Content Protection for Recordable Media Specification

Network Download Book

Intel Corporation
International Business Machines Corporation
Matsushita Electric Industrial Co., Ltd.
Toshiba Corporation

Revision 0.90
August 5, 2004

This page intentionally left blank.

Preface

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. IBM, Intel, MEI, and Toshiba disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

This document is an intermediate draft and is subject to change without notice. Adopters and other users of this specification are cautioned that products based on it may not be interoperable with the final version or subsequent versions thereof.

Intellectual Property

Implementation of this specification requires a license from the 4C Entity, LLC.

Contact Information

Please address inquiries, feedback, and licensing requests to the 4C Entity, LLC:

- Licensing inquiries and requests should be addressed to cprm-licensing@4Centity.com.
- Feedback on this specification should be addressed to cprm-comment@4Centity.com.

The URL for the 4C Entity, LLC web site is http://www.4Centity.com.

This page intentionally left blank.

Table of Contents

This page intentionally left blank.

Chapter 1
Introduction

1. Introduction

1.1 Purpose and Scope

The *Content Protection for Recordable Media Specification* (CPRM) defines a robust and renewable method for protecting content stored on a number of physical media types. The specification is organized into several "books". The *Introduction and Common Cryptographic Elements* book provides a brief overview of CPRM, and defines cryptographic procedures that are common among its different uses. This document (the *Network Download Book*) specifies additional details for using CPRM technology as part of an electronic delivery of content that is encrypted by CPRM as specified in one of the other books in this specification. Note that such use of CPRM for electronic delivery may involve a separate license category with separate associated fees, as indicated by the CPRM license.

1.2 Document Organization

This specification is organized as follows:

- Chapter 1 provides an introduction.
- Chapter 2 lists abbreviations, acronyms, and notation used in this document.
- Chapter 3 describes the use of CPRM during a network download.
- Appendix A is an informative section, giving an example use of the network download protocol using Hypertext Transfer Protocol (HTTP).

1.3 References

This specification shall be used in conjunction with the following publications. When the publications are superceded by an approved revision, the revision shall apply.

4C Entity, LLC, *CPRM License Agreement.*

4C Entity, LLC, *CPRM Specification: Introduction and Common Cryptographic Elements, Revision 1.00*

4C Entity, LLC, *Content Protection System Architecture White Paper, Revision 0.81*

1.4 Future Directions

This document may describe other example protocols for media binding, in addition to the HTTP protocol described in the Appendix.

1.5 Notation

Except where specifically noted otherwise, this document uses the same notations and conventions for numerical values, operations, and bit/byte ordering as described in the *Introduction and Common Cryptographic Elements* book of this specification.

This page intentionally left blank.

Chapter 2
Abbreviations and Acronyms

2. Alphabetical List of Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document:

4C	4 Companies (IBM, Intel, MEI, and Toshiba)
C2	Cryptomeria Cipher
CCI	Copy Control Information
Cmd.	Command
CPRM	Content Protection for Recordable Media
HTTP	Hypertext Transfer Protocol
ID	Identifier
LLC	Limited Liability Company
lsb	Least Significant Bit
MKB	Media Key Block
MKBext	Media Key Block Extension
msb	Most Significant Bit
PC	Personal Computer
URL	Uniform Resource Locator
XOR	Exclusive-OR

In URLs, optional arguments are denoted by surrounding them in brackets [].

This page is intentionally left blank.

