문서번호 : ANA-20160801-01

문서내용 : 전자책 DRM 상호운용성 평가 기준

문서버전 : v1.0

전자책 DRM 상호운용성 평가 시스템 개발을 위한

전자책 DRM 상호운용성 평가 기준

2016년 08월

목차

[1. 개요 1](#_Toc460409648)

[1.1. 배경 1](#_Toc460409649)

[1.1.1. 전자책 DRM 암호화 명세서 2](#_Toc460409650)

[1.1.2. 전자책 DRM 전자서명 명세서 2](#_Toc460409651)

[1.1.3. 전자책 DRM 인증서 명세서 2](#_Toc460409652)

[1.1.4. 전자책 DRM 라이선스 명세서 3](#_Toc460409653)

[1.2. 표준정합성 검사 목적 및 필요성 3](#_Toc460409654)

[1.3. 문서의 구성 3](#_Toc460409655)

[1.4. 표준정합성 검사 대상 4](#_Toc460409656)

[1.5. 정의 4](#_Toc460409657)

[2. 정합성 검증 절차 8](#_Toc460409658)

[2.1. 데이터 정합성 검증 8](#_Toc460409659)

[2.1.1. 기본 문서 검증 8](#_Toc460409660)

[2.1.1.1. 암호화 문서의 존재 확인 9](#_Toc460409661)

[2.1.1.2. 라이선스 문서의 존재 확인 및 profile 확인 10](#_Toc460409662)

[2.1.2. 암호화 문서 검증 11](#_Toc460409663)

[2.1.2.1. 암호화 문서의 스키마 검증 11](#_Toc460409664)

[2.1.2.2. key의 URI, type 검증 12](#_Toc460409665)

[2.1.2.3. 암호화된 파일 리스트와 실제 파일의 존재 확인 13](#_Toc460409666)

[2.1.2.4. 암호화 제외 대상 파일에 대한 검증 14](#_Toc460409667)

[2.1.2.5. 미디어 타입의 리소스의 압축 여부 검증 16](#_Toc460409668)

[2.1.3. 라이선스 문서 검증 17](#_Toc460409669)

[2.1.3.1. 라이선스 문서 스키마 검증 17](#_Toc460409670)

[2.1.3.2. 라이선스 문서의 key 항목 중복 검증 18](#_Toc460409671)

[2.1.4. 프로파일 검증 19](#_Toc460409672)

[2.1.4.1. 콘텐츠 리소스 암호화 알고리즘 확인 19](#_Toc460409673)

[2.1.4.2. Content key 알고리즘 확인 20](#_Toc460409674)

[2.1.4.3. ECK 기술의 표준 준수 확인 21](#_Toc460409675)

[2.1.4.4. 라이선스 문서의 전자서명 알고리즘 확인 23](#_Toc460409676)

[2.1.4.5. Key 추출 확인 24](#_Toc460409677)

[2.1.4.6. Key check 확인 26](#_Toc460409678)

[2.1.4.7. User object 암호화 확인 28](#_Toc460409679)

[2.1.4.8. 라이선스 문서의 전자서명 확인 30](#_Toc460409680)

[2.1.4.9. 리소스의 실제 복호화 검증 31](#_Toc460409681)

[2.1.5. 전자서명 문서 검증 33](#_Toc460409682)

[2.1.5.1. 전자서명 문서의 존재 확인 33](#_Toc460409683)

[2.1.5.2. 전자서명 문서의 스키마 검증 34](#_Toc460409684)

[2.1.5.3. 전자서명 알고리즘 확인 35](#_Toc460409685)

[2.1.5.4. 정규화 알고리즘 확인 36](#_Toc460409686)

[2.1.5.5. 리소스의 해시 알고리즘 확인 38](#_Toc460409687)

[2.1.5.6. 리소스 데이터의 해시값 검증 40](#_Toc460409688)

[2.1.5.7. 전자서명 내의 해시값 검증 42](#_Toc460409689)

[2.1.5.8. 인증서 체인 검증 44](#_Toc460409690)

[2.1.5.9. 전자서명 확인 46](#_Toc460409691)

[2.1.6. 인증서 검증 47](#_Toc460409692)

[2.1.6.1. 인증서의 존재 확인 47](#_Toc460409693)

[2.1.6.2. 인증서 표현방식 확인 48](#_Toc460409694)

[2.1.6.3. 시리얼번호 양식 확인 49](#_Toc460409695)

[2.1.6.4. 전자서명 알고리즘 확인 50](#_Toc460409696)

[2.1.6.5. 인증서의 체인 검증 확인 51](#_Toc460409697)

[2.1.6.6. 인증서 공개키의 무인 검증 52](#_Toc460409698)

[2.1.6.7. issuer, subject 정보 확인 53](#_Toc460409699)

[2.1.6.8. 기간 검증 확인 54](#_Toc460409700)

[2.1.6.9. 공개키 비도 확인 55](#_Toc460409701)

[2.1.6.10. AuthorityKey 식별자 정보 확인 56](#_Toc460409702)

[2.1.6.11. 상위 인증서 정보 검증 57](#_Toc460409703)

[2.1.6.12. 말단 인증서 정보 검증 58](#_Toc460409704)

[2.2. 기능 정합성 검증 59](#_Toc460409705)

[2.2.1. 기본 문서 검증 59](#_Toc460409706)

[2.2.1.1. 정합 데이터 검증 59](#_Toc460409707)

[2.2.1.1.1. 전자서명 문서가 없는 전자책 열람 확인 59](#_Toc460409708)

[2.2.1.2. 비정합 데이터 검증 60](#_Toc460409709)

[2.2.1.2.1. 라이선스 문서가 없는 전자책 오류 처리 60](#_Toc460409710)

[2.2.1.2.2. 암호화 문서가 없는 전자책 오류 처리 61](#_Toc460409711)

[2.2.2. 복호화 처리 기능 검증 62](#_Toc460409712)

[2.2.2.1. 정합 데이터 검증 62](#_Toc460409713)

[2.2.2.1.1. 암호화된 리소스를 포함한 전자책 열람 확인 62](#_Toc460409714)

[2.2.2.1.2. 압축 암호화된 리소스를 포함한 전자책 열람 확인 63](#_Toc460409715)

[2.2.2.1.3. 압축/비압축 암호화된 리소스들을 포함한 전자책 열람 확인 64](#_Toc460409716)

[2.2.2.1.4. 평문/암호화된 리소스를 포함한 전자책 열람 확인 65](#_Toc460409717)

[2.2.2.1.5. 암호화된 리소스 파일이 없는 암호화문서를 포함하는 전자책 열람 확인 66](#_Toc460409718)

[2.2.2.1.6. User key 방식의 DRM이 적용된 전자책 열람 확인 67](#_Toc460409719)

[2.2.2.1.7. 콘텐츠 암호화 알고리즘이 AES-256-GCM 방식인 전자책 열람 확인 68](#_Toc460409720)

[2.2.2.1.8. 암호화된 이미지 파일이 포함된 전자책 열람 확인 69](#_Toc460409721)

[2.2.2.1.9. 암호화된 음악 파일이 포함된 전자책 열람 확인 70](#_Toc460409722)

[2.2.2.1.10. 암호화된 스크립트 파일이 포함된 전자책 열람 확인 71](#_Toc460409723)

[2.2.2.1.11. 암호화된 동영상 파일이 포함된 전자책 열람 확인 72](#_Toc460409724)

[2.2.2.2. 비정합 데이터 검증 73](#_Toc460409725)

[2.2.2.2.1. 올바르지 않은 암호화 문서를 포함한 전자책 오류 처리 73](#_Toc460409726)

[2.2.2.2.2. Key의 URI, type 정보가 비표준인 전자책 오류 처리 74](#_Toc460409727)

[2.2.2.2.3. 라이선스 문서의 content\_key 영역에 ECK가 없는 전자책 오류 처리 75](#_Toc460409728)

[2.2.2.2.4. 검증대상 기기의 device key가 아닌 key로 암호화된 전자책 오류 처리 76](#_Toc460409729)

[2.2.2.2.5. User key 방식의 전자책에 틀린 비밀번호를 입력하는 경우의 오류 처리 77](#_Toc460409730)

[2.2.2.2.6. 암호화문서에 등록된 암호화 대상 리소스가 존재하는 않는 경우의 오류 처리 78](#_Toc460409731)

[2.2.2.2.7. 암호화된 리소스 파일이 손상된 전자책의 오류 처리 79](#_Toc460409732)

[2.2.3. 인증서 확인 기능 검증 80](#_Toc460409733)

[2.2.3.1. 비정합 데이터 검증 80](#_Toc460409734)

[2.2.3.1.1. 인증서의 기간이 80](#_Toc460409735)

[2.2.4. 전자서명 확인 기능 검증 80](#_Toc460409736)

