Lightweight Content Protection (LCP) 1.0

Status: Draft
Editors: Hadrien Gardeur (Feedbooks), James Pritchett (Learning Ally)
Authors: Jean-Philippe Bougie (De Marque)

Table of Contents

1. Overview

1.1 Purpose and Scope

This section is informative

In order to deliver a Publication to a User with specific rights and restrictions, a Content Provider may want to encrypt the Publication's Resources and associate them to a specific license.

This specification, Lightweight Content Protection (LCP) 1.0, defines a standard License Document and an Encryption Profile for encrypting resources in EPUB 3 Publications, expressing rights and restrictions, and for securely delivering decryption keys to Reading Systems via licenses for specific Users.

LCP also defines a simple passphrase-based authentication method for Reading Systems to access encrypted resources and verify a license.

1.2. Terminology

EPUB terms

This specification adopts terms defined in the EPUB 3 family of specifications. Important terms used include:

Publication
A logical document entity consisting of a set of interrelated resources and packaged in an EPUB Container, as defined by the EPUB 3 specifications.

Publication Resource (or Resource)
A resource that contains content or instructions that contribute to the logic and rendering of the EPUB Publication. In the absence of this resource, the Publication might not render as intended by the Author. Examples of Publication Resources include the Package Document, EPUB Content Documents, EPUB Style Sheets, audio, video, images, embedded fonts and scripts. With the exception of the Package Document itself, Publication Resources must be listed in the manifest element from [Publications] and must be bundled in the EPUB container file unless specified otherwise in Publication Resource Locations from [Publications].

Examples of resources that are not Publication Resources include those identified by the Package Document link element from [Publications] and those identified in outbound hyperlinks that resolve outside the EPUB Container (e.g., referenced from an [HTML5] a element href attribute).

Package Document
A Publication Resource carrying bibliographical and structural metadata about the EPUB Publication, as defined in Package Documents from [Publications].

EPUB Container (or Container)
The ZIP-based packaging and distribution format for EPUB Publications defined in [OCF].

User
An individual that consumes an EPUB Publication using an EPUB Reading System.

EPUB Reading System (or Reading System)
A system that processes EPUB Publications for presentation to a User in a manner conformant with the EPUB 3 specifications.

LCP terms

The following terms are defined by this specification:

Protected Publication

A Publication that has been protected according to this specification.

License Document

Document that contains references to the various keys, links to related external resources, rights and restrictions that are applied to the Protected Publication, and user information. When embedded in a Protected Publication this file is always located at `META-INF/license.lcpl`

Content Key

Symmetric key used to encrypt the resources of the Protected Publication. In the License Document, this Content Key will be encrypted using the User Key.

User Passphrase

A string of text entered by the user that is used to generate the User Key.

User Key

A hash of the User Passphrase that is used to decrypt the Content Key and selected user information fields.

Encryption Profile

A set of encryption algorithms used in a specific Protected Publication and associated Licence Document.

Content Provider (or Provider)

An entity that delivers LCP licenses for Protected Publications to Users.

Provider Certificate

A certificate that is included in the License Document to identify the Content Provider and validate the signature of the License Document.

Root Certificate

A certificate that is embedded in the Reading System in order to confirm that the Provider Certificate is valid.

1.3. LCP Overview and Example

This section is informative

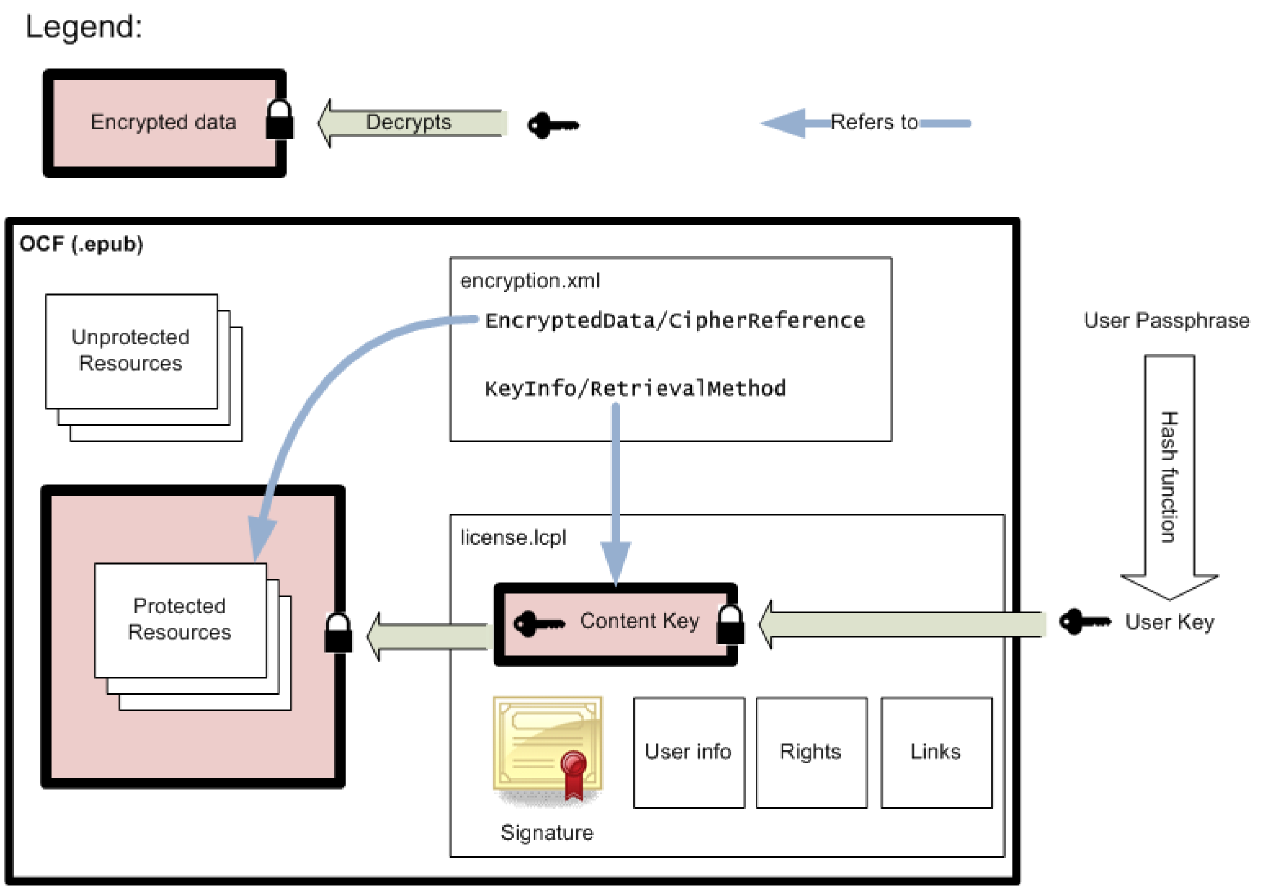
In order to encrypt Resources and expose its Encryption Profile, LCP relies on two different files:

- License Document (`META-INF/license.lcpl`)
- `META-INF/encryption.xml`

Both of these files are contained inside the EPUB Container, although the License Document can be transmitted initially outside the Container. Following the [OCF] specification, `META-INF/encryption.xml` identifies the Publication Resources that are protected and points to the Content Key needed to decrypt them. This Content Key is located inside the License Document and is itself encrypted using the third element of LCP, the User Key. The User Key is generated by taking a hash of a User Passphrase. The User Key is used to decrypt the Content Key, which in turn is used to decrypt the Publication Resources.

The License Document may also contain links to external resources, information identifying the User, and information about what rights are conveyed to the User and which are not. Rights information may include things like the time for which the license is valid, whether the book may be printed or copied, etc. Finally, the License Document always includes a digital signature to prevent modification of any of its components.

Figure 1 shows the relationships among the various components of LCP.



Protecting the Publication

To protect a Publication, the Provider follows these steps:

1. Generate a unique Content Key for the Publication.

2. Store this Content Key for future use in licensing the Publication.
3. Encrypt each protected Resource using that key.
4. Add these protected Resources to the Container, replacing the unprotected versions.
5. Create a `META-INF/encryption.xml` document (as described in section 2.2) that includes an EncryptedData element for each protected Resource that contains:
    a. An `EncryptionMethod` element that lists the algorithm used
    b. A `KeyInfo` element with a `RetrievalMethod` child that points to the Content Key in the License Document
    c. A `CipherData` element that identifies the protected Resource
6. Add `META-INF/encryption.xml` to the Container.

The Publication is now protected (i.e., has become a Protected Publication) and is ready for licensing to one or more Users.

Licensing the Publication

After a User requests a Protected Publication, the following steps are followed by the Provider to license the Protected Publication:

1. Generate the User Key by hashing the User Passphrase (as described in section 4.2). It is assumed that the User and their User Passphrase are already known to the Provider.
2. Encrypt the Content Key for the Protected Publication using the User Key.
3. Create a License Document (`META-INF/license.lcpl`) with the following contents (as described in section 3):
    a. A unique ID for this license
    b. The date the license was issued
    c. The URI that identifies the Content Provider
    d. The encrypted Content Key
    e. Information about the User Key
    f. Links to additional information stored outside of the Protected Publication and License Document (optional)
    g. Information on specific rights being granted to the User (optional)
    h. Information identifying the User (optional). Some of the fields in this section may be encrypted using the User Key.
4. Generate a digital signature for the License Document data and add to the License Document.

There are then two different methods to deliver the License Document and Protected Publication to the User:

1. **License Document included inside Protected Publication:** The Provider adds the License Document to the protected Protected Publication's Container and delivers this to the User.
2. **License Document delivered separately:** The Provider includes a link from the License Document to the Protected Publication, and then delivers just the License Document to the User. The Reading System processing the License Document will

retrieve the Protected Publication and add the License Document to the Container of this Protected Publication.

Whichever method is used, the Reading System will be presented with an EPUB Container that includes the Protected Publication and the License Document.

Reading the Publication

In order to decrypt and render a Protected Publication, the User's Reading System will follow these steps:

1. Verify the signature for the License Document.
2. Get the User Key (if already stored) or generate it by hashing the User Passphrase.
3. Decrypt the Content Key using the User Key.
4. Decrypt the protected Resources using the Content Key.

1.4 Conformance Statements

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC2119].

All sections of this specification are normative except where identified by the informative status label "This section is informative". The application of informative status to sections and appendices applies to all child content and subsections they may contain.

All examples in this specification are informative.

2. Publication

2.1. Introduction

This section is informative

LCP follows the [OCF] specification to identify encrypted Resources within the EPUB Container. Specifically, LCP uses `META-INF/encryption.xml` to indicate which Resources listed in the Package Document are encrypted and to reference the Content Key.

2.1. Encrypted Resources

In order to make sure that Protected Publications remain identifiable by Reading Systems, a certain number of Resources are prohibited from being encrypted.

LCP inherits the following list of files prohibited from being encrypted from the [OCF] specification:

- `mimetype`
- `META-INF/container.xml`
- `META-INF/encryption.xml`
- `META-INF/manifest.xml`
- `META-INF/metadata.xml`
- `META-INF/rights.xml`
- `META-INF/signatures.xml`
- EPUB `rootfiles` (the Package Document for any rendition)

In addition, this specification defines that the following files MUST NOT be encrypted:

- `META-INF/license.lcpl`
- Navigation Documents referenced in any Package Document from the Publication
- NCX documents referenced in any Package Document from the Publication
- Cover images (all Publication Resources listed in the Publication manifest with the "cover-image" property).