Chapter 3
CPRM for Network Download

3. CPRM for Network Download

3.1 Basic Approach

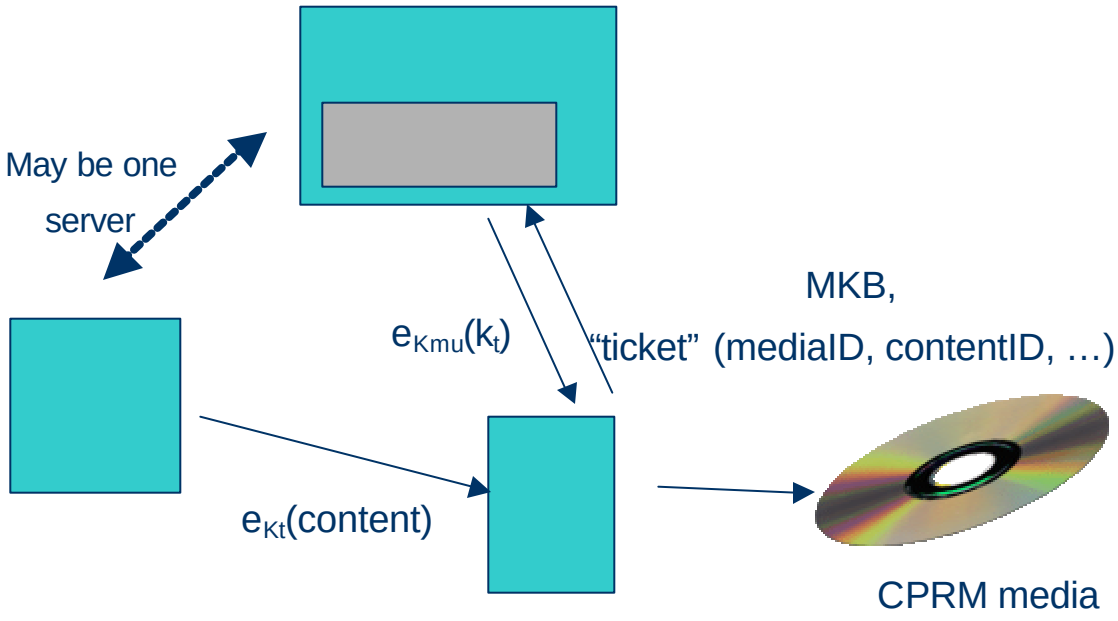

Figure 1: Example System for CPRM Network Download

Figure 1 shows an example system. It consists of a Service Provider, a Security Provider, and a Client, although the Service Provider and the Security Provider functions may be performed by a single entity.

The Service Provider delivers content prepared for recording on CPRM media. This means that the content is encrypted with C2 using a randomly selected title key, K_t, as specified for the given format in one of the other books of this specification. Although this figure shows a DVD recordable disc as an example, the target could be any CPRM-compliant media. The content, as stored on the Service Provider, is not yet playable by CPRM devices. It is missing an additional step, involving the Security Provider.

That missing step is the *media binding* operation. The Security Provider receives the Media ID and the Media Key Block for a specific piece of CPRM media from the client. Using them, it calculates the Media Unique Key (K_{mu}). Using the Media Unique Key, it encrypts the Title Key, as specified for the given format in one of the other books of this specification. It sends the encrypted Title Key to the client, and the Client places the encrypted title key along with the encrypted content from the Service Provider on the CPRM media, as specified in the corresponding CPRM book. At this point, the content can be played by any CPRM-compliant playing device.

How the Security Provider learns the Title Key is implementation-specific. For example, the Security Provider could have a data base associating pieces of content to title keys. Alternatively, the content from the Service Provider might contain the title key, encrypted using a key that only the Security Provider knows. This will be referred to as the *prepared title key*, to distinguish it from the encrypted title key that is the final output of the Security Provider. In the latter case, that prepared title key data would be transmitted from the Client to the Security Provider as part of the *ticket* data.

3.2 The Ticket

The term "ticket" will be used to describe the data that passes from the Client to the Security Provider. The ticket serves as a "proof of purchase" or "proof of entitlement" to a piece of content. At a minimum, the ticket shall contain the following information:

1. The Media ID to which the content will be bound.
2. The identification of the content. For example, this may be a content identifier or a prepared Title Key.

That might be the entire ticket: for example, a system might operate by mailing special blank media to its subscribers. Possession of the special media is proof of entitlement, and, since the Security Provider would only bind to special media, no cryptographic authentication on the ticket would be needed. However, in most systems the ticket is more involved, and may contain things like a ticket ID, a store ID, and/or a Client ID, and may be digitally signed to prove authenticity.

How the ticket is delivered to the Client is outside the scope of this specification. For example, it may be delivered by an online store as part of a financial transaction. In that case, the consumer's client software would transmit the Media ID when it purchased the content.