[2.2.4.1. 정합 데이터 검증 80](#_Toc460409737)

[2.2.4.1.1. 전자서명 문서가 포함된 전자책 열람 확인 80](#_Toc460409738)

[2.2.4.2. 비정합 데이터 검증 81](#_Toc460409739)

[2.2.4.2.1. 올바르지 않은 전자서명 문서를 포함한 전자책 오류 처리 81](#_Toc460409740)

[2.2.4.2.2. 리소스의 해시값이 유효하지 않은 전자책 오류 처리 82](#_Toc460409741)

[2.2.4.2.3. 인증서 체인이 유효하지 않은 경우의 오류 처리 83](#_Toc460409742)

[2.2.4.2.4. 전자서명이 유효하지 않은 경우의 오류 처리 84](#_Toc460409743)

[2.2.5. 라이선스 확인 기능 검증 85](#_Toc460409744)

[2.2.5.1. 정합 데이터 검증 85](#_Toc460409745)

[2.2.5.1.1. 기간 권한이 유효한 전자책 열람 확인 85](#_Toc460409746)

[2.2.5.1.2. copy 권한 제어 확인 86](#_Toc460409747)

[2.2.5.1.3. print 권한 제어 확인 87](#_Toc460409748)

[2.2.5.2. 비정합 데이터 검증 88](#_Toc460409749)

[2.2.5.2.1. 올바르지 않은 라이선스 문서를 포함한 전자책 오류 처리 88](#_Toc460409750)

[2.2.5.2.2. 인증서 체인이 유효하지 않은 경우의 오류 처리 89](#_Toc460409751)

[2.2.5.2.3. 전자서명이 유효하지 않은 경우의 오류 처리 90](#_Toc460409752)

[2.2.5.2.4. 라이선스 문서의 시작일이 검사일 이후인 전자책의 오류 처리 91](#_Toc460409753)

[2.2.5.2.5. 라이선스 문서의 종료일이 검사일 이전인 전자책의 오류 처리 92](#_Toc460409754)

[3. 정합성 검증 방법 93](#_Toc460409755)

[3.1. 데이터 정합성 검증 93](#_Toc460409756)

[3.2. 기능 정합성 검증 93](#_Toc460409757)

[3.3. 보고서 93](#_Toc460409758)

[4. 정합성 검증을 위한 자료 및 도구 94](#_Toc460409759)

[4.1. 개요 94](#_Toc460409760)

[4.2. 검증도구 94](#_Toc460409761)

[4.2.1. 검증 툴 94](#_Toc460409762)

[4.2.2. 검증용 스키마 파일 95](#_Toc460409763)

[4.2.2.1. epub-container.xsd 95](#_Toc460409764)

[4.2.2.2. w3c\_xml\_encryption.profile.xsd 95](#_Toc460409765)

[4.2.2.3. w3c\_xml\_signature\_profile.xsd 97](#_Toc460409766)

[4.2.2.4. readium-ep.xsd 99](#_Toc460409767)

[4.2.2.5. license\_schema.json 99](#_Toc460409768)

[4.3. 검증자료 104](#_Toc460409769)

[4.3.1. 개요 104](#_Toc460409770)

[4.3.2. 기본 문서 검증자료 제작 방법 104](#_Toc460409771)

[4.3.2.1. 전자서명 문서가 없는 전자책 104](#_Toc460409772)

[4.3.2.2. 라이선스 문서가 없는 전자책 104](#_Toc460409773)

[4.3.2.3. 암호화 문서가 없는 전자책 104](#_Toc460409774)

[4.3.3. 복호화 처리 기능 검증 자료 제작 방법 104](#_Toc460409775)

[4.3.3.1. 암호화된 리소스를 포함한 전자책 104](#_Toc460409776)

[4.3.3.2. 압축 암호화된 리소스를 포함한 전자책 105](#_Toc460409777)

[4.3.3.3. 압축/비압축 암호화된 리소스들을 포함한 전자책 105](#_Toc460409778)

[4.3.3.4. 평문/암호화된 리소스들을 포함한 전자책 105](#_Toc460409779)

[4.3.3.5. 암호화된 리소스 파일이 없는 암호화문서를 포함하는 전자책 105](#_Toc460409780)

[4.3.3.6. User key 방식의 DRM이 적용된 전자책 105](#_Toc460409781)

[4.3.3.7. 콘텐츠 암호화 알고리즘이 AES-256-GCM 방식인 전자책 106](#_Toc460409782)

[4.3.3.8. 암호화된 이미지 파일이 포함된 전자책 106](#_Toc460409783)

[4.3.3.9. 암호화된 음악 파일이 포함된 전자책 106](#_Toc460409784)

[4.3.3.10. 암호화된 스크립트 파일이 포함된 전자책 106](#_Toc460409785)

[4.3.3.11. 암호화된 동영상 파일이 포함된 전자책 106](#_Toc460409786)

[4.3.3.12. 올바르지 않은 암호화 문서를 포함한 전자책1 107](#_Toc460409787)

[4.3.3.13. 올바르지 않은 암호화 문서를 포함한 전자책2 107](#_Toc460409788)

[4.3.3.14. key의 URI 정보가 비표준인 전자책 107](#_Toc460409789)

[4.3.3.15. key의 type 정보가 비표준인 전자책 107](#_Toc460409790)

[4.3.3.16. 라이선스 문서의 content\_key 영역에 ECK가 없는 전자책 108](#_Toc460409791)

[4.3.3.17. 검증대상 기기의 device key가 아닌 key로 암호화된 경우 108](#_Toc460409792)

[4.3.3.18. 암호화문서에 등록된 암호화 대상 리소스가 존재하지 않는 전자책 108](#_Toc460409793)

[4.3.3.19. 암호화된 리소스 파일이 손상된 전자책 108](#_Toc460409794)

[4.3.4. 전자서명 확인 기능 검증 자료 제작 방법 109](#_Toc460409795)

[4.3.4.1. 전자서명 문서를 포함한 전자책 109](#_Toc460409796)

[4.3.4.2. 올바르지 않은 전자서명 문서를 포함한 전자책1 109](#_Toc460409797)

[4.3.4.3. 올바르지 않은 전자서명 문서를 포함한 전자책2 109](#_Toc460409798)

[4.3.4.4. 리소스의 해시값이 유효하지 않은 전자책 109](#_Toc460409799)

[4.3.4.5. 인증서 체인이 유효하지 않은 signatures.xml을 포함하는 전자책 109](#_Toc460409800)

[4.3.4.6. 만료된 인증서로 signatures.xml 문서가 전자서명된 전자책 110](#_Toc460409801)

[4.3.4.7. 비도가 낮은 인증서가 포함된 인증서 체인으로 signatures.xml 문서가 전자서명된 전자책 110](#_Toc460409802)

[4.3.4.8. 전자서명이 유효하지 않은 signautres.xml을 포함하는 전자책 110](#_Toc460409803)

[4.3.5. 라이선스 확인 기능 검증 자료 제작 방법 110](#_Toc460409804)

[4.3.5.1. 기간 권한이 유효한 전자책 110](#_Toc460409805)

[4.3.5.2. copy 권한이 없는 전자책 111](#_Toc460409806)

[4.3.5.3. 제한된 copy 권한이 있는 전자책 111](#_Toc460409807)

[4.3.5.4. 무제한의 copy 권한이 있는 전자책 111](#_Toc460409808)

[4.3.5.5. print 권한이 없는 전자책 111](#_Toc460409809)

[4.3.5.6. 제한이 있는 print 권한이 있는 전자책 112](#_Toc460409810)

[4.3.5.7. 무제한의 print 권한이 있는 전자책 112](#_Toc460409811)

[4.3.5.8. 올바르지 않은 라이선스 문서를 포함한 전자책1 112](#_Toc460409812)

[4.3.5.9. 올바르지 않은 라이선스 문서를 포함한 전자책2 112](#_Toc460409813)

[4.3.5.10. 인증서 체인이 유효하지 않은 license.lcpl을 포함한 전자책 112](#_Toc460409814)

[4.3.5.11. 만료된 인증서로 license.lcpl 문서가 전자서명된 전자책 113](#_Toc460409815)

[4.3.5.12. 비도가 낮은 인증서가 포함된 인증서체인으로 license.lcpl 문서가 전자서명된 전자책 113](#_Toc460409816)

[4.3.5.13. 전자서명이 유효하지 않은 license.lcpl를 포함하는 전자책 113](#_Toc460409817)

[4.3.5.14. 라이선스 문서의 시작일이 검사일 이후인 전자책 113](#_Toc460409818)

[4.3.5.15. 라이선스 문서의 종료일이 검사일 이전인 전자책 114](#_Toc460409819)

[4.4. 소스코드 114](#_Toc460409820)

