of the observations we made in implementing the IPMP Components in Section 7. While some aspects of the MPEG-21 specification seem problematic, we were able to construct a digital rights management framework that could be applied to a variety of scenarios beyond the intellectual property scenarios contemplated by MPEG. Our system could be improved, however, by more efficient construction of authorisation proofs and an infrastructure for distributing IPMP tools.

# 2 Related Work

Many digital rights management systems have been developed and numerous proprietary systems are currently on the market. The lack of inter-operability between digital rights management schemes, however, is a well-known barrier to popular acceptance of digital rights management.

Several specification bodies have attempted to address inter-operability by proposing specifications for either self-contained digital rights management regimes [27, 26], or frameworks for exchanging rights-managed information between different regimes [6]. MPEG-21's IPMP Components [16] – and the related specification released by the Digital Media Project [9] – take a "configuration-driven" approach [23] in which critical rights management functionality is left to the developers of proprietary plug-ins called "IPMP tools". IPMP tools have an additional advantage in that they provide for renewability of digital rights management components that have been compromised or othwerwise made obsolete. This paper focuses on the latter approach.

## 2.1 MPEG-21

Though reference implementations of other parts of the MPEG-21 Framework are available [18], no reference implementation is currently available for the IPMP Components and in this paper we describe an implementation that has been made independently of MPEG.

Brox [3] and Karpouzis, et al. [21] note the value of including the digital rights management features of MPEG-21 into platforms for health records and sports broadcasting, respectively, but neither describe an actual implementation of the IPMP Components used in their systems.

Suen [34] has implemented an MPEG REL evaluator based on an expert system. Suen's parser views a set of MPEG REL licences as a set of predicates, and a request to perform an action as a hypothesis. If the expert system determines that the hypothesis is true given the set of predicates, the action is permitted. We will consider the relationship between Suen's evaluator and ours in Section 5. Other MPEG REL evaluators are available [18, 5] but we do not know how they are designed.

## 2.2 MPEG-4

Like MPEG-21, MPEG-4 employs the notion of IPMP tools in order to provide inter-operability and renewability

for digital rights management systems. Also like MPEG-21, the original version of MPEG-4 IPMP did not specify the interface to IPMP tools and IPMP tool implementations, e.g. [13], depended on the interface defined by a particular MPEG-4 player.

MPEG recognised that this approach had drawbacks in that tools had to be re-written for every player, and proposed the IPMP Extensions to remedy this [20]. The IPMP Extensions define a message-passing interface between terminals and tools so that any tool can communicate with any terminal. Implementations of the IPMP Extensions are described by Fan, et al. [11] and Serrão, et al. [31].

While a message-passing interface has a number of advantages, we chose to use a functional interface for our tools as it is considerably easier to implement. Our interface could be adapted to a message-passing one but this is left as future work.

### 2.3 The Digital Media Project

The Digital Media Project was established in 2003 with a vision to promote widespread use of digital media and, in particular, an inter-operable digital rights management regime. At the time of writing, the Digital Media Project had just released Version 2.1 of its specification for an inter-operable digital rights management platform [7]. This specification adopts MPEG-21's Digital Item Declaration Language and IPMP Components, together with MPEG-4's IPMP Extensions for defining an interface for IPMP tools (called "DRM tools" by the Digital Media Project).

The Digital Media Project is working on reference software that implements its specification, known as "Chillout" [8]. No formal release had been made at the time of writing, though some source code was available from the Digital Media Project's version control system, and some incomplete documentation became available at around the same time this paper was written.

There are necessarily a number of similarities between Chillout and SITDRM since they are implementing very similar specifications. SITDRM's approach to IPMP tools, however, is quite different to that used in the Digital Media Project in that IPMP tools in SITDRM are bound to the intepretation and enforcement of licences, while DRM tools in Chillout are focused on the protection of multimedia files.

### 2.4 Other Systems

The conditional access community has proposed configuration-driven systems apart from those espoused by MPEG, such as the Open Platform Initiative for Multimedia Access ("OPIMA") [28, 31]. Compared to the Digital Media Project, OPIMA's approach to IPMP tools is at the opposite end of MPEG-21's spectrum of IPMP tools in that OPIMA's "IPMP Systems" are complete digital rights management or conditional access systems in their own right. SITDRM, on the other hand, views tools as a means of implementing a particular system based on the MPEG Rights Expression Language.

On a theoretical level, SITDRM's approach bears some resemblance to aspects of the "layered" approach to digital rights management proposed by Jamkhedkar and Heileman [19, 12]. Jamkhedkar and Heileman have the REL interpreter situated in a "rights expression and interpretation layer" that calls upon a "digital rights enforcement upper layer" to perform cryptographic operations. SITDRM provides a concrete example of this kind of architecture in which the digital rights enforcement upper layer is populated by IPMP tools, though some functions of SITDRM's tools might also fall into the rights expression and interpretation layer in Jamkhedkar and Heileman's model.

## 3 MPEG-21

Unlike previous MPEG standards, MPEG-21 does not define the way in which individual multimedia presentations are encoded, but defines ways in which atomic multimedia objects can be combined, navigated and referenced. It consists of numerous parts ratified by the International Standards Organisation as the ISO/IEC 21000 series of standards.

This section gives a brief description of the components of MPEG-21 required to understand this paper. Sections 4 and 5 give further details about our interpretation of and extensions to the MPEG-21 Framework.

```
<didl:DIDL>
  <didl:Item>
    <didl:Component>
      <didl:Descriptor>
        <didl:Statement>
          <dii:Identifier>
            urn:org:doc:1
          </dii:Identifier>
        </didl:Statement>
      </didl:Descriptor>
      <didl:Resource>
         <myxml:MyXML>...</myxml:MyXML>
      </didl:Resource>
    </didl:Component>
  </didl:Item>
</didl:DIDL>
```

Figure 1: A simple digital item declaration.

### 3.1 Digital Items

The core notion in MPEG-21 is the notion of a *digital item* [15], which represents a collection of multimedia objects related in some way. Digital items are described using the XML-based *digital item declaration language* ("DIDL"), which organises content and meta-data into a hierarchical structure. The most important elements for understanding this paper are:

**Resources.** Atomic multimedia objects such as images, sounds and videos.

**Descriptors.** Elements containing meta-data about a resource, such as identifiers, abstracts, MPEG-7 descriptors and so on.

**Components.** Resources together with their descriptors.

**Items.** Complex multimedia objects, made up of sub-items and/or components.

Figure 1 shows a simple digital item declaration, consisting of a single item containing a single component. The resource is an XML document contained by the *MyXML* tags (the body of the document has been omitted for brevity), and is identified by the URN `urn:org:doc:1`.

### 3.2 Intellectual Property Management and Protection

*Intellectual property management and protection* ("IPMP") is MPEG's term for digital rights management [16]. MPEG-21 does not fix a particular digital rights management system, but assumes that IPMP functionality is provided by vendor-specific *IPMP tools* that can be downloaded and made accessible to the terminal as necessary. IPMP tools may implement basic functions such as decryption and watermarking, or may implement complete digital rights management systems in their own right.

We say an element is *governed* if it is protected by one or more IPMP tools. Governed portions of a digital item are expressed in the IPMP Digital Item Declaration Language ("IPMP DIDL"). IPMP DIDL has the same structure as DIDL, but associates each element with a plaintext identifier and an *IPMP information descriptor* that associates the resource with a licence and identifies the IPMP tools required to access the element, as shown in Figure 2. In this paper, elements of the IPMP DIDL namespace are distinguished from their equivalents in the DIDL namespace using the namespace prefix `ipmpdidl`.