After verifying that the ticket is authentic, the Security Provider performs the media binding operation, confident that the user is entitled to the content.

Note that because the Media ID is included in the ticket, a "replay attack" is not enabled; if the client fails to receive the encrypted title key for any reason, the media binding operation can be safely repeated. The resend does not constitute an additional copy, because the ticket specifies a single piece of media.

3.3 Content Delivery

This specification is applicable to a variety of content delivery methods. For example, the encrypted content may be delivered ahead of time, before it is actually purchased, and the purchase would involve only the delivery of the encrypted title key. Alternatively, the delivery of the content may wait until the user has a valid ticket.

In some cases, the content may not even be delivered by electronic download. For example, the content may be delivered on prerecorded media. In that case, the client software would copy the prerecorded content onto recordable CPRM media (if it were not already there), and then connect to the Security Provider to obtain the encrypted title key to make it playable.

3.4 Client-to-Security-Provider Protocol

The particular syntax of the Client to Security Provider protocol is implementation specific. An example syntax using HTTP is provided for informational purposes in Appendix A. Regardless of the syntax, the protocol must proceed as follows:

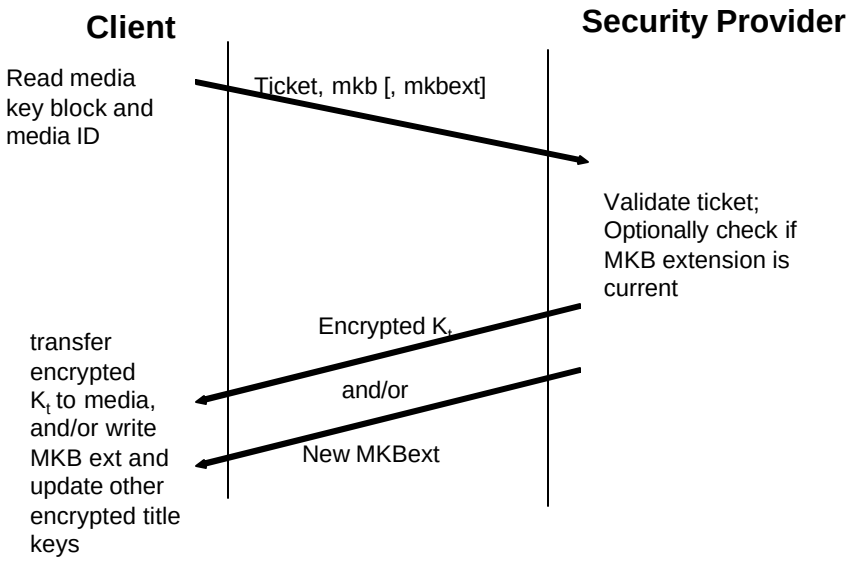

Figure 2 – Normal Client-to-Service-Provider Flow

The client connects to the Security Provider, transmitting the ticket and the Media Key Block on the user's media. Additionally, if the media currently has a Media Key Block Extension, it is also transmitted.

The Security Provider normally responds with the encrypted title key, which the Client records on the media.

Not all CPRM media and content types have defined Media Key Block Extensions, but many have. In that case, the Security Provider may want to update the Media Key Block Extension on a given piece of media. This specification does not require that the Security Provider support this function; however, the Security Provider operates on behalf of the content owners, and the content owners have a vested interest in having Media Key Block Extensions be at the latest level. Therefore, a Security Provider might be required to support updating the Media Key Block Extension by the content owners. Instead of or addition to responding with the encrypted title key, such a Security Provider responds with the Media Key Block Extension itself. The Client writes the Media Key Block Extension in the appropriate file on the media. If the encrypted title key has not yet been transmitted, the Client simply retries the original request, this time transmitting the new Media Key Block Extension. This will be acceptable to the Security Provider, and the modified request will succeed.

3.4.1 Updating Other Encrypted Title Keys

If media already contains other content, a new Media Key Block Extension will render that content inaccessible unless the corresponding encrypted title keys are updated. If the Client has a set of device keys, it can perform such an update itself. However, there are many advantages if the Client can remain secret-free, so the Security Provider can also perform such an update.