[4.4.1. compression\_checker 114](#_Toc460409821)

[4.4.2. key\_checker 115](#_Toc460409822)

[4.4.3. resource\_decrypter 116](#_Toc460409823)

[4.4.4. user\_info\_decrypter 119](#_Toc460409824)

[4.4.5. u\_checker.py 121](#_Toc460409825)

[4.4.6. signatureValidator.java 123](#_Toc460409826)

문서 버전

|  |  |  |
| --- | --- | --- |
| 일자 | 버전 | 변경사항 |
| 2016.08.01 | V1.0 | 초기 문서 배포 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 개요

## 배경

2007년 9월 IDPF(International Digital Publishing Forum)에서는 e-Book 포맷에 대한 국제표준 전자책 DRM 1.0을 발표하였고 버전 3.0이 발표된 현재 시장에서는 이미 다양한 e-Book 단말기들이 IDPF의 전자책 DRM 표준을 준수하고 있다. 전자책 DRM 표준은 e-Book 콘텐츠에 대한 무차별 복제 및 재배포를 방지하고자 표준문서에 기술적 보호조치 항목을 포함하고 있는데, 이는 W3C XML 표준을 기반으로 한 암호화와 전자서명에 대한 정보공유 방법만을 정의하고 있기 때문에 권리정보 나 키 관리와 같은 DRM(Digital Rights Management)의 추가적인 기능들에 대한 표준이 부재한 상태이다.

이러한 상황에서 전자책 DRM 표준을 준수하고 있는 대다수의 상용 e-Book 솔루션들은 자체적으로 채택하고 있는 DRM의 권리정보 및 키 관리 방법을 사용하고 있기 때문에 기술적 보호조치 부분에 대해서는 호환성을 유지하지 못하고 있다. 이에 따라 IDPF에서는 e-Book 콘텐츠의 저작권 보호를 위해 별도의 DRM 표준화를 추진하고 있으나 관련 산업계의 이해대립으로 구체적인 DRM 기술규격에 대한 표준화를 하지 못하고 있는 실정이다.

이에 대해 저작권위원회는 2011년 e-Book DRM 호환성 해결방안의 일환으로 기존 ePub의 표준 규격을 기준으로 e-Book 콘텐츠 파일의 암호화 방식, 키 전달 방식, 라이선스 정보 및 형식에 대한 권고 규격을 정하고 구체적 구현 샘플과 가이드를 제공하기 위한 프로젝트를 진행하였다. 이에 대한 결과로 2012년 말에 다음과 같이 전자책의 기술적 보호조치를 위한 3가지 표준이 작성되었다.

* 전자책 DRM 암호화 명세서
* 전자책 DRM 전자서명 명세서
* 전자책 DRM 인증서 명세서

위에서 언급된 전자책 DRM 암호화 및 전자서명개의 표준 표준들은 IDPF의 EPUB에서 참조하고 있는 참조표준에 대한 프로파일과 IDPF에서는 언급되지 않았던 부분들에 대한 표준을 포함하고 있다. 각 표준들에 대한 내용을 다음과 같다.

### 전자책 DRM 암호화 명세서

본 표준은 전자책 콘텐츠의 기술적 보호조치를 위해 필요한 암호화 표준 명세서를 정의해 놓은 것이다. 전자책에서의 기존 암호화 표준은 IDPF에서 정의해 놓은 전자책 DRM 표준이 있고, 여기에서는 암호화를 위해 사용되는 표준으로 W3C XML Encryption[[1]](#footnote-2)을 사용하도록 하고 있다. 그러나 W3C 표준에서 사용될 수 있는 암호화 알고리즘과 비도, 키 저장방식이 매우 다양한 형태로 사용될 수 있기 때문에 취약성이 있는 알고리즘과 키 비도가 사용될 경우 보안적인 허점이 존재할 가능성이 존재한다. 또한 표준 명세서를 기반으로 솔루션을 제작해야 하는 산업체 입장에서는 다양한 가능성이 존재하는 W3C 표준을 포괄적으로 적용할 수 있는 제품을 만들기가 매우 어렵다는 문제가 있다. 따라서 본 표준에서는 W3C XML Encryption 표준의 틀의 벗어나지 않는 범위에서 구현이 용이하고 보안적인 취약성이 존재하지 않는 W3C Encryption 프로파일을 지정하고자 한다.

### 전자책 DRM 전자서명 명세서

본 표준은 전자책 콘텐츠의 기술적 보호조치를 위해 필요한 전자서명 표준 명세서를 정의해 놓은 것이다. 전자책에서의 기존 전자서명 표준은 IDPF에서 정의해 놓은 전자책 DRM 표준이 있고, 여기에서는 전자서명을 위해 사용되는 표준으로 W3C XML Signature[[2]](#footnote-3)을 사용하도록 하고 있다. 그러나 W3C 표준에서 사용될 수 있는 전자서명/해시 알고리즘과 비도, 키 관리방식이 매우 다양한 형태로 사용될 수 있기 때문에 취약성이 있는 알고리즘과 키 비도가 사용될 경우 보안적인 허점이 존재할 가능성이 존재한다. 또한 표준 명세서를 기반으로 솔루션을 제작해야하는 산업체 입장에서는 다양한 가능성이 존재하는 W3C 표준을 포괄적으로 적용할 수 있는 제품을 만들기가 매우 어렵다는 문제가 있다. 따라서 본 표준에서는 W3C XML Signature 표준의 틀의 벗어나지 않는 범위에서 구현이 용이하고 보안적인 취약성이 존재하지 않는 W3C XML Signature 프로파일을 지정하고자 한다.

### 전자책 DRM 인증서 명세서

본 표준은 전자책 콘텐츠의 기술적 보호조치를 위해 필요한 X.509 인증서의 프로파일을 정의해 놓은 것이다. 전자책 콘텐츠의 기술적 보호조치를 구현함에 있어 암호화와 전자서명에 사용되는 인증서는 기본적으로 X.509 표준을 따르는 것을 원칙으로 한다. 그러나 현재의 X.509 포맷은 사용할 수 있는 알고리즘 및 형식이 매우 다양하기 때문에, 범용표준으로 적용하기에는 구현 및 검증이 매우 복잡하다. 또한 알고리즘의 종류 및 비도 등이 구체적이지 않아 보안에 취약한 알고리즘과 비도가 사용될 가능성도 존재한다. 따라서 전자책 DRM 표준에서는 이러한 문제점을 해결하기 위해 ITU-T X.509 인증서 표준의 사용에 있어 프로파일을 정의하여 표준 구현에 따른 용이함을 제공하고자 한다.

### 전자책 DRM 라이선스 명세서

본 표준은 전자책 콘텐츠의 기술적 보호조치를 위해 필요한 라이선스 표준 명세서를 정의해 놓은 것이다. 전자책에서의 기존 라이선스 표준은 IDPF에서 정의해 놓은 전자책 DRM 표준이 Readium Licensed Content Protection Profile에 정의되어 있다. 그러나 ECK의 암호화에 대칭키 방식의 알고리즘만을 사용하고 있는 등의 문제가 있다. 따라서 전자책 DRM 표준에서는 이러한 문제점을 해결하기 위해 Readium Licensed Content Protection Profile을 확장한 표준을 정의해 놓은 것이다.

## 표준정합성 검사 목적 및 필요성

비록 전자책 DRM에 대한 표준들이 기존 참조표준을 단순화한 프로파일 형태를 유지하고 있기는 하지만, 솔루션 개발업체들은 표준 문서에 대한 해석을 잘못하거나, 구현상의 실수로 인해 표준에서 명시하고 있는 명세서를 만족시키지 못할 경우가 발생할 수 있다. 이는 표준대로 제품을 개발하였다고 하더라도 타 제품과의 호환성이 보장되지 않는 결과를 낳게 된다. 이에 따라 개발된 전자책 DRM 응용프로그램이나 기술적 보호조치가 된 전자책을 제작하였을 때, 표준에 적합하게 작업이 완료되었는지를 검사할 수 있는 테스트 시나리오 및 테스트 데이터에 대한 표준화 작업이 요구된다.

테스트 시나리오는 우선 기술적 보호조치된 전자책이 표준에 부합하는지를 검증할 수 있어야 하고, 전자책 응용프로그램이 표준 전자책에 대해서 정상적으로 작동을 하는지, 표준이 아닌 전자책에 대해서 적절하게 대처를 하는지에 대한 검증작업이 필요하다. 그리고 전자책 응용프로그램의 기술적 보호조치 각각의 기능을 검증하기 위해 검증데이터들도 함께 제공되어야 하는데, 이는 표준에 부합하는 데이터와 표준에 부합하지 않는 데이터를 포함하고 있어야 한다.

## 문서의 구성

본 문서는 다음과 같은 구성으로 이루어져 있다.