MPEG-21 does little to define the scope and form of IPMP tools. In particular, MPEG-21 does not define a programming or communications interface between MPEG-21 terminals and IPMP tools, and it is unclear how IPMP tool implementors could construct tools to work with any MPEG-21 implementation.

For our implementation, we developed an interface based on a DIDL parser (itself based on the Document Object Model for XML documents [37]) and the MPEG Rights Expression Language. Our interface is described in Section 4 of the present paper, and the specifics of the IPMP tools that we implemented are described in Section 6.

### 3.3 Rights Expression Language

Though MPEG-21 does not define a full digital rights management system, it does define a rights expression language known as "MPEG REL" [17]. MPEG REL is closely based on the Extensible Rights Markup Language ("XrML") [4].

An MPEG REL licence is structured as a collection of

```
<didl:DIDL>
  <didl:Item>
    <didl:Component>
      <ipmpdidl:Resource>

        <ipmpdidl:Identifier>
          <dii:Identifier>
            urn:org:doc:1
          </dii:Identifier>
        </ipmpdidl:Identifier>

        <ipmpdidl:Info>
          <ipmpinfo:IPMPInfoDescriptor>
            ...
          </ipmpinfo:IPMPInfoDescriptor>
        </ipmpdidl:Info>

        <ipmpdidl:Contents>
          <xenc:EncryptedData>
            ...
          </xenc:EncryptedData>
        </ipmpdidl:Contents>

      </ipmpdidl:Resource>
    </didl:Component>
  </didl:Item>
</didl:DIDL>
```

Figure 2: A governed version of the digital item shown in Figure 1.

*grants* issued by some licence issuer. Each grant awards some *right* over some specified *resource* to a specified *principal*, that is, user of a resource. Each grant may be subject to a *condition*, such that the right contained in the grant may not be exercised unless the condition is satisfied.

Each of the four components of an MPEG REL grant is associated with an abstract XML schema type. In any actual licence, each of these abstract types must be instantiated by a concrete type representing a particular principal, right, resource or condition.

In order to perform some action on a resource, a user (principal) must possess a licence containing a grant that awards the right to perform that action on that resource, and satisfy the associated condition. The terminal must check this before exercising the right.

MPEG REL is defined as a collection of three XML schemata, called the *core schema* (denoted by the XML namespace prefix `r` in this paper), the *standard extension schema* (prefix `sx`) and the *multimedia extension schema* (prefix `mx`). These schemata define the fundamental elements of the language, some widely-useful conditions, and elements useful in copyright protection applications, respectively.

Figure 3 shows an example of an MPEG REL grant allowing a principal (`r:keyHolder`) identified by his or her public key to print a resource (`mx:diReference`) identified by a digital item identifier `urn:org:doc:1`. The principal is only permitted to print the resource once (`sx:ExerciseLimit`).

A number of the principals and conditions defined by MPEG REL, such as the `PropertyPossessor` principal and `PrerequisiteRight` condition, depend on the existence of other grants to be authorised. Thus a series of grants known as an *authorisation proof* may be required to authorise a single action. Authorisation proofs play an important role in the SITDRM system and will be discussed further in Section 5.

# 4 An MPEG-21 IPMP Engine

ISO/IEC 21000-4 leaves the scope and form of IPMP tools almost entirely undefined. We suppose that IPMP tools are intended to perform the functions performed by the "navigation" and "resource access" tools in this paper,

```
<r:grant>
  <r:keyHolder>
    <r:info>
      <dsig:KeyValue>
        <dsig:RSAKeyValue>
          ...
        </dsig:RSAKeyValue>
      </dsig:KeyValue>
    </r:info>
  </r:keyHolder>
  <mx:print/>
  <mx:diReference>
    <mx:identifier>
      urn:org:doc:1
    </mx:identifier>
  </mx:diReference>
  <sx:ExerciseLimit>
    <sx:count>1</sx:count>
  </sx:ExerciseLimit>
<r:grant>
```

Figure 3: An MPEG REL grant.

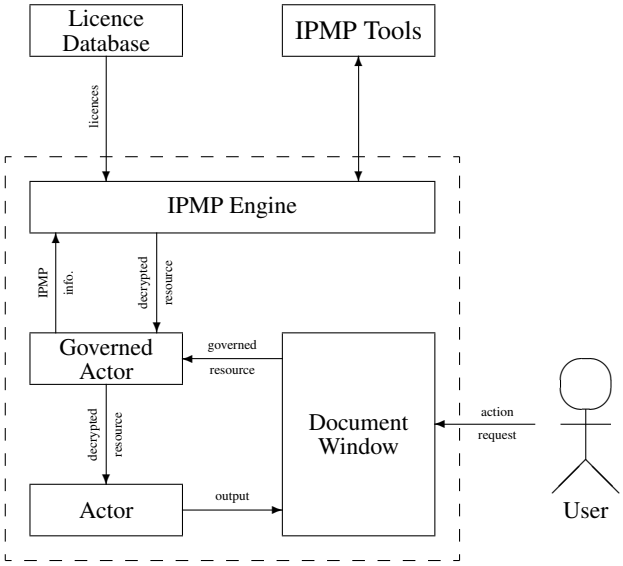


Figure 4: Performing a governed action with IPDoc. The dashed line shows the boundary of the terminal.

much like the Digital Media Project, but ISO/IEC 21000-4 does not specify any interface through which the terminal and tools communicate in order to achieve this.

SITDRM's approach to configuring the terminal's IPMP capabilities is derived from the typing structures of IPMP DIDL and MPEG REL. *Navigation tools* provide an implementation of digital item elements from the IPMP DIDL namespace, while *rights enforcement tools* provide implementations of the principals, rights, resources and conditions of an MPEG REL grant.

The general idea is that each element of IPMP DIDL and MPEG REL be associated with a tool that is responsible for parsing that element and presenting a standard interface to the DIDL and REL parsers, respectively, such that the parsers and IPMP engine can interact with the elements without needing to understand the elements themselves. The particular interfaces for each kind of tool will be described later in this section.

The vendor of any particular IPMP system is responsible for providing tools that implement all of the elements used in digital items and licences supported by that vendor.

## 4.1 Terminal Architecture

Figure 4 shows the architecture of our MPEG-21 terminal, known as "IPDoc". Throughout this paper, we assume that IPDoc is executing in a trusted, tamper-resistant environment such that any secret information obtained by IPDoc cannot be accessed outside the IPDoc process; we will give more detail about our environment in Section 6.1.

IPDoc is written in Java, and the user interacts with the software via a document window that allows him or her to navigate a digital item and perform actions upon its resources, such as "play", "print", etc.

When the user attempts to perform an action on a governed resource, the document window creates an "actor" (that is, a Java object that will carry out that action) as well as a "governed actor" that acts as a bridge between the real actor and the IPMP engine. The governed actor is a programming convenience that allows the document window to perform both governed and ungoverned actions using the same interface.

The governed actor instantiates the IPMP engine and supplies it with the IPMP information descriptor associated with the resource by its digital item. This information must include a list of *licence element builders* that are able to parse each element of the licence and to construct IPMP tools that enforce the licence elements that they represent. Vendors must implement a builder, element representation and IPMP tool for every kind of principal, condition, resource and right they support.