2.2. Using `META-INF/encryption.xml` for LCP

As defined in the [OCF] specification, all encrypted Publication Resources MUST be identified in the well-known file `META-INF/encryption.xml` using [XML-ENC].

In Publications protected using LCP, there are additional requirements for identifying the key used to encrypt these Resources (the LCP Content Key):

1. The `ds:KeyInfo` element MUST point to the Content Key using the `ds:RetrievalMethod` element.
2. The `URI` attribute of `ds:RetrievalMethod` MUST use a value of `"license.lcpl#/encryption/content_key"` to point to the encrypted Content Key stored in the License Document. This URI follows the [JSON Pointer] specification.
3. The `Type` attribute MUST use a value of `"http://readium.org/2014/01/lcp#EncryptedContentKey"` to identify the target of the URI as an encrypted Content Key.

In the following example, adapted from [OCF], an image.jpeg resource is encrypted using AES with the Content Key.

```
<encryption
    xmlns ="urn:oasis:names:tc:opendocument:xmlns:container"
    xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

    <enc:EncryptedData Id="ED1">
        <enc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
        <ds:KeyInfo>
            <ds:RetrievalMethod URI="license.lcpl#/encryption/content_key"
                Type="http://readium.org/2014/01/lcp#EncryptedContentKey"/>
        </ds:KeyInfo>
        <enc:CipherData>
            <enc:CipherReference URI="image.jpeg"/>
        </enc:CipherData>
    </enc:EncryptedData>

</encryption>
```

3. License Document

3.1. Introduction

This section is informative

While `META-INF/encryption.xml` describes how the Resources are encrypted and where the encrypted Content Key is located, all of the other relevant information for LCP is stored in the License Document.

This specification defines the License Document's syntax, location in the Container, media type, file extension and processing model.

3.2. Content Conformance

A License Document MUST meet all of the following criteria:

Document properties

- It MUST meet the conformance constraints for JSON documents as defined in [JSON].
- It MUST be encoded using UTF-8.

File properties

- Its filename MUST use the file extension `.lcpl`.

- Its MIME media type is `application/vnd.readium.lcp.license-1.0+json`
- Its location in the Container MUST be `META-INF/license.lcpl`

3.3. Core License Information

The License Document MUST contain the following name/value pairs:

Name	Value	Format/data type
`id`	A unique identifier for this license	String
`issued`	Date when the license was first issued	ISO 8601
`provider`	Unique identifier for the Provider	URI

In addition, the License Document MAY contain the following name/value pair:

Name	Value	Format/data type
`updated`	Date when the license was last updated	ISO 8601

3.4. Transmitting keys: The `encryption` object

In addition to Core License information, the License Document MUST contain an `encryption` object with the following name/value pair:

Name	Value	Format/data type
`profile`	Identifies the Encryption Profile used by this LCP-protected Publication	URI

The `encryption` object MUST also contain the following two objects: `content_key` and `user_key`.

The `encryption/content_key` object contains the Content Key (encrypted using the User Key) used to encrypt the Publication Resources. It MUST contain the following name/value pairs:

Name	Value	Format/data type
`encrypted_value`	Encrypted Content Key	Base 64 encoded octet sequence
`algorithm`	Algorithm used to encrypt the Content Key, identified using the URIs defined in [XML-ENC]. This MUST match the Content Key encryption algorithm named in the Encryption Profile identified in `encryption/profile`.	URI

The `encryption/user_key` object contains information regarding the User Key used to encrypt the Content Key. It MUST contain the following name/value pairs:

Name	Value	Format/data type
`text_hint`	A hint to be displayed to the User to help them remember the User Passphrase	String
`algorithm`	Algorithm used to generate the User Key from the User Passphrase, identified using the URIs defined in [XML-ENC]. This MUST match the User Key hash algorithm named in the Encryption Profile identified in `encryption/profile`.	URI
`key_check`	The value of the License Document's `id` field, encrypted using the User Key and the same algorithm identified for Content Key encryption in `encryption/content_key/algorithm`. This is used to verify that the Reading System has the correct User Key.	Base 64 encoded octet sequence

This example shows the encryption information for a License Document that uses the base Encryption Profile for LCP 1.0.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "issued": "2013-11-04T01:08:15+01:00",
    "updated": "2014-02-21T09:44:17+01:00"
    "provider": "http://www.imaginaryebookretailer.com",
    "encryption": {
        "profile": "http://readium.org/lcp/profile-1.0",
        "content_key": {
            "encrypted_value":
"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1
X0aA3z55boQ==",
            "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc"
        },
        "user_key": {
            "text_hint": "Enter your email address",
            "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
           "key_check":
"jJEjUDipHK3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww"
        }
    },
    "links": …,
    "signature": …
}
```

3.5. Pointing to external resources: the `links` object

A License Document MUST also contain a `links` object. This is used to associate the License Document with resources that are not locally available. Each object nested in `links` is a link where the object name is a link relation and its content is the link URI and an optional set of other link properties.

This specification introduces two link relations for the `links` object:

Relation	**Semantics**	**Required?**
`hint`	Location where a Reading System can redirect a User looking for additional information about the User Passphrase.	Yes
`publication`	Location where the Publication associated to the License Document can be downloaded	Yes, if the License Document is not distributed inside the Publication's Container.

In addition to these two link relations, this specification introduces the LCP Link Relations Registry. All official link relations used in the License Document and declared in official LCP specification documents MUST be referenced in the registry.

Link relations MAY also be extended for vendor-specific applications. Such links MUST use a URL instead of a string to identify their link relations.

Each object nested within the `links` object contains the following name/value pairs:

Name	**Value**	**Format/data type**	**Required?**
`href`	Link location	URI	Yes
`title`	Title of the link	String	No
`type`	Expected MIME media type value for the external resources	MIME media type	No, but highly recommended
`length`	Content length in octets	Integer	No
`hash`	SHA-256 hash of the resource	Base 64 encoded octet sequence	No

In this example, the License Document points to a publication, contains the location of a hint about its User Passphrase and uses an extension to provide an authentication service.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "provider": "http://www.imaginaryebookretailer.com",
    "issued": "2013-11-04T01:08:15+01:00",
    "updated": "2013-11-04T01:08:15+01:00",
    "encryption": … ,
    "links": {
        "publication": {
            "href": "http://www.example.com/file.epub",
            "type": "application/epub+zip",
            "length": "",
            "hash": ""
        },
        "hint": {
            "href": "http://www.example.com/passphraseHint?user_id=1234",
            "type": "text/html"
        },
        "http://mylcpextension.com/authentication": {
            "href": "http://www.example.com/authenticateMe",
            "title": "Authentication service",
            "type": "application/vnd.myextension.authentication+json"
        }
    },
    "signature": …
}
```