1. 개요

표준정합성 검사 배경 및 문서 개요

1. 정합성 검증 절차

전자책 DRM 데이터 검증, 기능 정합성 검증 절차 기술

1. 정합성 검증 방법

전자책 DRM 데이터/기능 정합성 검증 방법 기술 및 결과 보고서 작성 방법 기술

1. 정합성 검증을 위한 자료 및 도구

전자책 DRM 데이터/기능 정합성 검증에 사용되는 검증 자료 및 검증 도구에 대한 기술

## 표준정합성 검사 대상

전자책 DRM 정합성 검증 대상은 다음과 같은 2가지 분류로 구분된다.

* 전자책 DRM 데이터

전자책 DRM 포맷(encryption.xml, signatures.xml, license.lcpl 및 전자서명에 사용된 인증서)

* 전자책 DRM 뷰어 시스템

전자책 DRM 콘텐츠를 열람하기 위한 전자책 DRM Viewer

## 정의

**AES (Advanced Encryption Standard)**

미국 NIST(National Institute of Standard and Technology)에서 표준으로 정하고 있는 대칭키 암호화 알고리즘. 전자책 DRM 표준에서는 콘텐츠를 암호화하는 알고리즘으로 사용된다.

**ASN.1 (Abstract Syntax Notation.1)**

통신 프로토콜을 위한 메시지를 정의하는 용도로 사용되는 추상적인 개념의 데이터 표현방식으로 실질적으로 데이터를 전송하기 위해서는 다양한 형태의 인코딩 방식이 사용된다. 본 표준에서는 X.509 인증서 데이터를 표현하기 위한 방법으로 사용된다.

**BER (Basic Encoding Rule)**

ASN.1을 인코딩하기 위한 하나의 방식으로 TLV(Type-Length-Value) 형태의 데이터 포맷을 갖는다

**CA (Certificate Authority)**

인증서를 발급해 주는 곳으로 root 인증서를 가지고 있다.

**Canonicalization**

전자서명의 대상이 되는 XML문서는 comment, whitespace, namespace 등의 영향으로 XML 콘텐츠 내용과는 무관하게 전자서명 검증에 대한 오류가 발생할 가능성이 존재하는데, 이를 방지하고자 XML 문서를 정규화하는 방법을 표준화하고 이를 전자서명 생성 및 검증시 적용하는 것을 말한다.

**CEK (Content Encryption Key)**

전자책 콘텐츠를 암호화 및 복호화하는데 사용되는 키

**DER (Distinguished Encoding Rule)**

ASN.1을 인코딩하기 위한 하나의 방식으로 BER 방식의 부분집합이다. BER 방식에서 전송자 옵션부분이 삭제된 규격이 DER 방식이다.

**DN (Distinguished Name)**

인증서 표준인 X.509에서 subject 또는 issuer 이름에 사용되는 고유 ID로 RFC-2253 표준방식을 준수하여야 한다.

**DnQualifier**

인증서의 공개키 값에 대한 SHA1 해시값으로 인증서의 고유 식별자로 사용된다.

**ECK**

Content key를 암호화 하는데 사용되는 키. 본 표준에서는 user key 또는 device key를 말한다.

**EPUB (Electronic Publishing)**

IDPF에서 지정한 전자책 출판 표준 포맷

**GT (Generalized Time)**

날짜와 시간을 표시하는 방식으로 YYYYMMDDhhmmssZ 와 같은 형식으로 표시된다.

**HMAC (Keyed-Hash Message Authentication Code)**

미국 NIST(National Institute of Standard and Technology)에서 표준으로 정하고 있는 키 기반의 메시지 인증 알고리즘.

**IDPF (International Digital Publishing Forum)**

전자책 표준 포맷을 위해 1999년에 결성된 OEBF(Open eBook Forum)가 2005년 명칭이 변경된 포럼으로 120개 이상의 디지털 출판 산업의 기업과 단체들이 결성한 전자출판 표준화 기구이다.

**KEK (Key Encryption Key)**

CEK를 암호화하는데 사용되는 키

**KDK (Key Decryption Key)**

CEK를 복호화하는데 사용되는 키

**Leaf 인증서**

인증서 체인상 최하단에 존재하는 인증서로서, 전자서명 확인용과 키 암호화용으로 용도가 분류된다.

**MGF1 (Mask Generation Function 1)**

해시를 기반으로 한 키 전달 알고리즘에 사용되는 함수

**MIME (Multipurpose Internet Mail Extension)**

e-Mail을 콘텐츠 타입을 표기하기 위한 인터넷 표준으로 본 표준에서는 전자책 콘텐츠의 타입을 나타내기 위해 사용되고 있다.

**OAEP (Optimal Asymmetric Encryption Padding)**

비대칭키 암호화 방식에서 암호화할 대상의 길이가 작을 때 발생할 수 있는 다양한 해킹 시도에 대한 취약성을 방지하기 위해 사용되는 패딩방식이다.

**OPF (Open Packaging Format)**

전자책 국제 표준인 IDPF에서 규정한 전자책 내부의 메타정보, 구조 및 콘텐츠 위치, 목차개요 등에 대한 정보를 담고 있는 파일 형식

**PEM (Privacy Enhanced Mail)**

이진형태의 데이터를 base64인코딩을 통해 문자열 형태로 보여주거나 저장하는 형식으로, 본 표준에서는 DER 포맷의 인증서 이진데이터를 텍스트 형태의 파일로 저장하기 위한 포맷으로 사용된다.

**RSA (Rivest, Shamir, Adlerman)**

비대칭키 암호화 방식으로 본 표준에서는 CEK를 암호화하는 알고리즘으로 사용된다.

**Root 인증서**

인증서 체인상 최상단에 존재하는 인증서로서, CA가 보유하고 있으며 signer 또는 leaf 인증서 서명확인 용도로만 사용된다.

**SHA1 (Standard Hash Algorithm)**

미국 NIST(National Institute of Standard and Technology)에서 표준으로 정하고 있는 메시지 축약 알고리즘으로 결과 값이 160비트로 출력된다. 본 표준에서는 DnQualifier를 생성하기 위한 메시지 축약 알고리즘으로 사용된다.

**SHA256 (Standard Hash Algorithm)**

미국 NIST(National Institute of Standard and Technology)에서 표준으로 정하고 있는 메시지 축약 알고리즘으로 결과 값이 256비트로 출력된다. 본 표준에서는 인증서 서명 대상을 만들기 위한 메시지 축약 알고리즘으로 사용된다.

**Signer 인증서**

인증서 체인상의 중간 단계의 인증서로서, signer 또는 leaf 인증서 서명확인 용도로만 사용된다.

**Transform**

전자서명의 대상이 되는 XML 콘텐츠의 hash값을 추출하기 전에 사용될 정규화 방법

**UTC (Coordinated Universal Time)**

날짜와 시간을 표시하는 방식으로 YYMMDDhhmmssZ와 같은 형식으로 표시된다.

**X.509 Certificate**

ITU-T X.509 표준을 준수하는 인증서로서 본 표준에서 전자서명용 인증서로 사용된다.

# 정합성 검증 절차

전자책에 대한 기술적 보호조치 표준의 준수여부를 확인하기 위한 정합성 검증대상은 다음과 같이 4가지 표준 명세서를 기반으로 한다.

* 전자책 DRM 암호화 명세서
* 전자책 DRM 라이선스 명세서
* 전자책 DRM 전자서명 명세서
* 전자책 DRM 인증서 명세서

검증은 검증 대상의 특성에 따라 검증 방법이 달라지는데, 전자책 콘텐츠가 각각의 표준명세서에 부합하는지 여부를 확인하기 위한 데이터 정합성 검증이 있고, 전자책 콘텐츠를 처리하는 뷰어 시스템이 표준을 준수하는 전자책을 정상적으로 처리할 수 있는지 여부를 확인하기 위한 기능 정합성 검증이 있다. 본 장에서는 전자책의 기술적 보호조치에 필요한 표준의 준수 여부를 확인하기 위한 이들 데이터 정합성 검증, 기능 정합성 검증의 자세한 방법에 대하여 설명한다.

## 데이터 정합성 검증

전자책 기술적 보호조치 표준의 데이터 정합성 검증에는 표준에서 지정하고 있는 기본 문서의 포함 여부 검증, 암호화 문서의 표준정합성 준수에 대한 검증, 라이선스 문서 의 표준정합성 준수에 대한 표준정합성 준수에 대한 검증, 표준 프로파일의 준수에 대한 검증, 전자서명 문서의 표준정합성 준수에 대한 검증, 전자서명에 사용된 인증서의 표준정합성 준수에 대한 검증 과정이 포함된다.