The IPMP engine attempts to find a set of licences that permit the action being requested, as described in detail in Section 5. If successful, the engine instantiates the principal validation and condition validation tools required to enforce those licences. Finally, if the validation tools permit the action to proceed, the output of the navigation and resource access tools is connected to the input of the real actor, and the actor's output is sent back to the document window.

## 4.2 IPMP Descriptors

Every governed element must contain an IPMP information descriptor that identifies the IPMP tools required for accessing the content of the element. In SITDRM, IPMP tools are identified by their Java class names, and an IPMP descriptor must contain the Java class names of all of the tool builders required to parse a licence for the associated digital item element, as shown in Figure 5.

As MPEG REL supports a very large number of elements, listing a builder for every possible element in an IPMP information descriptor could become very cumbersome. For this reason, it is possible to define a "super-builder" able to construct many different tools. We imagine that particular vendors might support a specific set of tools for which they could issue a single super-builder.

## 4.3 IPMP Tools

In SITDRM, IPMP tools are Java classes that can be loaded using the usual Java class loader. Every tool must implement an interface through which the IPMP engine communicates with the tool. We will describe the interface for each kind of tool informally in this section. The formal definitions for each interface are given in Appendix A.

```
<ipmpinfo:IPMPInfoDescriptor>

  <!-- rights enforcement tool builder -->
  <ipmpinfo:Tool>
    <ipmpinfo:ToolBaseDescription>
      <ipmpinfo:ToolID>
        au.com.smartinternet.drm.impl.
          SitLicenseElementBuilder
      </ipmpinfo:ToolID>
    </ipmpinfo:ToolBaseDescription>
  </ipmpinfo:Tool>

  <!-- XML encryption tool builder -->
  <ipmpinfo:Tool>
    <ipmpinfo:ToolBaseDescription>
      <ipmpinfo:ToolID>
        au.com.smartinternet.drm.impl.
          navigation.SitXmlEncryptionBuilder
      </ipmpinfo:ToolID>
    </ipmpinfo:ToolBaseDescription>
  </ipmpinfo:Tool>

</ipmpinfo:IPMPInfoDescriptor>
```

Figure 5: An IPMP information descriptor. The `SitLicenseElementBuilder` class is a "super-builder" able to build any of the rights enforcement tools used in our applications.

Since both the terminal and the IPMP tools have access to secret information, and, in general, exist inside distinct boundaries of trust, it is necessary for the terminal and tools to perform mutual authentication before they can safely co-operate with each other. We have not yet implemented mutual authentication in SITDRM, but we assume that the vendors of both terminals and IPMP tools provide certificates for their software by which mutual authentication can be performed. We will discuss this issue further in Section 7.4.1.

#### 4.3.1 The IPMP Tool Listener

The IPMP engine implements an *IPMP tool listener* interface through which IPMP tools can interact with the engine, the terminal and the user. The listener interface provides methods that allow tools to

- send cryptographic challenges to the terminal and user;
- obtain information about the action that they are being used to perform;
- display messages to the user and get input from the user; and
- access functions of the IPMP engine to obtain authorisation and cryptographic keys, as described in Section 5.

#### 4.3.2 IPMP Navigation Tools

SITDRM's DIDL parser implements an interface for digital item elements closely based on the `Node` interface of the Document Object Model [37]. This interface provides methods to retrieve and manipulate the children and siblings of an element. The interface is implemented internally for ungoverned digital item elements, but not for governed elements.

The interface for a governed element must be implemented by an IPMP navigation tool. Once the IPMP engine has determined that access to a governed element is permitted, it initialises the navigation tool with cryptographic information extracted from the licence as described in Section 5.3. The tool then provides a `Node`-like interface to the DIDL parser identical to that for an ungoverned item.

#### 4.3.3 Principal Validation Tools

Principal validation tools implement a single method that is called by the IPMP engine in order to check that the current human user of the terminal is, in fact, the principal referred to by a grant. Every implementation of the MPEG REL `Principal` type must be associated with a principal validation tool that is able to authenticate this kind of principal.

#### 4.3.4 Condition Validation Tools

Condition validation tools implement an interface that allows the IPMP engine to

- check whether or not the corresponding condition is true;
- be notified if the condition subsequently ceases to be true; and
- update stored state information for stateful conditions such as `ExerciseLimit` and `TrackReport`.

Every implementation of the MPEG REL `Condition` type must be associated with a condition validation tool able to perform all of the above functions.

#### 4.3.5 Resource Access Tools

Resource access tools act as a filter between a governed resource and the renderer for that resource. When supplied with the resource key for a governed resource, they provide an implementation of Java's `InputStream` interface that allows the renderer to access the governed resource as if it were unencrypted. Every implementation of the MPEG REL `Resource` type must be associated with a resource access tool.

Our interface assumes that the resources of a digital item are simple atomic objects such as JPEG images, text documents or raw data. A more sophisticated interface may be required if resources are permitted to be complex multimedia items in their own right, such as MPEG-4 files in which individual tracks may encrypted using independent keys and some header information may not be encrypted at all.

#### 4.3.6 Rights Tools?

SITDRM uses rights builders to parse instances of the MPEG REL `Right` type as for the other types, but we have not (yet) found any role for "rights tools" in enforcing a licence. When a user wishes to exercise a particular right, the terminal simply passes the name of the right to the IPMP engine, which checks that a licence granting that right exists. It is up to the terminal to ensure that the action it takes conforms to the definition of the right.

# 5 Licences

The MPEG-21 IPMP Components define an element called `RightsDescriptor` for including licences within the IPMP information descriptor of a governed digital item. The rights descriptor may contain an in-line licence; a reference to an external licence; or a reference to a web service that will supply a licence.

Associating a governed item with a fixed set of in-line or external licences, however, does not work very well in the scenarios for which we developed SITDRM. In our scenarios, the creator of a governed item does not necessarily know which licences will be used in order to access that item. In a super-distribution scenario, for example, a governed item might be passed around amongst a collection of friends, each of whom must apply to a licence issuer for his or her own individual licence to use the item.

Rather than rely on rights descriptors, SITDRM's IPMP engine attempts to build a dynamic *authorisation proof* at the time permission is request to perform an action. The authorisation proof is built using a combination of licences taken from some logical pool of licences, which may consist of one or more physical databases. In general, "databases" may include the contents of `RightsDescriptor` elements, though this has not yet been implemented in SITDRM.

An alternative approach – arguably more consistent with the intention of MPEG-21 – might be to specify a licence service for a governed item within its rights descriptor, and have that service build an authorisation proof from a pool of licences using the procedure described in this section. We found it more convenient, however, to allow the IPMP engine to build proofs directly by consulting whichever licence databases it had access to.

## 5.1 Authorisation Proofs

Formally, an MPEG REL *authorisation request* consists of a principal, a right and a resource together with the context information required to make an authorisation decision. An authorisation request will be accepted only if there is an *authorisation proof* for that principal, right and resource.

An authorisation proof is a properly-authorised *authorisation story*. An authorisation story consists of

- a primitive grant with principal, right and resource corresponding to an authorisation request;
- a grant or grant group from which the first grant can be derived (that is, obtained by reasoning in the obvious way); and
- optionally, an authoriser who has authorised the grant group.

An authorisation proof is therefore an authorisation story in which the first grant is correctly derived from the grant group, and the grant group is itself correctly authorised by the authoriser.