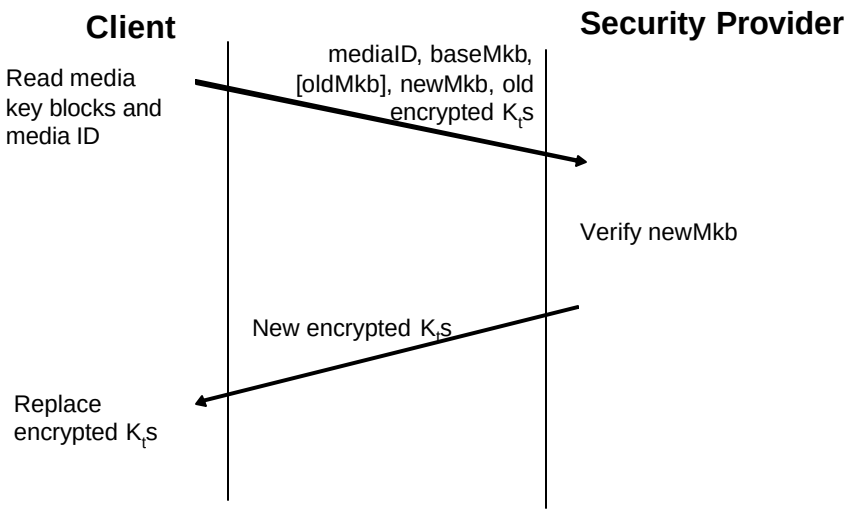

Figure 3 – Updating Existing Encrypted Title Keys

If the Security Provider is performing the update of the existing title keys, the protocol proceeds as shown in Figure 3. The Client transmits the Media ID, the base Media Key Block, the existing Media Key Block Extension (if it exists), and the new Media Key Block Extension. The Client also transmits the encrypted title keys.

The Security Provider shall verify that the new Media Key Block Extension is one that it is currently using. In other words, the Security Provider must not act as a general-purpose "transformation engine" from one Media Key Block to another. It is *not* required, however, to verify that the Media ID corresponds to a Media ID used in a previously validated ticket.

It is an error if the new Media Key Block Extension is out-of-date or is not one that originated from the Security Provider. The Media Key Block Extension should not be out of date if the Client is acting in a timely manner; the Media Key Block Extension would have been the one just delivered. Nonetheless, if an error occurs, the Client shall simply restart the original transaction, using the old Media Key Block (and Extension). This will force the Client and the Security Provider to synchronize on Media Key Blocks.

3.4.2 Caching Media Key Blocks

It is permissible for the Security Provider to cache Media Key Blocks and Media Keys, which greatly reduces the amount of data that the Client usually needs to transfer to the Security Provider. To this end, a system may be designed for the Client to simply transmit the 8-byte verification data (the data part of the Verification Record in the Media Key Block) instead of transmitting the entire corresponding Media Key Block (or Extension). The protocols described above work exactly the same, except for that substitution.

However, the protocols all have a possible additional response from the Security Provider: the Security Provider can respond that it does not currently have the corresponding Media Key Block in its cache. In that case, the protocols proceed as illustrated in Figure 4.