### 기본 문서 검증

전자책 DRM 상호운용성 평가를 위해 검증 대상 전자책이 표준 DRM이 적용된 전자책 콘텐츠인지 확인이 필요하다. 표준 DRM의 적용여부는 암호화 문서와 라이선스 문서, 라이선스 문서의 프로파일 영역을 확인한다.

#### 암호화 문서의 존재 확인

* **목적**

검증 대상 전자책에 암호화 문서가 “META-INF/encryption.xml” 경로에 존재하는지 검증한다.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령문을 통해 encryption.xml 파일을 추출한다. META-INF 폴더에 encryption.xml 파일이 추출되지 않거나, 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다. 테스트에 실패한 경우 나머지 테스트는 진행하지 않는다.

|  |
| --- |
| $unzip test.epub META-INF/encryption.xml  Archive: test.epub  extracting: META-INF/encryption.xml |

1. 파일 추출에 성공하였다면 다음의 명령문을 통해 문서의 charset을 확인한다. UTF-8이 아닌 경우 이 검증은 실패이다.

|  |
| --- |
| $file -bi META-INF/encryption.xml  application/xml; charset=utf-8 |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [전자책 DRM Open Container Format(OCF) 3.0] | - 2.5.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - unzip |  |

#### 라이선스 문서의 존재 확인 및 profile 확인

* **목적**

검증 대상 전자책에 라이선스 문서가 "META-INF/license.lcpl" 경로에 존재하는지 검증한다.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령문을 통해 license.lcpl 파일을 추출한다. META-INF 폴더에 license.lcpl 파일이 추출되지 않거나, 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다. 검증에 실패한 경우 나머지 테스트는 진행하지 않는다.

|  |
| --- |
| $unzip test.epub META-INF/license.lcpl  Archive: test.epub  extracting: META-INF/license.lcpl |

1. 파일 추출에 성공하였다면 다음의 명령문을 통해 문서의 charset을 확인한다. UTF-8이 아닌경우 이 검증은 실패이다.

|  |
| --- |
| $file -bi META-INF/license.lcpl  text/plain; charset=utf-8 |

1. 다음의 명령문을 통해 라이선스의 profile을 확인한다. "http://kats.org/lcp/profile-1.0"가 아닌 경우 이 검증은 실패이다.

|  |
| --- |
| $jq '.encryption.profile' META-INF/license.lcpl  "http://kats.org/lcp/profile-1.0" |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.1, 4.4.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - unzip  - jq |  |

### 암호화 문서 검증

IDPF EPUB표준의 encryption.xml은 전자책 콘텐츠를 보호하기 위해 암호화한 콘텐츠들의 파일정보와 암호화에 사용된 키 정보들이 W3C XML Encryption 표준을 사용하여 표기되어 있다. 본 표준정합성 검사 계획문서에서는 IDPF 전자책 DRM 표준의 encryption.xml의 프로파일 형태로 TTA 산업표준으로 제안된 ‘전자책 DRM 암호화 명세서’에 대한 표준 정합성을 검증하는 과정을 다룬다.

#### 암호화 문서의 스키마 검증

* **목적**

검증 대상 전자책의 encryption.xml 문서가 표준 스키마 문법을 준수하는지 검증한다.

* **사전조건**

2.1.1.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령문을 통해 추출한 encryption.xml 파일이 표준 스키마를 준수하는지 확인한다. 어떠한 오류 메시지가 표시된다면 이 검증은 실패이고 나머지 테스트는 진행하지 않는다.

|  |
| --- |
| $xmllint -schema epub-container.xsd META-INF/encryption.xml –noout  META-INF/encryption.xml validates |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제2부 암호화 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmllint | - epub-container.xsd  - w3c\_xml\_encryption.profile.xsd  - w3c\_xml\_signature\_profile.xsd  - readium-ep.xsd |

#### key의 URI, type 검증

* **목적**

검증 대상 전자책의 encryption.xml 문서의 모든 EncryptedData에 대해서, KeyInfo의 @URI와 @Type이 표준을 준수하는지 검증한다.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령문을 통해 추출한 encryption.xml 파일이 콘텐츠 암호화 키에 대해 @URI 값이 "license.lcpl#/encryption/content\_key"임을 확인한다. 표준과 맞지 않다면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -N y="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:EncryptedData/y:KeyInfo/y:RetrievalMethod/@URI" META-INF/encryption.xml  license.lcpl#/encryption/content\_key  license.lcpl#/encryption/content\_key  ... |

1. 다음의 명령문을 통해 추출한 encryption.xml 파일의 콘텐츠 암호화 키에 대한 @Type 값이 "http://readium.org/2014/01/lcp#EncryptedContentKey"임을 확인한다. 만약 같지 않다면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -N y="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:EncryptedData/y:KeyInfo/y:RetrievalMethod/@Type" META-INF/encryption.xml  http://readium.org/2014/01/lcp#EncryptedContentKey  http://readium.org/2014/01/lcp#EncryptedContentKey  ... |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제2부 암호화 | - 4.4.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### 암호화된 파일 리스트와 실제 파일의 존재 확인

* **목적**

검증 대상 전자책의 encryption.xml 문서에 등록된 모든 암호화 대상 파일이 전자책 컨테이너에 존재하는지 검증한다.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령문을 통해 추출한 encryption.xml 파일에 등록된 암호화 대상 파일의 리스트를 구한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -N y="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:EncryptedData/x:CipherData/x:CipherReference/@URI" META-INF/encryption.xml  EPUB/c2.xhtml EPUB/c1.xhtml  ... |

1. 다음의 명령문을 통해 전자책 컨테이너에 위에서 구한 파일이 존재하는지 확인한다. 만약 존재하지 않는다면 이 검증은 실패이다.

|  |
| --- |
| $unzip -l trees\_user\_repack.epub | grep 'EPUB/c1.xhtml'  1344 08-02-2016 20:27 EPUB/c1.xhtml  $unzip -l trees\_user\_repack.epub | grep 'EPUB/c2.xhtml' 2064 08-02-2016 20:27 EPUB/c2.xhml  ... |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제2부 암호화 | - 4.2.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - unzip |  |

#### 암호화 제외 대상 파일에 대한 검증

* **목적**

검증대상 전자책이 표준에서 지정한 암호화 금지대상 파일에 대해 암호화 하지 않는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 암호화하지 말아야 하는 시스템 파일의 리스트는 다음과 같다.

- mimetype

- META-INF/container.xml

- META-INF/encryption.xml

- META-INF/license.lcpl

- META-INF/manifest.xml

- META-INF/metadata.xml

- META-INF/rights.xml

- META-INF/signatures.xml

1. 다음의 명령문은 해당 [@URI=] 옵션을 통해 CipherReference에 해당 URI에 맞는 항목을 보여준다. 이 명령문을 통해 추출한 encryption.xml 파일에 암호화 되지 말아야 할 시스템 파일이 포함되어 있는지 확인한다. 입력한 URI가 결과로 표시되면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -t -v "//x:EncryptedData/x:CipherData/x:CipherReference[@URI='mimetype']/@URI" META-INF/encryption.xml |

1. 다음의 명령문을 통해 container.xml을 추출하고, container.xml에 등록된 opf 파일의 경로를 구한다.

|  |
| --- |
| $unzip trees.epub META-INF/container.xml  Archive: trees.epub  extracting: META-INF/container.xml |

|  |
| --- |
| $xmlstarlet sel -N x="urn:oasis:names:tc:opendocument:xmlns:container" -t -v "//x:rootfile/@full-path" META-INF/container.xml  EPUB/package.opf |

1. 같은 방식으로 opf 파일을 추출하고, 다음의 명령문을 통해 opf 파일의 properties가 'nav', 'cover-image'인 파일, 미디어 타입이 'application/x-dtbncx+xml'인 파일이 존재한다면 그 경로를 구한다.

|  |
| --- |
| $unzip trees.epub EPUB/package.opf  Archive: trees.epub  extracting: EPUB/package.opf |

|  |
| --- |
| $xmlstarlet sel -N x="http://www.idpf.org/2007/opf" -t -v "//x:manifest/x:item[@properties='nav']/@href" EPUB/package.opf  bk01-toc.xhtml |

|  |
| --- |
| $xmlstarlet sel -N x="http://www.idpf.org/2007/opf" -t -v "//x:manifest/x:item[@properties='cover-image']/@href" EPUB/package.opf  image/cover.jpg |

|  |
| --- |
| $xmlstarlet sel -N x="http://www.idpf.org/2007/opf" -t -v "//x:manifest/x:item[@media-type='application/x-dtbncx+xml']/@href" EPUB/package.opf  bc03-ncx.xhtml |

1. 위에서 구한 모든 파일들에 대하여 암호화된 리스트에 존재하는지 확인한다. URI 값이 출력된다면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -t -v "//x:CipherData/x:CipherReference[@URI='EPUB/bk01-toc.xhtml']/@URI" META-INF/encryption.xml |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - Readium Licensed Content Protection(LCP) 1.0 | - 2.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - unzip |  |

#### 미디어 타입의 리소스의 압축 여부 검증

* **목적**

검증 대상 전자책이 표준에서 지정한 압축 금지 대상 미디어 타입 리소스 파일에 대해 압축 하는지 검증한다.