If an authorisation story does not contain an authoriser, the grant group must be derived from a set of *root grants* in order to be valid. The root grants are a set of grants that have been defined to be authorised for a specific application. A typical root grant, for example, might grant the system's licence issuer the right to issue licences.

SITDRM does not use authorisers in authorisation proofs. SITDRM's root grants allow any principal to issue licences concerning resources owned by that principal. It is not necessary to check root grants explicitly, however: since only the true owner of a resource possesses the resource key for that resource, only that owner can produce licences containing the valid resource key as described in Section 5.3.

## 5.2 Forming Authorisation Proofs

Obviously it would be infeasible for the enforcement engine to try every possible combination of licences in the pool to check whether or not that combination forms an authorisation proof for a particular request.

SITDRM's IPMP engine attempts to construct an authorisation proof using a depth-first search of the licences

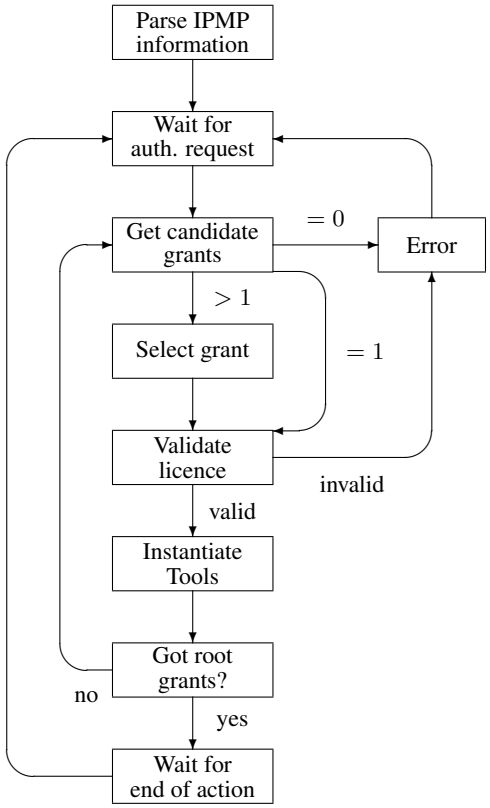


Figure 6: Constructing an authorisation proof.

in the pool, as shown in Figure 6. When it receives an authorisation request, it first searches the pool for a grant that might authorise that request. If the grant has been correctly signed by its issuer, it instantiates all of the principal validation and condition validation tools required to enforce that grant.

If a tool represents a principal or condition that depends on another grant in order to be authorised (e.g. the `PropertyPossessor` principal validation tool), the tool asks the IPMP engine to locate this grant. The cycle continues until either a valid authorisation proof is found, or the IPMP engine cannot find a grant required by one of the IPMP tools and the authorisation request is denied.

SITDRM searches for grants using relational database queries. Every grant is represented by a three-tuple (`Principal`, `Right`, `Resource`) in a database table. Given an authorisation request, the IPMP engine searches the database table for grants that might satisfy that request in the obvious way. Note that, since the logical licence pool may be distributed over several physical databases, it may be necessary to execute the query on several different databases and aggregate the results.

Other authors have proposed the use of inference engines to generate authorisation proofs [34] or "proof-carrying authorisations" [2]. In these systems, grants (equivalently, "facts" in proof-carrying authorisations) are represented as statements in some logic and an authorisation request is viewed as a theorem to be proved using those statements. These representations are arguably more elegant than ours. The general idea, however, is much the same: at each step of building an authorisation proof, the engine examines a claim and proceeds by attempting to prove all of the claims on which the current claim depends.

#### 5.2.1 Multiple Authorisation Proofs

It may happen that there exists more than one valid authorisation proof for a given authorisation request. This does not matter if authorisation proofs are stateless, and the inference engines described above do not appear to consider the possibility that there might be a need to distinguish between multiple valid authorisation proofs.

Digital rights management systems, however, make frequent use of stateful conditions. Suppose, for example, a user possesses two licences for the same item: one that allows him or her to use it on a pay-per-view basis, and another that requires him or her to have a subscription. In order to maintain the state information associated with these licences correctly, we need to take care which grants are exercised even though both imply authorisation to view the item.

Whenever SITDRM's IPMP engine finds two or more grants that might authorise a request, it instructs the terminal to ask the user to select which grant he or she wishes to exercise. In the above example, the user would be presented with a choice to pay for a single view of the item, or to take out a subscription which would remain valid for future views of the item.

## 5.3 Conveying Cryptographic Keys

MPEG-21 does not consider how terminals obtain the cryptographic information required to access governed resources – obtaining this information is presumably the responsibility of IPMP tools.

In most digital rights management systems, however, the distribution of cryptographic information is tightly bound to the distribution of licences: all of the OMA DRM, Marlin and Digital Media Project specifications, for example, have resource keys being distributed inside licences or equivalent objects. The philosophy here (we assume) is that a resource key is of no use unless there is a licence to authorise its use, and a licence is of no use without a resource key to make its execution possible.

SITDRM adopts this approach so that access to keys is tightly bound to the existence of authorisation proofs that authorise the use of those keys. While we can conceive of systems in which IPMP tools obtain keys independently of licences (such as ephemeral keys used to protect streaming media, for example), this seemed to be duplicating the work of licence distribution for the applications that we had in mind. Even if licences did not contain the keys themselves, we expect that they could be used to carry the information required to obtain the relevant keys.

The general idea is that every licence that grants a right must the contain the key required to exercise that right, encrypted by the public key of the entity that is permitted to exercise it. In this way, an authorisation proof forms a chain of keys from the entity that is permitted to perform an action, to the resource key that is required to access the resource on which the action is permitted. Without a valid authorisation proof rooted at a trusted actor, it is not possible to recover the resource key. A similar concept is used in Marlin, in which "nodes" take on the role of "licences" here.

Every resource $x$ is associated with a resource key $k_x$. This key is assumed to be chosen at random by the resource's creator, and is only distributed in an encrypted form as described here. Resource keys may be either symmetric or asymmetric.

Every grant that confers a right over a resource $x$ must contain $k_x$ encrypted according to an algorithm defined by the principal of the grant. For the principal types that we have implemented, the algorithms are:

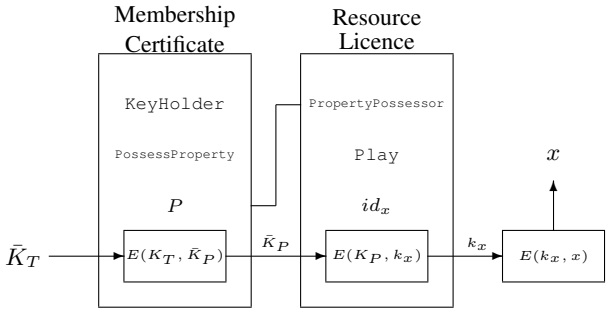


Figure 7: Obtaining a resource key via an authorisation proof.

**KeyHolder.** $k_x$ must be encrypted with the public key of the terminal on which the grant is to be exercised. Note this is not necessarily the same as the key supplied in the `KeyHolder` element, which may refer to a human principal rather than the terminal itself.

**PropertyPossessor.** Every property $P$ is assumed to be associated with a public and private key pair $(K_P, \bar{K}_P)$ and $k_x$ must be encrypted with $K_P$.

Similar algorithms could be defined for other kinds of principals.