3.6. Identifying rights and restrictions: the `rights` object

The License Document MAY also express a series of rights using the `rights` object. The `rights` object MAY include the following fields:

Name	**Value**	**Format/data type**	**Default**
`print`	Maximum number of pages that can be printed at a time	Integer	Unlimited
`copy`	Maximum number of characters that can be copied to the clipboard at a time	Integer	Unlimited
`tts`	Indication whether the Publication can be read in audio via text-to-speech	true/false	true
`start`	Date when the license begins	ISO 8601 date	None (perpetual license)
`end`	Date when the license ends	ISO 8601 date	None (perpetual license)
`[Extension URI]`	[Defined by implementor]	[Defined by implementor]	[Defined by implementor]

All name/value pairs using an integer as their data type (print and copy for this specification) MUST use the value with the lowest number of digits possible (ie "9" instead of "09).

Define what a page means outside of pre-paginated context (1024 characters ?).
Copy only covers ability to copy to clipboard. Limited to text.

In addition to these rights information, this specification introduces the LCP Rights Registry. All official rights information used in the License Document and declared in official LCP specification documents MUST be referenced in the registry.

The `rights` object may be extended to include any number of implementor-specific rights. Each extension right MUST be identified using a URI controlled by the implementor.

In this example, the License Document grants the following rights: not allowed to print at all, copy up to 2048 characters at a time of the book, allowed to use text to speech, allowed to edit the book, and the license has an expiration date.
There is also a vendor extension granting the right to tweet parts of this book.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "provider": "http://www.imaginaryebookretailer.com",
    "issued": "2013-11-04T01:08:15+01:00",
    "updated": "2013-11-04T01:08:15+01:00",
    "encryption": … ,
    "links": … ,
    "rights": {
        "print": 0,
        "copy": 2048,
        "tts": true,
        "start": "2013-11-04T01:08:15+01:00",
        "end": "2013-11-25T01:08:15+01:00",
        "http://www.imaginaryebookretailer.com/lcp/rights/tweet": true
    },
    "signature": …
}
```

3.7. Identifying the user: the `user` object

The License Document MAY embed information about the user using the `user` object. The `user` object includes the following fields:

Name	Value	Format/data type	Required?
`id`	Unique identifier for the User at a specific Provider	String	No, but highly recommended
`email`	The User's e-mail address	String	No
`name`	The User's name	String	No
`encrypted`	A list of which `user` object values are encrypted in this License Document	Array of one or more strings matching the above names or an extension	Yes, if encryption is used for any field
`[Extension URI]`	[Defined by implementor]	[Defined by implementor]	[Defined by implementor]

In addition to these user information, this specification introduces the LCP User Fields Registry. All official user fields used in the License Document and declared in official LCP specification documents MUST be referenced in the registry.

As with rights, The `user` object may be extended to include any number of implementor-specific fields. Each extension field MUST be identified by a URI controlled by the implementor.

To protect private User data, any of these fields MAY be encrypted, except for the `provider` and `encrypted` fields, which MUST remain in plain text. If encrypted, the field values MUST be encrypted using the User Key and the same encryption algorithm identified in the `encryption/content_key` object. The names of all encrypted fields MUST be listed in the `encrypted` array.

In the following example, an identifier, a provider and an email are provided. There is also an extension to indicate the user's preferred language. The email is encrypted.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "provider": "http://www.imaginaryebookretailer.com",
    "issued": "2013-11-04T01:08:15+01:00",
    "updated": "2013-11-04T01:08:15+01:00",
    "encryption": … ,
    "links": … ,
    "rights": … ,
    "user": {
        "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597",
        "email":
"EnCt2b8c6d2afd94ae4ed201b27049d8ce1afe31a90ceb8c6d2afd94ae4ed201b2704RjkaXR
veAAarHwdlID1KCIwEmS",
        "encrypted": ["email"],
        "http://www.imaginaryebookretailer.com/lcp/user/language": "tlh"
    },
    "signature": …
}
```

3.8. Signing the license: the `signature` object

As described in 5. Signature and Public Key Infrastructure, the License Document includes a digital signature to validate that it has not been altered. The License Document MUST include information about the signature using the `signature` object. The `signature` object MUST include the following fields:

Name	Value	Format/data type
`algorithm`	Algorithm used to calculate the signature, identified using the URIs given in [XML-SIG]. This MUST match the signature algorithm named in the	URI

	Encryption Profile identified in `encryption/profile`.	
`certificate`	The Provider Certificate: an X509 certificate used by the Content Provider	Base 64 encoded DER certificate
`value`	Value of the signature	Base 64 encoded octet sequence

For more information on how the signature and the certificate should be calculated, encoded and processed, see 5. Signature and Public Key Infrastructure.