* **사전조건**
* **검증절차**

1. 2.1.2.4. 검증 항목에서 추출한 opf 파일을 준비한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.idpf.org/2007/opf" -t -v "//x:manifest/x:item[contains(@media-type, \"audio\") or contains(@media-type, \"video\")]/@href" EPUB/package.opf  video/shared-culture.mp4 audio/asharedculture\_soundtrack.mp3 |

1. 다음의 명령문을 통해 검증 대상 전자책이 미디어 파일을 포함하는지 확인한다.
2. 다음의 툴과 명령문을 통해 미디어 파일을 압축하지 않고 있음을 확인한다. 이때 두번째 입력 파라메터는 opf의 폴더 경로를 추가한 값을 입력한다. 만약 압축되었다거나 어떠한 에러 메시지가 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $./compression\_checker “META-INF/encryption.xml” “EPUB/video/shared-culture.mp4”  Resource is uncompressed |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - compression\_checker |  |

### 라이선스 문서 검증

#### 라이선스 문서 스키마 검증

* **목적**

검증 대상 전자책의 라이선스 문서가 표준 스키마 문법을 준수하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서와 license\_schema.json파일을 준비한다.
2. 다음의 명령문을 통해 추출한 라이선스 문서가 표준 스키마를 준수하는지 확인한다. 어떠한 오류 메시지가 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $java -jar json-schema-validator-2.2.6-lib.jar license\_schema.json META-INF/license.lcpl  --- BEGIN /root/META-INF/license.lcpl--- validation: SUCCESS --- END /root/META-INF/license.lcpl--- |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - license\_schema.json | - license\_schema.json |

#### 라이선스 문서의 key 항목 중복 검증

* **목적**

검증 대상 전자책의 라이선스 문서에서 key를 중복사용 하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서를 준비한다.
2. unique check tool과 다음의 명령문을 통해 추출한 license.lcpl 문서의 key값이 unique한지 확인한다. unque check tool은 추가적으로 encryption의 user\_key, device\_key의 중복 사용 또는 존재하지 않는지를 검증한다. 만약 license.lcpl 표준에서 사용하고 있는 key값의 중복, encryption의 user\_key, device\_key 관련 에러 메시지가 표시된다면 이 검증은 실패이다. 단, license.lcpl 표준에서 사용하지 않는 key의 중복은 고려치 않는다.

|  |
| --- |
| $python u\_check.py META-INF/license.lcpl  //encryption/content\_key/: "algorithm" is duplicated. (content\_key/algorithm 중복) //encryption/: ECK is duplicated. (user\_key, device\_key 복수 존재) |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - python  - u\_check.py |  |

### 프로파일 검증

#### 콘텐츠 리소스 암호화 알고리즘 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 콘텐츠 암호화 알고리즘을 사용하고 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 encryption.xml 문서 파일을 준비한다.
2. xmlstarlet 툴과 다음의 명령문을 통해 콘텐츠 암호화 알고리즘이 표준을 준수하는지 확인한다. 만약 표준이 아닌 알고리즘이라면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2001/04/xmlenc#" -t -v "//x:EncryptionMethod/@Algorithm" META-INF/encryption.xml  http://www.w3.org/2001/04/xmlenc#aes256-cbc http://www.w3.org/2001/04/xmlenc#aes256-gcm .. |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 8.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### Content key 알고리즘 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 콘텐츠 키 암호화 알고리즘을 사용하고 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다.
2. jq 툴과 다음의 명령문을 통해 콘텐츠 키 암호화 알고리즘이 표준을 준수하는지 확인한다. 만약 표준이 아닌 알고리즘이라면 이 검증은 실패이다.

|  |
| --- |
| $jq .encryption.content\_key.algorithm META-INF/license.lcpl  "http://www.w3.org/2001/04/xmlenc#aes256-cbc"  "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 8.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq |  |

#### ECK 기술의 표준 준수 확인

* **목적**

검증 대상 전자책의 라이선스 문서의 ECK에 대한 기술(記述)이 표준을 준수하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다. 만약 license.lcpl의 ECK가 존재하지 않는다면 이 검증은 실패이다.
2. 만약 license.lcpl의 ECK가 user\_key가 아니라면 이 검증은 진행하지 않는다. jq 툴과 다음의 명령문을 통해 유저 키 암호화 알고리즘이 표준을 준수하는지 확인한다. 만약 표준이 아닌 알고리즘이라면 이 검증은 실패이다.

|  |
| --- |
| $jq .encryption.user\_key.algorithm META-INF/license.lcpl  “http://www.w3.org/2001/04/xmlenc#sha256” |

1. 만약 license.lcpl의 ECK가 device\_key가 아니라면 이 검증은 진행하지 않는다. 다음의 과정을 통해 device key의 key name을 검증한다.
   1. jq 툴과 다음의 명령문을 통해 디바이스 키의 키 네임을 구한다.

|  |
| --- |
| $jq .encryption.device\_key.key\_name META-INF/license.lcpl  "dnQualifier=vkHUvyFj1z28vPzHKKLyQTLuec0=, CN=device:dev00111234, O=book company, OU=book team" |

* 1. 검증 대상 전자책 콘텐츠의 packaging에 사용된 device 인증서와 다음의 명령문을 통해 디바이스 인증서의 Subject 정보를 구한다.

|  |
| --- |
| $openssl x509 -text -noout -in <certificate> | grep Subject:  Subject: dnQualifier=vkHUvyFj1z28vPzHKKLyQTLuec0=, CN=device:dev00111234, O=book company, OU=book team |

* 1. 위의 두 결과 값에 O, OU, CN, dnQualifier 항목이 존재하지 않거나, 서로 다를 경우 이 검증은 실패이다.
* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq  - openssl |  |

#### 라이선스 문서의 전자서명 알고리즘 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 전자서명 알고리즘을 사용하고 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다.
2. jq 툴과 다음의 명령문을 통해 라이선스 전자서명 알고리즘이 표준을 준수하는지 확인한다. 만약 표준이 아닌 알고리즘이라면 이 검증은 실패이다.

|  |
| --- |
| $jq .signature.algorithm META-INF/license.lcpl  "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 8.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq |  |

#### Key 추출 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 AES 암호화 알고리즘의 패딩 방식을 사용하고 있는지 검증한다.

* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다. 만약 license.lcpl의 ECK가 user\_key이고, content\_key의 알고리즘이 “http://www.w3.org/2001/04/xmlenc#aes256-cbc”가 아니라면 검증은 진행하지 않는다.
2. jq 툴과 다음의 명령문을 통해 라이선스 콘텐트 키 암호화 값의 길이를 확인한다. 만약 결과값이 64가 아니라면 이 검증은 실패이다.

|  |
| --- |
| $jq -r .encryption.content\_key.encrypted\_value META-INF/license.lcpl | openssl base64 -d -A | wc -c  64 |

1. jq 툴과 다음의 명령문을 통해 user\_key/key\_check 값의 길이를 확인한다. 만약 결과값이 예상값과 다르다면 이 검증은 실패이다.

|  |
| --- |
| $original\_size=$(cat META-INF/license.lcpl | jq -r .id | wc -c)  $echo $(($original\_size + 32 - $original\_size%16))  64 |

|  |
| --- |
| $jq -r .encryption.content\_key.user\_key.key\_check META-INF/license.lcpl | openssl base64 -d -A | wc -c  64 |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 8.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq  - openssl |  |

#### Key check 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 방식으로 key check를 추가하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다. ECK가 user\_key라면 password를 device\_key라면 private key 파일을 준비한다.
2. 키체크 항목을 복호화한 값은 라이선스 문서의 id 값이다. id 값은 다음의 명령문을 통해 확인할 수 있다.

|  |
| --- |
| $jq -r .id META-INF/license.lcpl  id231-123-1231 |

1. 디바이스 키라면 이 항목은 검증 하지 않는다. key\_checker와 다음의 명령문을 통해 유저 키를 체크한다. 만약 passphrase is correct가 표시되지 않거나, 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $./key\_checker META-INF/license.lcpl <password>  iv:: ead3481eca910ab7e97b08fc1ddc0ec7  key:: 6293a54f2fc91f6600498fd5aa37baa7d805bd621b95502aa8b7b55ebe8d753c  original\_id :: id231-123-1231  decrypted\_key\_check :: id231-123-1231  passphrase is correct |