In the system described by [32], for example, "membership certificates" award a property (that is, role) to a human user at a particular terminal using a `KeyHolder` principal. "Resource licences" award the right to perform an action to a role using a `PropertyPossessor` principal. An authorisation proof consists of a resource licence together with a membership certificate, as shown in Figure 7.

The resource licence contains the resource key $k_x$ encrypted by the public key of the property $K_P$, and the membership certificate contains the private key of the property encrypted by the public key of the terminal $K_T$. Given a valid authorisation proof to access a resource, the terminal can obtain the resource key by first decrypting the role's private key from the membership certificate, then using this to decrypt the resource key from the resource licence.

**Note.** Since SITDRM assumes that anyone may issue licences, a naïve implementation of the foregoing may allow dishonest users to construct apparently valid

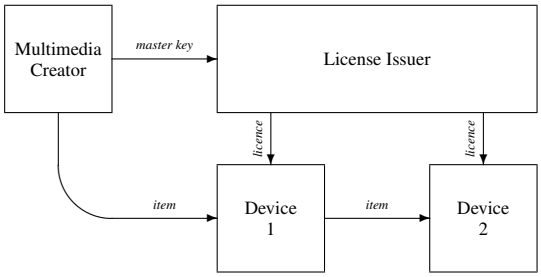


Figure 8: Using SITDRM for super-distribution.

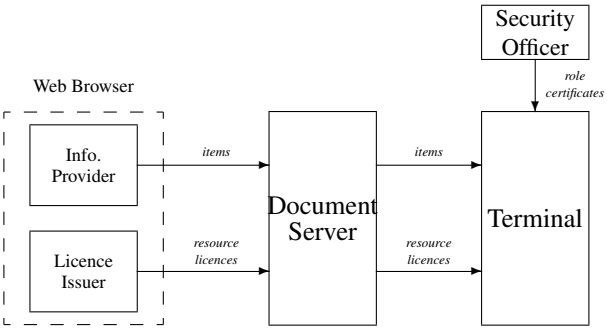


Figure 9: Using SITDRM for privacy protection.

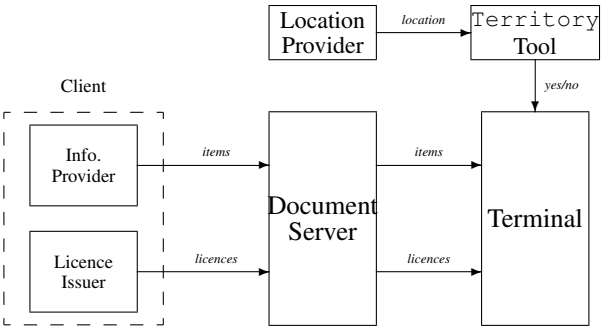


Figure 10: Using SITDRM for enterprise DRM.

licences for resources owned by other people. This can be prevented by including a nonce in both licences and resources, as described in the appendix of [32]. The nonce enables a terminal to check that the licence and the resource were, in fact, constructed by the same person, effectively verifying that SITDRM's root grant is satisfied.

# 6 SITDRM Applications

In addition to implementing the MPEG-21 IPMP Components and Rights Expression Language as described in the previous two sections, we implemented a collection of IPMP tools sufficient to build several demonstration applications. Each application is structured as a suite of software linked against the core SITDRM library.

Our original application for SITDRM was the classical copyright-protection application for which the MPEG-21 IPMP Components are intended (Figure 8). Multimedia creators could distribute their creations as governed digital items using any convenient method. Users wishing to access these items were then required to visit a web site from which they could purchase a licence. The system was later extended to allow users to transfer their licences from device to device such that only one copy of the licence was valid at any one time [25].

We later applied SITDRM to a privacy protection scenario in which individuals could submit contact information to a web site in the form of a governed digital item (Figure 9) [32]. At the same time as they submitted their information, individuals could create a licence that gave permission to use that information only to the people who filled particular roles within the organisation that received the information. It is difficult to make MPEG REL accessible to ordinary users, however, and the system was later adapted to generate MPEG REL licences from P3P policies that could be rendered in a more familiar way [29].

Finally, we applied SITDRM to an enterprise DRM scenario that allowed employees of an organisation to create governed items that could only be read by other employees located within a particular area (Figure 10). Even though a copy of the governed item could be taken anywhere on a mobile device, the IPMP engine refused permission to access it unless the device could be located within an approved region (typically, the organisation's offices).

We used the same security architecture and IPMP tools for all of the above applications though not all aspects of the architecture are present in all of the applications. In the remainder of this section, we will describe the par-

ticular security architecture and IPMP tools used in our applications.

## 6.1 Security Architecture

This section outlines the cryptographic architecture used in SITDRM. It is broadly similar to that used in other digital rights management systems, with some straightforward additions to cope with the particular structure of MPEG-21 and the trust management architecture of our applications.

SITDRM uses the well-known RSA algorithm for all public key cryptography, AES for all symmetric key cryptography, and SHA-1 for all cryptographic hashes. Other algorithms could be substituted for these without affecting the structure of the system.

### 6.1.1 Secure Environments and Terminals

The fundamental requirement for the system to be secure is that there exist tamper-resistant *secure environments* (computer systems) whose configuration can be attested to in the fashion of the Trusted Computing Group's specifications [35]. Every particular instance of a secure environment $E$ is assumed to have a public key $K_E$ and corresponding private key $\bar{K}_E$ such that the public key can be verified using some public key infrastructure and the private key is known only to the environment.

A secure environment may host one or more *terminals* that form the fundamental unit of trust in the SITDRM framework. All authorisation proofs must be rooted in a grant awarded to a terminal whose configuration has been attested to by a trusted secure environment.

In our applications, the "terminal" is a software process executing in a secure environment, but in principle it may also be a hardware device. Each terminal is assumed to be isolated from any other processes executing in the same environment, including the operating system and other terminals. More detail can be found in [33].

The secure environment assigns a public key $K_T$ to every terminal $T$ that it hosts. The corresponding private key $\bar{K}_T$ will only be released to the terminal if it is in the same configuration as it was when the key was created. The terminal can attest to being in this configuration using the attestation feature of the secure environment.

### 6.1.2 Users and Roles

Every human user $U$ (whether they are information creators or information users) of the system is assumed to possess a public key $K_U$ and corresponding private key $\bar{K}_U$ such that the public key can be verified by some public key infrastructure, and the private key is accessible only to the user[1]. This key pair is used purely for identifying the users – e.g. as a `KeyHolder` principal – and signing licences; it is not used for protecting resource keys since human users are not necessarily trusted to abide by the terms of licences.

Roles are represented as "properties" in MPEG REL, that is, membership of a role $R$ is expressed as the possession of a property $P_R$ using the `PossessProperty` right. Every role $R$ is associated with a public key $K_R$ and private key $\bar{K}_R$ such that the public key can be verified by some public key infrastructure and the private key is known only to the security officer responsible for assigning people to roles.

SITDRM does not explicitly support termination of a user's role membership, though it is possible to limit the duration of a user's role membership using the `ValidityInterval` condition. MPEG REL supports revocation via the `RevocationFreshness` condition, which obliges the IPMP engine to check whether or not the licence has been revoked before permitting the action to proceed. We have not yet implemented a condition validation tool for this condition, however.

### 6.1.3 Resource Keys

There are two kinds of resources in the existing SITDRM applications: multimedia resources and properties (roles). For properties, the resource key is the private key of the role so that a terminal that possesses a valid licence to use that role may access information encrypted for that role, as described in Section 5.3.