This example shows the License Document signature.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "provider": "http://www.imaginaryebookretailer.com",
    "issued": "2013-11-04T01:08:15+01:00",
    "updated": "2013-11-04T01:08:15+01:00",
    "encryption": … ,
    "links": … ,
    "rights": … ,
    "user": … ,
    "signature": {
        "algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
        "certificate":
"MIIDEjCCAfoCCQDwMOjkYYOjPjANBgkqhkiG9w0BAQUFADBLMQswCQYDVQQGEwJVUzETMBEGA1U
ECBMKQ2FsaWZvcm5pYTETMBEGA1UEBxMKRXZlcnl3aGVyZTESMBAGA1UEAxMJbG9jYWxob3N0MB4
XDTE0MDEwMjIxMjYxNloXDTE1MDEwMjIxMjYxNlowSzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkN
hbGlmb3JuaWExEzARBgNVBAcTCkV2ZXJ5d2hlcmUxEjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQY
JKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOpCRECG7icpf0H37kuAM7s42oqggBoikoTpo5yapy+
s5eFSp8HSqwhIYgZ4SghNLkj3e652SALav7chyZ2vWvitZycY+aq50n5UTTxDvdwsC5ZNeTycuzV
WZALKGhV7VUPEhtWZNm0gruntronNa8l2WS0aF7P5SbhJ65SDQGprFFaYOSyN6550P3kqaAO7tDd
dcA1cmuIIDRf8tOIIeMkBFk1Qf+lh+3uRP2wztOTECSMRxX/hIkCe5DRFDK2MuDUyc/iY8IbY0hM
FFGw5J7MWOwZLBOaZHX+Lf5lOYByPbMH78O0dda6T+tLYAVzsmJdHJFtaRguCaJVtSXKQUAMCAwE
AATANBgkqhkiG9w0BAQUFAAOCAQEAi9HIM+FMfqXsRUY0rGxLlw403f3YtAG/ohzt5i8DKiKUG3Y
AnwRbL/VzXLZaHru7XBC40wmKefKqoA0RHyNEddXgtY/aXzOlfTvp+xirop+D4DwJIbaj8/wHKWY
GBucA/VgGY7JeSYYTUSuz2RoYtjPNRELIXN8A+D+nkJ3dxdFQ6jFfVfahN3nCIgRqRIOt1KaNI39
CShccCaWJ5DeSASLXLPcEjrTi/pyDzC4kLF0VjHYlKT7lq5RkMO6GeC+7YFvJtAyssM2nqunA2lU
gyQHb1q4Ih/dcYOACubtBwW0ITpHz8N7eO+r1dtH/BF4yxeWl6p5kGLvuPXNU21ThgA==",
        "value":
"q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ
/hnt8+GWrEx1QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8
m/ZlMxDcS3BZnVhSEKzUNQn1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA4lM8XNfuk
EYQJnivqhqMLOGM33RnS5nZKrPPK/c2F/vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsn
MLAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw=="
    }
}
```

4. User Key

4.1. Introduction

This section is informative

In order for any symmetrically-encrypted message to work, there must be an agreed-upon key shared by the sender and receiver. In LCP, this is the User Key. The Provider must have access to the User Key in order to secure the Content Key within the License Document. The Reading System must also have the User Key in order to decrypt the Content Key in the License Document.

LCP uses a passphrase model for sharing the User Key: in a simple implementation, when the Reading System receives a new License Document, it prompts the User for a passphrase to access the Content Key. LCP defines the User Key as a hash of this User Passphrase. This passphrase can be anything at all: a User-defined password, a Provider-defined password, an e-mail address, a library card number, etc.

4.2. Calculating the User Key

The User Passphrase MUST be a UTF-8 encoded string. There are no restrictions on the length or content of the User Passphrase. There are no requirements for how it is to be created.

The User Key is a hash of the User Passphrase, using the hash algorithm provided in the `encryption/user_key` object in the License Document. Processing of any kind, including whitespace escaping or normalization, SHALL NOT be done on the User Passphrase before hashing.

4.3. Hints

In order to facilitate the entry of the passphrase by the User, the LCP License Document supports two ways for the Reading System to provide passphrase prompts or hints to the User:

1. The `user_key/text_hint` field is a simple prompt that can be displayed to the User when entering the User Passphrase (e.g., “Enter your Imaginary Book Retailer password”).
2. The `links/hint` object points to a location that provides passphrase hints or other assistance.

The content of `user_key/text_hint` and of the resource pointed to by `links/hint` SHOULD be human-readable, directed to the User, and SHOULD help the User enter their passphrase.

4.4. Best Practices for the User Key & User Passphrase

In order to simplify the process for accessing Protected Publications, the Provider SHOULD use the same User Key for multiple Protected Publications licensed to the same User as much as possible.

The security of LCP depends in large part on the security of the User Key and User Passphrase. Therefore, special care should be taken to secure these throughout the licensing workflow:

1. Passphrases SHOULD be sufficiently complex to prevent brute-force attacks.
2. Passphrases MUST NOT be transmitted in plaintext.
3. If the Provider needs to share User Key information among multiple systems, they SHOULD transmit User Keys and not User Passphrases, and SHOULD transmit this information over secure channels.
4. User Passphrases MUST NOT be stored by the Reading System; instead it SHOULD only store User Keys.

5. Signature and Public Key Infrastructure

5.1. Introduction

This section is informative

Given the importance of the precise expression of various objects in the License Document, it is critical that the Reading System be able to verify that the content of the License Document is authentic and has not been altered. This is done via a digital signature that is verified via a Public Key Infrastructure as defined in [X509].

The steps required of the Provider to sign the License Document are:

1. The contents of the License Document (minus the signature object) are put in a canonical form: alphabetized with non-significant whitespace removed (see Section 5.3)
2. The signature value for the canonical form of the License Document is calculated following the algorithm identified in the Encryption Profile. This typically will involve taking a hash of the data and encrypting it using the private key.
3. The signature value and the Provider Certificate are added to the License Document in the `signature` object, as described in Section 3.8.

The steps required of the Reading System to validate the signature are:

1. Obtain the Root Certificate from the Readium Foundation. This certificate is embedded in the Reading System to validate all Provider Certificates.
4. Using this Root Certificate, validate the Provider Certificate that is included in the `signature` object. This prevents someone who is not the Provider from forging License Documents.
5. Strip the `signature` object from the License Document.
2. Put the remainder of the License Document contents into canonical form.
3. Calculate the hash of the canonical License Document data using the signature algorithm identified in the Encryption Profile.
4. Decrypt the signature value given in the signature object using the public key given in the Provider Certificate and confirm that it matches the calculated hash. This confirms that the contents of the License Document have not been altered in transit.

Because the Provider Certificate is included in the License Document and the Root Certificate is embedded in the Reading System, the entire validation process can take place without any Internet connectivity.

To make sure that the Provider Certificate has not been revoked, the Reading System also checks a Certificate Revocation List maintained by the Readium Foundation. In order to facilitate offline reading, the Reading System does not need to check the revocation list every time it processes a License Document. It is only necessary that it updates the list regularly when an Internet connection is available (e.g., every time it downloads a new License or book).

Calculating a signature using the RSA set of algorithms is done on a byte stream, which is unique, while the License Document is a JSON document where multiple representations might lead to the same structure. Thus, to ensure a stable signature between the Reading System and the Content Provider, some transformations must be applied prior to signing the Document and verifying the signature.

5.2. Certificates

5.2.1 Provider Certificates

Content Providers MUST have a Certificate in the [X509] v3 format issued and signed by the Readium Foundation using the Root Certificate: this is referred to here as the Provider Certificate. The issuer of the Provider Certificate MUST be the Readium Foundation and the issuer MUST match the subject of the Root Certificate. The subject of the Provider Certificate SHOULD represent the Content Provider.

Content Providers MUST distribute their Provider Certificate in any License Document they issue in the `signature/certificate` field. They also MUST use the key from their Provider Certificate's subject public key info to sign the License Document. For a License Document to be considered valid, the Provider Certificate MUST have been valid at the time the License Document was issued (as indicated by the `issued` field), and the Provider Certificate MUST NOT have been revoked.

5.2.1 Root Certificate

Reading Systems MUST obtain the Root Certificate in the [X509] v3 format from the Readium Foundation, and SHOULD keep it up to date. It MUST be embedded in the Reading System for offline use.

5.3. Canonical form of the License Document

The canonical form of the License Document is used when calculating and validating the signature. To create the canonical form of the License Document, the following serialization rules must be followed:

1. Since it is product of the calculation, the `signature` object of the License Document MUST be removed.
2. All object members (name/value pairs) of the License Document MUST be sorted in lexicographical order of their names according to their representation in UTF-8 (United Character Set code point value). Note that this rule is recursive, so that members are sorted at all levels of object nesting.
3. Within arrays, the order of elements MUST NOT be altered.
4. Numbers MUST NOT include leading or trailing zeroes. Numbers that include a fraction part (non-integers) MUST be expressed as a number, fraction, and exponent (normalized scientific notation) using an upper-case "E".
5. Strings MUST use escaping only for those characters for which it is required by [JSON]: backslash (\), double-quotation mark ("), and control characters (U+0000 through U+001F). When escaping control characters, the hexadecimal digits MUST be upper case.
6. Non-significant whitespace (as defined in [JSON]) MUST be removed. Whitespace found within strings MUST be kept.

5.3.1. Example

This section is informative

For this example, we'll use a License Document which contains a link (hint) and basic user information.