1. 유저 키라면 이 항목은 검증 하지 않는다. 비밀키와 다음의 명령문을 통해 디바이스 키를 체크한다. 결과값이 라이선스 문서의 id 값과 같지 않다면 이 검증은 실패이다.

|  |
| --- |
| $echo `jq -r .encryption.device\_key.key\_check META-INF/license.lcpl | openssl base64 -d -A | openssl rsautl -decrypt -oaep -inkey key.pem`  id231-123-1231 |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4.3, 4.4.4. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq  - openssl  - key\_checker |  |

#### User object 암호화 확인

* **목적**

검증 대상 전자책이 표준에서 지정하고 있는 방식으로 유저 정보를 보호하고 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl 문서 파일을 준비한다. ECK가 user\_key라면 password를 device\_key라면 private key 파일을 준비한다.
2. 유저 정보를 암호화한 방식의 key의 암호화 방식과 동일하다. 암호화된 유저 정보의 리스트는 다음의 명령문을 통해 알수 있다. 암호화된 유저 정보의 리스트가 없다면 이 검증은 진행하지 않는다.

|  |
| --- |
| $jq -r .user.encrypted META-INF/license.lcpl  [  "email"  ] |

1. ECK가 디바이스 키라면 이 항목은 검증하지 않고 4. 항목을 진행한다. user\_info\_decrypter와 다음의 명령문을 통해 암호화된 유저 정보를 확인한다. 만약 제대로된 유저 정보가 표시되지 않거나, 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $ ./user\_info\_decryptor <user info key> META-INF/license.lcpl <password>  iv:: ca859dfa264b514ad40017ba65ee7d6d  key:: 6293a54f2fc91f6600498fd5aa37baa7d805bd621b95502aa8b7b55ebe8d753c  encryped\_user\_info :: j6jkt8s7zk9qgBEtqVQ1I+mzNuzRqp5OOrQTxp7O8occ9hlSNS6ihEw5K1e+gyn3  decrypted\_user\_info:: testuser@email.com |

1. ECK가 유저 키라면 이 항목은 검증하지 않는다. 다음의 명령문을 통해 암호화된 유저 정보를 확인한다. 만약 제대로된 유저 정보가 표시되지 않거나, 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $echo `jq -r .user.email META-INF/license.lcpl | openssl base64 -d -A | openssl rsautl -decrypt -oaep -inkey key.pem`  testuser@email.com |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.7. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - jq |  |

#### 라이선스 문서의 전자서명 확인

* **목적**

signaturex.xml 문서의 전자서명의 유효성을 검증한다.

* **사전조건**
* **검증절차**

1. 추출한 license.lcpl과 root인증서를 준비한다.
2. signatureValidator 툴 다음의 명령문은 라이선스 전자서명에 사용된 인증서 체인의 정합성 검증과 전자서명 검증을 확인한다. signature is correct가 표시되지 않거나, 어떠한 에러 메시지가 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $java -cp .;./gson-2.3.1.jar;./bcprov-jdk15on-152.jar signatureValidator root.pem META-INF/license.lcpl  signature is correct |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 7.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - signatureValidator | - gson-2.3.1.jar  - bcprov-jdk15on-152.jar |

#### 리소스의 실제 복호화 검증

* **목적**

검증 대상 전자책이 encryption 문서에서 지정한 알고리즘으로 리소스를 암호화하고 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 license.lcpl문서 파일과 encryption.xml 문서 파일을 준비한다. 그리고 ECK가 user\_key라면 password를 device\_key라면 private key 파일을 준비한다.
2. ECK가 user key이면 resource\_decrypter와 비밀번호, 다음의 명령문을 통해 리소스를 복호화할 수 있다. 만약 어떠한 에러메시지가 표시되거나, 대상 파일이 제대로 복호화되지 않았다면 이 검증은 실패이다.

|  |
| --- |
| $./resource\_decrypter “EPUB/c1.xhtml” “META-INF/license.lcpl” “META-INF/encryption.xml” “lcptest”  license :: META-INF/license.lcpl  passphrase :: lcptest  userkey :: 6293a54f2fc91f6600498fd5aa37baa7d805bd621b95502aa8b7b55ebe8d753c  resource encryption algorithm= http://www.w3.org/2001/04/xmlenc#aes256-cbc  cek\_iv :: d19381dd83b54d29f30d948af47a483b  cek :: de84024a14852534e336df6fc366fb81d280678544134886a1fbf2d11ea26ea9  resource\_iv:: c6c6557d5a6fd903cc1eabe0cb424722  encryped\_resource :: EPUB/c1.xhtml  decrypted\_resource :: EPUB/c1.xhtml.out |

1. ECK가 device key라면 resource\_decrypter와 암호화 인증서의 private key, 다음의 명령문을 통해 리소스를 복호화할 수 있다. 만약 어떠한 에러메시지가 표시되거나, 대상 파일이 제대로 복호화되지 않았다면 이 검증은 실패이다.

|  |
| --- |
| $./resource\_decrypter “EPUB/c1.xhtml” “META-INF/license.lcpl” “META-INF/encryption.xml” “key.pem”  license :: META-INF/license.lcpl  device key :: key.pem  resource encryption algorithm= http://www.w3.org/2009/xmlenc11#aes256-gcm  cek\_iv :: 7964ffe5acf5076520b8cc483cb2bd76  cek :: 8af11b1390c7678878559c404dc1c16be63f10f232c2e170b0673e51df8c314a  resource\_iv:: fcbc8cf9d400b389b03c8fff  encryped\_resource :: EPUB/c1.xhtml  decrypted\_resource :: EPUB/c1.xhtml.out |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제2부 암호화  - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | -4.2.  - 4.4, 8.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - resource\_decrypter |  |

### 전자서명 문서 검증

#### 전자서명 문서의 존재 확인

* **목적**

검증 대상 전자책에 전자서명 문서가 "META-INF/signatures.xml" 경로에 존재하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠를 준비한다.
2. 다음의 명령어를 사용하여 signatures.xml 파일을 추출한다. 만약 signatures.xml 추출에 실패하거나, 어떤 에러 메시지라도 표시된다면 이 테스트의 결과는 실패이다.

|  |
| --- |
| $unzip test.epub META-INF/signatures.xml  Archive: test.epub extracting: META-INF/signatures.xml |

1. 테스트에 실패한 경우 [2.1.5. 전자서명 문서 확인]의 나머지 테스트는 진행하지 않는다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [전자책 DRM Open Container Format (OCF) 3.0] | - 2.5.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - unzip |  |

#### 전자서명 문서의 스키마 검증

* **목적**

검증 대상 전자책의 signatures.xml 문서가 표준 스키마 문법을 준수하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령어를 사용하여 signatures.xml이 표준 스키마 문법을 준수하는지 확인한다. 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $xmllint -schema epub-container.xsd META-INF/signatures.xml | grep validates  META-INF/signatures.xml validate |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmllint | - epub-container.xsd  - w3c\_xml\_encryption.profile.xsd  - w3c\_xml\_signature\_profile.xsd |

#### 전자서명 알고리즘 확인

* **목적**

전자서명에 사용되는 알고리즘이 표준에서 지정한 "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"인지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령어를 사용하여 전자서명 알고리즘이 "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"인지 확인한다. 다른 알고리즘이 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:SignatureMethod/@Algorithm" META-INF/signatures.xml  http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### 정규화 알고리즘 확인

* **목적**

signatures.xml에 사용된 정규화 알고리즘이 표준에서 지정한 "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"인지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령어를 사용하여 정규화 알고리즘이 "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"인지 확인한다. 표시된 알고리즘이 다른 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:CanonicalizationMethod/@Algorithm" META-INF/signatures.xml  http://www.w3.org/TR/2001/REC-xml-c14n-20010315 |

1. 다음의 명령어를 사용하여 signedInfo의 Transform의 개수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Signature/x:SignedInfo/x:Reference/x:Transforms/x:Transform)" META-INF/signatures.xml  2 |

1. 다음의 명령어를 사용하여 정규화 알고리즘이 "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"인지 확인한다. 3. 에서 나온 값과 갯수가 맞지 않거나, 다른 알고리즘이 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:Reference/x:Transforms/x:Transform/@Algorithm" META-INF/signatures.xml  http://www.w3.org/TR/2001/REC-xml-c14n-20010315  http://www.w3.org/TR/2001/REC-xml-c14n-20010315 |

1. 다음의 명령어를 사용하여 Object의 Transform의 개수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Object/x:Manifest/x:Reference/x:Transforms/x:Transform)" META-INF/signatures.xml  2 |

1. 다음의 명령어를 사용하여 정규화 알고리즘이 "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"인지 확인한다. 5. 에서 나온 값과 갯수가 맞지 않거나, 다른 알고리즘이 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Object/x:Manifest/x:Reference/x:Transforms/x:Transform/@Algorithm" META-INF/signatures.xml  http://www.w3.org/TR/2001/REC-xml-c14n-20010315  http://www.w3.org/TR/2001/REC-xml-c14n-20010315 |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2.1, 4.3.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### 리소스의 해시 알고리즘 확인