The resource key for multimedia items is a symmetric key, and in general it can be chosen at random. In our applications, however, it was convenient to make the resource key to be a one-way function of the item's identi-

[1] Technically, the terminal will only allow the private key to be used if it is satisfied that the key's owner is logged in using a password or similar, since we cannot expect human users to compute cryptographic functions themselves.

fier $id_x$ and the item's owner's *master key* $\mathcal{K}_{\mathcal{U}}$:

$$k_x = \mathrm{HMAC}(\mathcal{K}_{\mathcal{U}}, id_x).$$

This allows the same resource key to be generated at multiple sites by sharing the master key. This was particularly useful in the copyright protection application in which we had governed items being created by a packaging application and licences for those items being generated at some future date by a web site.

#### 6.1.4 Security Argument

The structure of authorisation proofs varies from application to application, but the fundamental requirement is that licences be issued only to properties or terminals that are trusted by the issuer. The resource key is encrypted by the public key of that property or terminal, as described in Section 5.3.

Trust in properties must be established using some out-of-band mechanism; it amounts to trusting the person responsible for managing a property. In particular, it amounts to trusting that person to issue the property only to terminals or other properties that are entitled to possess the property.

Trust in terminals is established via the secure environment that executes them. Once trust in the secure environment has been established using the mechanism specified by the designers of that environment, the environment can vouch for the configuration and public key of terminals executing on it. If the configuration is acceptable to the issuer, the issuer can safely issue a licence to that terminal.

An argument for the security of the system can be mounted by induction on the authorisation proof. The base case is satisfied if every authorisation proof is rooted at a trusted terminal as required above. The inductive hypothesis requires that a key in a grant only be accessible to the entity to which that grant has been awarded. This is true for the algorithms we described in Section 5.3.

### 6.2 IPMP Tools

#### 6.2.1 Navigation Tools

Our current implementation supports only one navigation tool, in which the `Contents` child of a governed element contains the protected document tree encrypted using the XML Encryption specification, as shown in Figure 2. Once the tool is supplied with the key required to access the protected document, it decrypts the document into memory and acts as a wrapper around the Document Object Model for that tree. This tool is similar to the "fixed DRM tools" used in the Digital Media Project's specification.

#### 6.2.2 Principal Validation Tools

Our current implementation supports two kinds of principal:

- the `KeyHolder` principal validation tool checks that the user logged in at the terminal possesses the private key corresponding to the public key supplied within the `KeyHolder` element; and
- the `PropertyPossessor` principal validation tool checks that the terminal has a valid licence granting the `PossessProperty` right over the property URI supplied within the `PropertyPossessor` element. Properties are used for implementing roles in the sense of role-based access control.

#### 6.2.3 Condition Validation Tools

MPEG REL supports a very large number of conditions, and we have only implemented those that have been useful in the specific applications to which we have applied SITDRM, including:

- the `ValidityInterval` condition checks that the current time (as reported by a secure clock) falls within a given interval;
- the `Territory` condition checks that the terminal is located in a particular geographic location;
- the `ContactMethods` condition checks that a particular mode of communication (telephone, e-mail, etc.) is in use for a `Contact` right [32]; and
- the `AllConditions` condition acts as a container for an arbitrary collection of other conditions, all of which must be satisfied.

#### 6.2.4 Resource Access Tools

Our current implementation supports only one resource access tool, in which the resource is encrypted as a whole using the AES cipher.

# 7 Discussion

## 7.1 Configuration-Driven vs. Full-Format Inter-operability

MPEG-21's approach to inter-operability in its IPMP Components differs markedly from the approach taken in the Rights Expression Language and Rights Data Dictionary. In the latter two parts, MPEG-21 exhaustively defines a detailed language and vocabulary that is supposed to encompass all rights management, while in the former MPEG-21 has avoided defining anything at all. In fact, the IPMP Components appear to resurrect problems stemming from under-specification in the original MPEG-4 IPMP specification.

While the IPMP Components appear to be geared towards IPMP tools that perform primitive functions such as encryption and watermarking, the specification does explicitly contemplate IPMP tools that implement "a complete IPMP system". This, taken together with MPEG REL, arguably lacks modularity: if we take "complete IPMP systems" to mean systems such as OMA DRM and Marlin, such systems may bring their own rights expression languages with them.

SITDRM binds the IPMP Components to the Rights Expression Language, both in the way IPMP tools are defined and in the way cryptographic information is distributed. Binding licences and cryptographic information is consistent with the way other digital rights management systems work, while binding the definition of IPMP tools to MPEG REL removes the possibility for tools to introduce a third-party language. SITDRM's approach is arguably rather elaborate compared to a monolothic approach, though it does have some advantages outlined in the next section.

## 7.2 "GURMS"

All of the published specifications for digital rights management systems, including those from MPEG, the Open Mobile Alliance, Marlin and the Digital Media Project, focus exclusively on copyright protection. This is understandable given the background and aims of the bodies that produced these specifications, and the clear need for inter-operability in existing copyright-sensitive applications.

With digital rights management also being applicable to sensitive corporate information and individuals' private information, however, we might wonder if a "grand unified rights management system" covering all of these applications is possible or appropriate. Microsoft, in fact, has been reported to be moving in this direction with its Windows Rights Management Services product [10].

SITDRM was originally designed for a copyright protection scenario of the kind contemplated by MPEG and others, and later adapted to privacy and enterprise DRM. The user-level applications changed from project to project, but the core digital rights management functions described in the present paper have remained constant throughout all of the projects. While we needed to introduce new rights and conditions to MPEG REL to cater for privacy scenarios, we haven't modified the cryptographic architecture of the system, or the structure of the digital item declaration or rights expression languages.

Inter-operability between privacy protection systems and enterprise DRM systems has obvious benefits similar to those in copyright protection, and our experience suggests that a grand unified rights management system might be practical, whether it is based on the MPEG specification or another.

One unanticipated advantage of the use of rights enforcement tools in SITDRM is that the core REL interpreter did not need to be modified in order to introduce new rights and conditions when we applied it to scenarios that weren't considered in the design of MPEG REL. This demonstrates some of the value of layered architectures compared to the traditional monolithic architectures.

The main point of contention here is whether or not we should introduce a "purpose" component to the rights expression language, as used in privacy protection languages such as P3P [36] and EPAL [30]. The concept of a "purpose" may also be useful in copyright protection – notably in the context of fair dealing exceptions in the copyright law of countries such as the UK and Australia, which allow copying for certain purposes such as education and news reporting.

It would be simple enough to add a `Purpose` element to MPEG REL, but SITDRM chose to treat "purpose" as being a kind of condition: "Alice may telephone Bob's number on the condition that her purpose is to renew his insurance", for example. Since the licence evaluator can only check a "purpose" in so far as it can be described as a logical condition subject to computer reasoning, this approach seemed appropriate.

It should be noted, however, that the number of scenarios to which we have so far applied SITDRM is relatively small: generally, one or two scenarios from each of copyright protection, privacy protection and enterprise DRM. It remains to be seen how SITDRM's approach works with more exotic conditions such as `PrerequisiteRight` or more complex licensing schemes.

## 7.3 On KeyHolder Principals

SITDRM's approach to resource key encryption for the `KeyHolder` principal is arguably anomalous and confusing, since it has the principal being associated with one key (the human user's) but the resource key being encrypted by an another key (the terminal's). The algorithm for the `PropertyPossessor` principal is much more straightforward in that the resource key is encrypted by the public key of the principal.