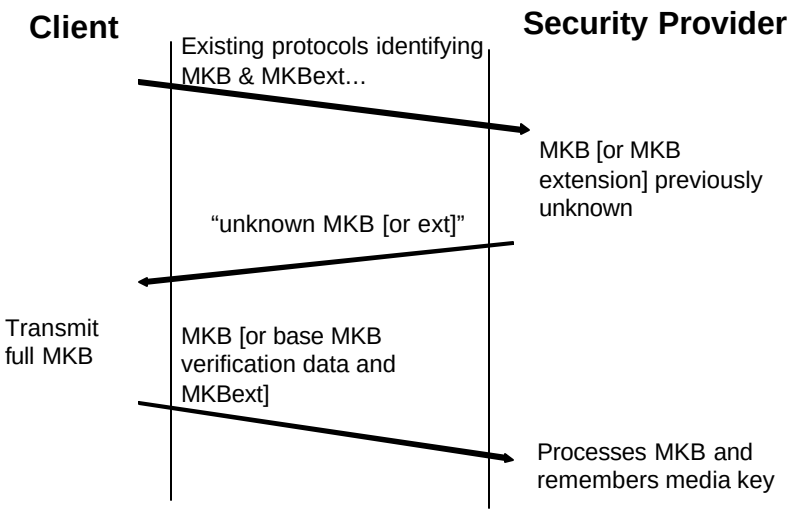

Figure 4 – Client-to-Security Provider Protocol when an MKB is not Known

In this case, the Security Provider responds with a message that indicates that either the Media Key Block or the Media Key Block Extension is unknown to it. The Client shall read the missing data from the media and transmit it. The Client then re-initiates the original protocol, which should now succeed. In the unlikely case that both the Media Key Block and the Media Key Block Extension are unknown to the Security Provider, the Security Provider shall first ask for the Media Key Block, and then, when the Client retries the original request, ask for the Media Key Block Extension. The Client continues to retry the original request, providing Media Key Blocks and Media Key Block Extension as needed, until it successfully obtains the encrypted title key.

In the case that the Client is asked to transfer the Media Key Block Extension, it shall also transmit the verification data of the base Media Key Block, so that the Security Provider knows how to correctly process the Media Key Block Extension.

The Security Provider must treat Media Keys as highly confidential data as described in their license agreement.

3.5 The Security Provider

The function of a Security Provider is to transform Title Keys so that they are bound to particular pieces of media. The Security Provider shall obtain title keys based on its relationships with content owners. The Security Provider shall *not* obtain title keys from existing pieces of media to copy them to new pieces of media. The Security Provider must not act as a copying center, copying content from one piece of media to another, *even if the copy control information on the source media permits a copy*.

However, the Security Provider may transform title keys within a single piece of media, as part of updating a Media Key Block Extension, as described in section 3.4.1 above.

3.6 Device Authentication

This specification does not require the Client to have CPRM device keys. However, certain CPRM media types require device keys or other secrets to read Media Key Blocks and/or write encrypted Title Keys, as part of device authentication. If the Client possesses secrets for this purpose, it shall be considered a CPRM recording device for the purpose of compliance and robustness rules. Alternatively, the Security Provider may possess secrets for this purpose and perform the device authentication through a more complex protocol with the Client. This is implementation-specific. For example, for DVD drive authentication, the Security Provider could (through the client) perform the CSS-based authentication necessary to read the Media Key Block.

This page intentionally left blank.

Appendix A
Example HTTP Protocol

This concrete protocol syntax is presented for informational purposes only. Other syntaxes, including other HTTP syntaxes, would be permitted, as long as they follow the generic protocol described in Chapter 3 of this specification. This example assumes that the Security Provider is caching Media Key Blocks, as explained in section 3.4.2.

A.1 Getting the Ticket

In many cases, getting a ticket involves a financial transaction. However, in some cases (e.g., a content subscription service) the ticket may be delivered for the asking, assuming the user had proven he was a valid subscriber. In such cases, assuming the user has previously identified himself with a userid and password, the following HTTP Get request would deliver a ticket:

`https://`*`url`*`?com=ticket&content=`*`xxxxxxxx`*`&mediaid=`*`yyyyyyyyyyyyyyyy`*

The *url* locates the Service Provider. The "com=ticket" indicates that this is a "get ticket" request. The content ID is xxxxxxx and the Media ID is yyyyyyyyyyyyyyyy, both in readable hexadecimal (i.e., they are the ASCII characters 0-9 and a-f or A-F).

The response is in MIME type `X-application-CPRM-download`, and has the following format:

SUCCESS

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0100	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to End	Ticket_Data	Ticket Data	ticket size

FAILURE

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0101	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to 7	Error_Code	error code	4 bytes

A.2 Getting the Content

The content is identified by its URL, which may have any form. The following example form, however, is consistent with the other URLs in this appendix:

```
http://url?com=content&content=abcdef[&file=yyyy
```

The *url* locates the Service Provider. The "com=content" indicates that this is a "get content" request. The "content=" keyword identifies the content with alphanumeric characters. If the content is divided into several files for the convenience of download, the particular file is identified by the string *yyyy*.