```
{
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "provider": "http://www.imaginaryebookretailer.com",
    "date": "2013-11-04T01:08:15+01:00",
    "encryption": {
        "profile": "http://readium.org/lcp/profile-1.0",
        "content_key": {
            "encrypted_value":
"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1
X0aA3z55boQ==",
            "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc"
        },
        "user_key": {
            "text_hint": "Enter your email address",
            "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
        }
    },
    "links": {
        "hint": {
            "href": "http://www.imaginaryebookretailer.com/lcp/hint",
            "type": "text/html"
        }
    },
    "user": {
        "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"
    }
}
```

First of all, let's sort this document. Every [JSON] object needs to be sorted:

```
{
    "date": "2013-11-04T01:08:15+01:00",
    "encryption": {
        "content_key": {
            "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc",
            "encrypted_value":
"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1
X0aA3z55boQ=="
        },
        "profile": "http://readium.org/lcp/profile-1.0",
        "user_key": {
            "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
            "text_hint": "Enter your email address"
        }
    },
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "links": {
        "hint": {
            "href": "http://www.imaginaryebookretailer.com/lcp/hint",
            "type": "text/html"
        }
    },
    "user": {
        "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597",
        "provider": "http://www.imaginaryebookretailer.com"
    }
}
```

Now that our document is sorted, we can strip all whitespaces and end of lines:

```
{"date":"2013-11-04T01:08:15+01:00","encryption":{"content_key":{"algorithm"
:"http://www.w3.org/2001/04/xmlenc#aes256-cbc","encrypted_value":"/k8RpXqf4E
2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1X0aA3z55boQ
=="},"profile":"http://readium.org/lcp/profile-1.0","user_key":{"algorithm":
"http://www.w3.org/2001/04/xmlenc#sha256","text_hint":"Enter your email
address"}},"id":"ef15e740-697f-11e3-949a-0800200c9a66","links":{"hint":{"hre
f":"http://www.imaginaryebookretailer.com/lcp/hint","type":"text/html"}},"us
er":{"id":"d9f298a7-7f34-49e7-8aae-4378ecb1d597","provider":"http://www.imag
inaryebookretailer.com"}}
```

5.4. Generating the signature

In order to sign a License Document, the Content Provider MUST go through the following steps in order:

1. The Content Provider MUST create the canonical form of the License Document following the rules given in 5.3. Canonical form of the License Document.
2. The Content Provider MUST calculate the signature using this canonical form of the License Document and using the algorithm described in the `algorithm` field with the private key of the Provider Certificate.
3. The signature MUST be embedded in the `value` field of the `signature` object using Base 64 encoding.
4. The Provider Certificate used to validate the signature MUST be inserted in the `certificate` field. It MUST use the DER notation and be encoded using Base 64.

5.4.1. Example

This section is informative

Given the License Document in its canonical form

```
{"date":"2013-11-04T01:08:15+01:00","encryption":{"content_key":{"algorithm"
:"http://www.w3.org/2001/04/xmlenc#aes256-cbc","encrypted_value":"/k8RpXqf4E
2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1X0aA3z55boQ
=="},"profile":"http://readium.org/lcp/profile-1.0","user_key":{"algorithm":
"http://www.w3.org/2001/04/xmlenc#sha256","text_hint":"Enter your email
address"}},"id":"ef15e740-697f-11e3-949a-0800200c9a66","links":{"hint":{"hre
f":"http://www.imaginaryebookretailer.com/lcp/hint","type":"text/html"}},"us
er":{"id":"d9f298a7-7f34-49e7-8aae-4378ecb1d597","provider":"http://www.imag
inaryebookretailer.com"}}
```

Using the LCP Profile 1.0 signature algorithm, `http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`, a Content Provider must first hash the License Document using SHA256, giving the following byte sequence, represented here in hexadecimal:

```
ce3fd2220fc26c89ed2190fd27af39c4016b623dd2bd5e5a3562dc0795769c92
```

This SHA-256 form must then be signed using the Content Provider's Private Key, giving the following result, in Base 64:

```
q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ/
hnt8+GWrEx1QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8m
/ZlMxDcS3BZnVhSEKzUNQn1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA4lM8XNfukE
YQJnivqhqMLOGM33RnS5nZKrPPK/c2F/vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsnM
LAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw==
```

With this signature and the certificate, a valid license may be created:

```
{
    "date": "2013-11-04T01:08:15+01:00",
    "encryption": {
        "content_key": {
            "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc",
            "encrypted_value":
"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmCK0nRLyAxs1
X0aA3z55boQ=="
        },
        "profile": "http://readium.org/lcp/profile-1.0",
        "user_key": {
            "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
            "text_hint": "Enter your email address"
        }
    },
    "id": "ef15e740-697f-11e3-949a-0800200c9a66",
    "links": {
        "hint": {
            "href": "http://www.imaginaryebookretailer.com/lcp/hint",
            "type": "text/html"
        }
    },
    "user": {
        "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"
    },
    "signature": {
        "algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
        "certificate":
"MIIDEjCCAfoCCQDwMOjkYYOjPjANBgkqhkiG9w0BAQUFADBLMQswCQYDVQQGEwJVUzETMBEGA1U
ECBMKQ2FsaWZvcm5pYTETMBEGA1UEBxMKRXZlcnl3aGVyZTESMBAGA1UEAxMJbG9jYWxob3N0MB4
XDTE0MDEwMjIxMjYxNloXDTE1MDEwMjIxMjYxNlowSzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkN
hbGlmb3JuaWExEzARBgNVBAcTCkV2ZXJ5d2hlcmUxEjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQY
JKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOpCRECG7icpf0H37kuAM7s42oqggBoikoTpo5yapy+
s5eFSp8HSqwhIYgZ4SghNLkj3e652SALav7chyZ2vWvitZycY+aq50n5UTTxDvdwsC5ZNeTycuzV
WZALKGhV7VUPEhtWZNm0gruntronNa8l2WS0aF7P5SbhJ65SDQGprFFaYOSyN6550P3kqaAO7tDd
dcA1cmuIIDRf8tOIIeMkBFk1Qf+lh+3uRP2wztOTECSMRxX/hIkCe5DRFDK2MuDUyc/iY8IbY0hM
FFGw5J7MWOwZLBOaZHX+Lf5lOYByPbMH78O0dda6T+tLYAVzsmJdHJFtaRguCaJVtSXKQUAMCAwE
AATANBgkqhkiG9w0BAQUFAAOCAQEAi9HIM+FMfqXsRUY0rGxLlw403f3YtAG/ohzt5i8DKiKUG3Y
AnwRbL/VzXLZaHru7XBC40wmKefKqoA0RHyNEddXgtY/aXzOlfTvp+xirop+D4DwJIbaj8/wHKWY
GBucA/VgGY7JeSYYTUSuz2RoYtjPNRELIXN8A+D+nkJ3dxdFQ6jFfVfahN3nCIgRqRIOt1KaNI39
CShccCaWJ5DeSASLXLPcEjrTi/pyDzC4kLF0VjHYlKT7lq5RkMO6GeC+7YFvJtAyssM2nqunA2lU
gyQHb1q4Ih/dcYOACubtBwW0ITpHz8N7eO+r1dtH/BF4yxeWl6p5kGLvuPXNU21ThgA==",
        "value":
"q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ
/hnt8+GWrEx1QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8
m/ZlMxDcS3BZnVhSEKzUNQn1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA4lM8XNfuk
EYQJnivqhqMLOGM33RnS5nZKrPPK/c2F/vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsn
MLAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw=="
    }
}
```

5.5. Verifying the signature

In order to validate a signature, the Reading System MUST go through the following steps in the order:

5.5.1. Validating the signature

5. The Reading System MUST calculate the canonical form of the License Document following the rules as expressed in 5.3. Canonical form of the License Document.
6. It MUST calculate the signature as defined in 5.4. Generating the signature.
7. It MUST verify that the calculated signature value is consistent with the one embedded in the License Document.

5.5.2. Validating the certificate

1. It MUST check that the Certificate was not expired when the License Document was last updated.
2. It MUST validate the presence of the Provider Certificate in the root chain. To do so, it MUST check the signature of the Provider Certificate using the public key of the Root Certificate.
3. It MUST validate that the certificate was not revoked as defined in [X509]. If a network connection is available, it MUST update its certificate revocation list before it checks the validity of the certificate.

6. Encryption profiles

6.1. Introduction

This section is informative

LCP is entirely based on standard encryption algorithms, as defined in [XML-ENC] and [XML-SIG]. In order to maintain maximum flexibility, no specific algorithms are mandated by this specification. Instead, the design of both `encryption.xml` and the License Document allow for the identification of encryption algorithms to be discovered by Reading Systems when presented with a Protected Publication.

In order to simplify this discovery process, the LCP 1.0 specification defines an Encryption Profile, which is the set of encryption algorithms used in a specific Protected Publication and associated Licence Document. Reading Systems that implement the algorithms identified in the Encryption Profile will be able to decrypt Protected Publications encoded using that Encryption Profile. For ease of discovery, the Encryption Profile is identified in the License Document.

This specification defines the LCP 1.0 Encryption Profile, along with a list of associated algorithms extracted from [XML-ENC] or [XML-SIG]. All future official or vendor-specific extensions will also define such an Encryption Profile for easy identification by Reading Systems and publish such profiles in the LCP Encryption Profiles Registry.

6.2. Encryption profile requirements

All Encryption Profiles MUST identify algorithms for the following targets:

1. Publication Resources
2. Content Key and User fields (if encrypted)
3. User Passphrase
4. Signature

All algorithms used in an Encryption Profile MUST be defined in [XML-ENC] or [XML-SIG].

All Encryption Profiles MUST use a URI to identify themselves in `profile` (contained in the `encryption` object of the License Document).

All Encryption Profiles MUST be registered in the LCP Encryption Profiles registry, as explicitly explained in the registry.

6.3. LCP 1.0 Encryption Profile

LCP 1.0 Encryption Profile is officially identified in the `encryption` object of the License Document using the URL `http://readium.org/lcp/profile-1.0` for the `profile` attribute value.

The following algorithms are associated to the LCP 1.0 Encryption Profile:

Encryption target	**Algorithm (name)**	**Algorithm (URI)**	**Identified in**
Publication Resources	AES 256 bits CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc	encryption.xml
Content Key, User fields (if encrypted)	AES 256 bits CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc	License Document
User Passphrase	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256	License Document

Signature	RSA with SHA-256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	License Document

7. Reading System Behavior

7.1. License Document processing

Overall

In processing a License Document, Reading Systems MUST ignore all name/value pairs they do not understand.

Reading Systems MUST NOT store unencrypted versions of the Content Key and/or encrypted user fields.

Validating the License Document

Reading Systems MUST:

1. Validate the syntax and completeness of the License Document
2. Validate the signature as defined in 5.5. Verifying the signature

Acquiring the Publication

When License Documents are delivered independently of the Protected Publication, Reading Systems MUST:

1. Download the Protected Publication at the given URL given in `links/publication/href`
2. Add the License Document to the downloaded Protected Publication at `META-INF/license.lcpl`

Reading Systems SHOULD verify the integrity of downloaded Protected Publication, if a hash is provided.

7.2. User Key processing

Reading Systems MUST:

- Show the text hint and URL when prompting the User for their User Passphrase

Reading Systems SHOULD:

- Store the User Key, but in a secured manner
- Try previously stored User Keys before prompting the User to enter their User Passphrase for a new Protected Publication

Reading Systems MAY:

- Use an alternate technique to discover and exchange the User Key in the background

Reading Systems MUST NOT:

- Store User Passphrase, only the User Key

7.3. Signature Processing

Reading Systems MUST:

- Validate the Signature and Provider Certificate as described in 5.5. Verifying the signature
- Update the Certification Revocation List on a regular basis

Reading Systems SHOULD:

- Be able to auto-update their Root Certificate

Reading Systems MAY:

- Choose to open a Publication even if their own Root Certificate is deprecated

Reading Systems MUST NOT:

- Open a Publication with an invalid Signature or Provider Certificate
- Block a User from opening a Publication if a network connection is unavailable or the Certificate Revocation List can't be reached

7.4. Publication Processing

Reading Systems MUST:

- Respect all rights limitations given in the License Document

Reading Systems MAY:

- Delete a Protected Publication if its License expired

Reading Systems MUST NOT:

- Store unencrypted Publication Resources

Normative references

[JSON] The application/json Media Type for JavaScript Object Notation (JSON).
[JSON Pointer] JavaScript Object Notation (JSON) Pointer.
[OCF] *Open Container Format 3.0.1*.
[Publications] *EPUB Publications 3.0.1*.
[X509] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
[XML-ENC] XML Encryption Syntax and Processing Version 1.1.
[XML-SIG] XML Signature Syntax and Processing (Second Edition).

Informative references

LCP Link Relations Registry
LCP Rights Registry
LCP User Fields Registry
LCP Encryption Profiles Registry

Annex 1. Full example