SITDRM's `KeyHolder` principal comes about because MPEG REL allows key holders to be humans, but the security of the system demands that resource keys be inaccessible to humans. The algorithm would be more straightforward if the `KeyHolder` principal referred to the terminal, so that the resource key could be encrypted with the public key of the principal as for the `PropertyPossessor` principal.

This approach would require human users to be represented in some other way, for example, by representing them as a property that could be possessed by all of the terminals that they own, as in the "personal entertainment domain" of Koster, et al. [24]. While this may make licensing appear more complicated, it arguably leads to a more elegant security architecture.

## 7.4 Future Work

### 7.4.1 Tool Distribution and Authentication

IPDoc currently requires IPMP tools to be installed manually, and does not support mutual authentication between tools and terminals. MPEG's intention is that tools be installed automatically and transparently as they are required, and mutual authentication is necessary for the system to be secure.

Automatic detection and installation of plug-ins is already supported by popular web browsers and it seems unlikely that implementing a similar feature for IPMP tools in a digital rights management terminal would present any unusual challenges apart from the need for mutual authentication.

While cryptographic mechanisms for mutual authentication are well-known, constructing an effective tool authentication architecture may be a significant challenge. An effective architecture must support

- a very large installed base of terminals and tools provided by many different manufacturers;
- tools and terminals that run on many different computing platforms; and
- revocation and renewal for tools and terminals that have been compromised.

### 7.4.2 Efficient Authorisation Proofs

Our current authorisation proof builder – and the others of which we are aware – have three major weaknesses in terms of efficiency.

First, many grants contain conditions that cannot be satisfied by the terminal, for example, because they require other grants to exist that do not or because they have exhausted some stateful condition. In a few cases, the IPMP engine can eliminate these when constructing an authorisation proof, but in most cases it cannot eliminate them until it attempts to execute the condition validation tool. This leads the IPMP engine to produce unusable authorisation "proofs" that users have to eliminate manually.

Second, the IPDoc terminal has menu options for many different actions but only a few of them might be permitted on any particular resource. Applications typically disable unusable menu items, but for IPDoc this would mean

it had to execute the IPMP engine for every possible action on its menu bar. This may be a very expensive operation for terminals that support a large number of actions.

Third, the licence pool may be distributed over a network, with the result that the IPMP engine needs to be constantly on-line and may generate large amounts of network traffic. An improved proof-building algorithm may be able to reduce the engine's reliance on the network by both reducing the number of queries that need to be made in the first place, and by caching the results of frequent queries in a local licence pool.

Our IPMP engine is designed to be a generic engine able to find an authorisation proof, if it exists, for any request over any pool of licences. If we look at the particular applications that use the IPMP engine, however, it is often easy to see much more efficient methods of producing valid authorisation proofs since these applications have a particular structure to their authorisation proofs. Ideally, we would like our applications to be able to communicate "hints" to the IPMP engine that allowed it to take advantage of application-specific knowledge without having to write a specialised proof builder for every application.

# 8 Conclusion

The MPEG-21 Intellectual Property Management and Protection Components provide the skeleton of a configuration-driven inter-operable digital rights management system, but do not specify an interface to IPMP tools. SITDRM defines an interface that binds tools to the MPEG Rights Expression Language, and provides a generic digital rights management framework that has been applied to scenarios in copyright protection, privacy protection and enterprise DRM.

The principal advantage of SITDRM's tool interface is that it allows a skeleton MPEG REL interpreter to be easily extended to support the set of principals, rights, resources and conditions appropriate to a particular application area. This architecture allowed us to apply the MPEG-21 IPMP components to a variety of scenarios without needing to build a monolithic universal MPEG REL interpreter.

SITDRM is not a complete system, however, and lacks some of the trust infrastructure that would need to be supported by a fully-functional digital rights management platform. Furthermore, the efficiency and usability of its IPMP engine could be greatly improved if it were able to take advantage of application-specific knowledge about the structure of authorisation proofs. We leave this as future work.

# 9 Acknowledgements

This work was partly funded by the Co-operative Research Centre for Smart Internet Technology, Australia. In particular, the author would like to thank Hartono Kurnio, Qiong Liu, Adam Muhlbauer, Rei Safavi-Naini, Farzad Salim and Sid Stamm for their contribution to projects surrounding the SITDRM software. The author would also like to thank the anonymous reviewers for their helpful comments on the paper.

# References

[1] A. Arnab and A. Hutchison. Requirement analysis of enterprise DRM systems. In *Information Security South Africa*, 2005.

[2] L. Bauer, M. A. Schneider, and E. W. Felten. A general and flexible access-control system for the Web. In *USENIX Security Symposium*, pages 93–108, 2002.

[3] G. A. Brox. MPEG-21 as an access control tool for the national health service care records service. *Journal of Telemedicine and Telecare*, 11:23–25, 2005.

[4] ContentGuard. Extensible Rights Markup Language. `http://www.xrml.org`, 2004.

[5] ContentGuard. MPEG REL SDK 1.0 for Java. `http://www.contentguard.com/MPEGREL_sdk.asp`, 2007.

[6] Coral Consortium. Welcome to Coral Consortium. `http://www.coral-interop.org`, 2007.

[7] Digital Media Project. Interoperable DRM Platform, version 2.1. `http://www.dmpf.org/project/ga14/idp-21.html`, 18 May 2007.

[8] Digital Media Project. Chillout – the interoperable DRM platform. http://chillout.dmpf.org, 2007.

[9] Digital Media Project. DMP home page. http://www.dmpf.org, 2007.

[10] DRMWatch. Microsoft releases Windows Rights Management Services. *DRMWatch*, 6 November 2003. http://www.drmwatch.com/drmtech/article.php/3105021.

[11] C.-W. Fan, F.-C. Chang, and H.-M. Hang. An MPEG-4 IPMPX design and implementation on MPEG-21 test bed. In *IEEE International Symposium on Circuits and Systems*, pages 4550–4553, 2005.

[12] G. L. Heileman and P. A. Jamkhedkar. DRM interoperability analysis from the perspective of a layered framework. In *ACM Workshop on Digital Rights Management*, pages 17–26, 2005.

[13] C.-C. Huang, H.-M. Hang, and H.-C. Huang. MPEG IPMP concepts and implementation. In *Pacific Rim Conference on Multimedia*, pages 344–352, 2002.

[14] International Standards Organisation. Information technology – multimedia framework (MPEG-21) – part 1: Vision, technologies and strategy. ISO/IEC 21000-1:2001.

[15] International Standards Organisation. Information technology – multimedia framework (MPEG-21) – part 2: Digital item declaration. ISO/IEC 21000-2:2003.

[16] International Standards Organisation. Information technology – multimedia framework (MPEG-21) – part 4: Intellectual property management and protection components. ISO/IEC 21000-4:2006.

[17] International Standards Organisation. Information technology – multimedia framework (MPEG-21) – part 5: Rights expression language. ISO/IEC 21000-5:2004.

[18] International Standards Organisation. Information technology – multimedia framework (MPEG-21) – part 8: Reference software. ISO/IEC 21000-8:2006.

[19] P. A. Jamkhedkar and G. L. Heileman. DRM as a layered system. In *ACM Workshop on Digital Rights Management*, pages 11–21, 2004.