The response is in MIME type `X-application-CPRM-download`, and has the following format:

SUCCESS:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0200	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to End	Content_Data	content data	content size

FAILURE:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0201	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to 7	Error_Code	error code	4 bytes

A.3 Getting the Encrypted Title Key

In this example, the Client transmits the ticket in binary form to the Security Provider with a HTTP Post, in a URL of the following form:

```
http://url?com=key&mkb=xxxxxxxxxxxxxxx  OR
http://url?com=key&mkbext=xxxxxxxxxxxxxxx
```

The string *url* locates the Security Provider. The "com=key" indicates that this is a "bind key" request (command). The value *xxxxxxxxxxxxx* is readable hexadecimal. It is the 8-byte verification string from the Verification Record in the Media Key Block (or Media Key Block Extension).

The Security Provider responds to this post in MIME type `X-application-CPRM-download`, in the following format:

SUCCESS:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0300	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to End	TKUR_Data	Title key and Usage Rule. The format for this data is defined in the different CPRM books. This may include an appended MKB extension if the Security Provider wishes to update the MKB extension.	The size of the title key and Usage Rule

FAILURE:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0301	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to 7	Error_Code	error code	4 bytes

A.4 Sending the Media Key Block

In the case that the Security Provider responds with an error indicating that the MKB is not cached, the Client initiates another HTTP Post with the Security Provider, with the requested MKB in binary form in the post data. The URL is the following:

`http://url?com=mkb[&mkb=xxxxxxxxxxxxxxxx`

The “mkb=” argument is used if and only if the MKB is an MKB extension. In that case, the “mkb=” argument is readable hexadecimal and identifies the base MKB on the media by its verification data.

The responses are in MIME type `X-application-CPRM-download`, in the following format:

SUCCESS:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0400	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte

FAILURE:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0401	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to 7	Error_Code	Error Code	4 bytes

A.5 Transforming the Encrypted Title Keys

If the Client wishes the Security Provider to update existing encrypted title keys after the Security Provider has sent a new Media Key Block Extension, the Client sends the binary encrypted title key data concatenated into a single HTTP Post, with the following URL:

`http://url?com=update&mediaID=`*xxxxxxxxxxxxxxxx*`&oldMkb=`*yyyyyyyyyyyyyyyy*`&newMKB=`*zzzzzzzzzzzzzzzzz*

The values *xx…xx*, *yy…yy*, and *zz…zz* are readable hexadecimal. The responses are in MIME type `X-application-CPRM-download`, in the following format:

SUCCESS:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0500	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to end	TKUR data	Re-encrypted title keys	Depending on the size of the request

FAILURE:

Bytes	Field Name	Contents	Number of bytes
0 to 1	Res_Type	0x0501	2 bytes
2	Generation	0x00	1 byte
3	Reserved	0x00	1 byte
4 to 7	Error_Code	Error Code	4 bytes

:

A.6 Ticket Format

The ticket is in XML form. Binary values are shown as readable hexadecimal. The ticket version is decimal.

For this example syntax, all keywords below must be present, and in the order specified. The "<content id>" tag may denote either a prepared title key or a title ID, depending on what technique the Security Provider uses. Additional tags may be used in certain systems; they must follow the other required tags but precede the <signature> tag. The <signature> line contains the SHA/DSA signature on all previous characters, including line feed and/or carriage return characters, up to, but not including, the string beginning "<signature>".

```
<ticket>
       <ticket_version>0.0</ticket_verison>
       <ticket_id>0DF3224A</ticket_id>
       <content_id>030045A3DF</content_id>
       <ticket_issuer_id>0435</ticket_issuer_id>
       <media_id>04556DF3E2</media_id>
       <signature>0EDA204FC2410</signature>

</ticket>
```

Figure 5 – Format of the HTTP Ticket (Informative)

A.7 Error Codes

Code	Name	Content
0x00000001	Unauthorized	Invalid ticket.
0x00000002	Unknown Key	The Security Provider does not have the requested title key.
0x00000003	Unknown MKB	The Security Provider does not cache the MKB currently
0x00000004	Unknown MKBEx	The Security Provider does not cache the MKB extension currently.