[20] M. Ji, S. M. Shen, W. Zeng, T. Senoh, T. Ueno, T. Aoki, Y. Hiroshi, and T. Kogure. MPEG-4 IPMP extension for interoperable protection of multimedia content. *EURASIP Journal on Applied Signal Processing*, 2004(14):2201–2213, 2004.

[21] K. Karpouzis, I. Maglogiannis, E. Pappaioannou, D. Vergados, and A. Rouskas. MPEG-21 digital items to support integration of heterogeneous multimedia content. *Computer Communications*, 30:592–607, 2007.

[22] S. Kenny and L. Korba. Applying digital rights management systems to privacy rights. *Computers & Security*, 21:648–664, 2002.

[23] R. H. Koenen, J. Lacy, M. Mackay, and S. Mitchell. The long march to interoperable digital rights management. *Proceedings of the IEEE*, 92:883–897, 2004.

[24] P. Koster, F. Kamperman, P. Lenoir, and K. Vrielink. Identity based DRM: Personal entertainment domain. In *IFIP Conference on Communications and Multimedia Security*, pages 42–54, 2005.

[25] Q. Liu, R. Safavi-Naini, and N. P. Sheppard. A license transfer system for supporting content portability in digital rights management. In *IFIP International Information Security Conference*, pages 189–204, Chiba, Japan, 2005.

[26] Marlin Developer Community. Marlin – core system specification version 1.2. http://www.marlin-community.com, 12 April 2006.

[27] Open Mobile Alliance. OMA DRM v2.0 approved enabler, 3 March 2006.

[28] Open Platform Initiative for Multimedia Access. Welcome to OPIMA. http://www.chiariglione.org/leonardo/standards/opima/index.htm, 2000.

[29] F. Salim, N. P. Sheppard, and R. Safavi-Naini. Enforcing P3P policies using a digital rights management system. In *International Workshop on Privacy Enhancing Technologies*, Ottawa, Canada, 2007.

[30] M. Schunter and C. Powers. The Enterprise Privacy Authorization Language (EPAL 1.1). `http://www.zurich.ibm.com/security/enterprise-privacy/epal`, 2003.

[31] C. Serrão, J. M. S. Dias, and P. Kudamakis. From OPIMA to MPEG IPMP-X: A standard's history across R&D projects. *Signal Processing: Image Communication*, 20:972–994, 2005.

[32] N. P. Sheppard and R. Safavi-Naini. Protecting privacy with the MPEG-21 IPMP framework. In *International Workshop on Privacy Enhancing Technologies*, pages 152–171, Cambridge, UK, 2006.

[33] S. Stamm, N. P. Sheppard, and R. Safavi-Naini. Implementing trusted terminals with a TPM and SITDRM. In *International Workshop on Run-Time Enforcement for Mobile and Distributed Systems*, Dresden, Germany, 2007.

[34] C. H. Suen. Efficient design of interpretation of REL license using expert systems. In *Computer Communications and Networking Conference Workshop on Digital Rights Management Impact on Consumer Communications*, 2007.

[35] The Trusted Computing Group. Trusted Computing Group: Home. `http://www.trustedcomputinggroup.org`, 2005.

[36] W3 Consortium. Platform for Privacy Preferences (P3P) project. `http://www.w3.org/P3P`, 2004.

[37] W3 Consortium. Document Object Model (DOM). `http://www.w3.org/DOM`, 2005.

# A SITDRM IPMP Tool API

SITDRM requires that IPMP tools implement a Java interface summarised in this appendix. The technical details of the classes used have been omitted for ease of exposition.

SITDRM is an on-going project and the interface described here is unlikely to be final or complete. We have not yet defined any mutual authentication methods, for example, and future condition and principal implementations may require the `IPMPToolListener` interface to be expanded in order to supply more information about the tool's environment.

## A.1 IPMPToolListener

The `IPMPToolListener` interface is implemented by the IPMP engine and provides call-back methods by which IPMP tools can gain information about the environment in which they are running and about the action that they have been instantiated to govern (referred to as "the action" below).

**`decrypt`**$(t, m)$
Ask the terminal or principal to decrypt a message.
$t$ – the terminal or principal
$m$ – the message to decrypt

**`forceEndGovernedAction`**$()$
Forcibly end the action.

**`getChoice`**$(m, t, c, i)$
Get a choice from the user.
$m$ – the question to ask the user
$t$ – the title for the dialogue box
$c$ – the list of choices
$i$ – the default choice

**`getConditionValidationTool`**$()$
Get the condition validation tool that is governing the action.

**`getGrant`**$(p, r, a)$
Search for a grant with a particular principal, resource and right.
$p$ – the principal
$r$ – the resource
$a$ – the right

**`getKeyInfo`**$()$
Get the key information from the grant that

authorised the action.

**getMeansUri**()
Get the means being used to carry out the action.

**getPrincipalValidationTool**()
Get the principal validation tool that is governing the action.

**getSessionId**()
Get the session in which the action is occurring.

**getTaskUri**()
Get the task to be carried out by the action.

**showMessage**$(m, t)$
Display a message to the user.
$m$ – the message
$t$ – the title for the dialogue box

## A.2 IPMPTool

The `IPMPTool` interface must be implemented by all IPMP tools. The tool is configured once at the time it is instantiated, then notified every time a new action begins. In a complete implementation, this interface would also include mutual authentication functions.

**configure**$(e, t, l)$
Configure the tool for use.
$e$ – the secure environment
$t$ – the terminal's identifier
$l$ – the `IPMPToolListener`

**init**()
Initialise the tool for a new action.

## A.3 IPMPNavigationTool

The `IPMPNavigationTool` interface allows the DIDL parser to navigate a governed element as if it were an ungoverned element. To this end, its interface is based very closely on the `Node` interface of the Document Object Model. Other aspects of the `Node` interface could be implemented but we have not needed them in our applications.

**appendChild**$(p, c)$
Append a child node to a parent node.
$p$ – the parent node
$c$ – the child node

**getFirstChild**$(p)$
Get the first child of a node.
$p$ – the parent node

**removeChild**$(p, c)$
Remove a child node from a parent node.
$p$ – the parent node
$c$ – the child node

**replaceChild**$(p, n, o)$
Replace a child node of a parent node.
$p$ – the parent node
$n$ – the new child node
$o$ – the child node to replace

## A.4 IPMPPrincipalValidationTool

The `IPMPPrincipalValidationTool` interface allows the IPMP engine to authenticate the principal of a grant.

**endAction**$(f)$
Called at the completion of an action.
$f$ – success flag

**validatePrincipal**()
Attempt to authenticate the principal.

## A.5 IPMPConditionValidationTool

The `IPMPConditionValidationTool` interface allows the IPMP engine to evaluate the condition of a grant. It is the tool's responsibility to update any state information associated with the condition when the `endAction` method is called.

**beginAction**()
Called at the beginning of an action to evaluate the condition.

**endAction**($f$)
Called at the completion of an action.
$f$ – success flag

## A.6 IPMPResourceAccessTool

The `IPMPResourceAccessTool` interface allows the IPMP engine to engine to create an instance of the Java `InputStream` class that reads a governed resource as if it were the equivalent ungoverned one. The tool obtains the key required to decrypt the resource by querying the IPMP engine for the key information in the grant that approves access to the resource.

**createFilterInputStream**($s$)
Create an `InputStream` that performs any decryption, etc. required to access the resource.
$s$ – the `InputStream` connected to the raw resource