* **목적**

signatures.xml에 사용된 리소스의 해시 알고리즘이 표준에서 지정된 "http://www.w3.org/2001/04/xmlenc#sha256"인지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령어로 DigestMethod 개수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Signature/x:SignedInfo/x:Reference/x:DigestMethod)" META-INF/signatures.xml  2 |

1. 다음의 명령어를 사용하여 해시 알고리즘이 "http://www.w3.org/2001/04/xmlenc#sha256"인지 확인한다. 나온 결과 개수가 위의 수와 맞지 않거나 다른 알고리즘이 표시된 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:Reference/x:DigestMethod/@Algorithm" META-INF/signatures.xml  http://www.w3.org/2001/04/xmlenc#sha256 http://www.w3.org/2001/04/xmlenc#sha256 |

1. 다음의 명령어로 DigestMethod 개수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Object/x:Manifest/x:Reference/x:DigestMethod)" META-INF/signatures.xml  13 |

1. 다음의 명령어를 사용하여 해시 알고리즘이 "http://www.w3.org/2001/04/xmlenc#sha256"인지 확인한다. 나온 결과 개수가 위의 수와 맞지 않거나 다른 알고리즘이 표시된 경우 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Object/x:Manifest/x:Reference/x:DigestMethod/@Algorithm" META-INF/signatures.xml  http://www.w3.org/2001/04/xmlenc#sha256 http://www.w3.org/2001/04/xmlenc#sha256 ... |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2.1, 4.3.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### 리소스 데이터의 해시값 검증

* **목적**

signatures.xml에 사용된 리소스의 해시 알고리즘이 표준에서 지정된 "http://www.w3.org/2001/04/xmlenc#sha256"인지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령문을 통해 해시된 파일의 경로와 해시된 값을 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -m "//x:Signature/x:Object/x:Manifest/x:Reference" -v @URI -o ' : ' -v x:DigestValue -n META-INF/signatures.xml  EPUB/c1.xhtml : 5cMgcTbPTQlFWDNv6wKP1pUzzb0O/dY+rdQWw2c5+8o= EPUB/c2.xhtml : QpSx7owlKIfGej6ipOYRcFFnhUsMd81XKw/q8RsCJxE= ... |

1. unzip 툴과 다음의 명령문을 사용하여 전자책 콘텐츠의 모든 리소스 파일을 추출한다. 추출과정에서 어떠한 에러 메시지가 표시되거나 2. 에서 구한 URI 경로에 파일이 없으면 이 검증은 실패이다.

|  |
| --- |
| $unzip tree.epub |

1. 등록된 모든 리스트에 대해서, 2. 항목에서 구한 DigestValue의 값과 다음의 명령문을 통해 구한 해시값을 비교한다. 일치하지 않으면 리소스 파일은 변조된 것이며, 이 검증은 실패이다.   
   만약 검사대상 Reference 항목에 Transforms 항목이 존재한다면 이 검증은 하지않고 대신 5. 항목을 검증한다.

|  |
| --- |
| $openssl dgst -sha256 -binary META-INF/encryption.xml | openssl base64  DiD35J8Wo/ogC2PeUp57E80Hog/RXlzx5c7V/kSeKO0= |

1. 해시값 검증 대상 파일의 Reference 영역에 Transforms/Transform의 @Algorithm이 “http://www.w3.org/TR/2001/REC-xml-c14n-20010315” 인 경우에는 다음의 명령문으로 검증한다. 만약 일치하지 않거나 어떠한 에러메시지가 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet c14n <(cat META-INF/signatures.xml) | openssl dgst -sha256 -binary | openssl base64  DiD35J8Wo/ogC2PeUp57E80Hog/RXlzx5c7V/kSeKO0= |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.3.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - openssl |  |

#### 전자서명 내의 해시값 검증

* **목적**

전자서명 내의 해시된 값에 대해 실제 해시값이 일치하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령문을 통해 refence의 갯수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Signature/x:Object/x:Manifest)" META-INF/signatures.xml  2 |

1. 다음의 명령어를 통해 URI의 값을 확인한다. 값의 첫문자가 #이 아니거나, 위의 값과 다른 갯수가 나오면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:Reference/@URI" META-INF/signatures.xml  #Manifest1  #Manifest2 |

1. 다음의 명령어를 통해 Object/Menifest Id 속성에 위의 URI값의 #이 제거된 값이 존재하는지 확인한다. 위의 URI값이 없으면 이 검증은 실패이다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:Object/x:Manifest/@Id" META-INF/signatures.xml  Manifest1  Manifest2 |

1. 모든 DigestValue에 대해서, 같은 URI&Id로 검색한 Object/Manifest의 해시값과 Referece/DigestValue의 값과 비교한다. 값이 다른 경우 이 검증은 실패이다.

|  |
| --- |
| $echo -e '1:' `xmlstarlet c14n <(xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -c '//x:Signature/x:Object/x:Manifest[@Id="Manifest1"]' META-INF/signatures.xml) | openssl dgst -sha256 -binary | openssl base64` "\n2:" `xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignedInfo/x:Reference[@URI='#Manifest1']/x:DigestValue" META-INF/signatures.xml`  1: 7mZC1OoKx0qOhBbWMiPacTW8MIcFbx9Dh7e+OUmAoWM=  2: 7mZC1OoKx0qOhBbWMiPacTW8MIcFbx9Dh7e+OUmAoWM= |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet |  |

#### 인증서 체인 검증

* **목적**

전자서명을 한 인증서가 정상적인 발행 기관으로부터 발급된 인증서인지 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 다음의 명령어를 통해 인증서의 개수를 확인한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "count(//x:Signature/x:KeyInfo/x:X509Data)" META-INF/signatures.xml  3 |

1. 위의 값만큼 다음의 명령어를 수를 증가시키며 실행하여 인증서를 추출한다.

|  |
| --- |
| $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -m "//x:Signature/x:KeyInfo/x:X509Data[1]" -v x:X509Certificate -n META-INF/signatures.xml | openssl base64 -d | openssl x509 -inform DER -out 1.pem  $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -m "//x:Signature/x:KeyInfo/x:X509Data[2]" -v x:X509Certificate -n META-INF/signatures.xml | openssl base64 -d | openssl x509 -inform DER -out 2.pem  $xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -m "//x:Signature/x:KeyInfo/x:X509Data[3]" -v x:X509Certificate -n META-INF/signatures.xml | openssl base64 -d | openssl x509 -inform DER -out 3.pem |

1. 다음의 명령문은 인증서의 issuer와 subject의 DN값을 보여준다. 명령문을 통해 두 값이 같은 Root 인증서를 찾는다. 만약 어떠한 에러 메시지라도 표시된다면 유효하지 않은 인증서이며, 이 검증은 실패이다.

|  |
| --- |
| $openssl x509 -in 1.pem -subject -issuer -noout  issuer= /dnQualifier=gymaNNiH8RTruzGvaPeIBMQDXog=/CN=inc auth/OU=ra/O=au  subject= /dnQualifier=vkHUvyFj1z28vPzHKKLyQTLuec0=/CN=device/OU=ra/O=au |

1. 검증항목 4. 에서 구한 Root 인증서가 전자책 DRM 표준에서 지정한 신뢰할 수 있는 CA에서 발급된 Root 인증서인지 확인한다. 만약 그렇지 않다면 이 검증은 실패이다.
2. Root 인증서의 subject DM값과 일치하는 issuer DN값을 가지는 인증서를 찾고, 나머지 인증서들에 대해 재귀적으로 반복한다. 마지막 Leaf 인증서까지 일치하지 않는 인증서가 나올 경우 이 검증은 실패한다.
3. 인증서들의 발급순서를 가지고 인증서 체인을 검증한다. Root 인증서는 맨 마지막에 들어가며, 나머지 인증서들은 cat 명령 뒤로 상위 인증서부터 입력한다. 만약 어떤 에러 메시지라도 표시된다면 이 검증은 실패이다.

|  |
| --- |
| $cat 2.pem 3.pem | openssl verify -CAfile 1.pem  stdin: OK |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명  - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제4부 인증서 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - openssl |  |

#### 전자서명 확인

* **목적**

signaturex.xml 문서의 전자서명의 유효성을 검증한다.

* **사전조건**
* **검증절차**

1. 검증 대상 전자책 DRM 콘텐츠에서 추출한 signatures.xml을 준비한다.
2. 2.1.5.8. 검증 항목에서 추출한 Leaf 인증서를 준비한다.
3. 다음의 명령문을 통해 SignedInfo값에 대한 전자서명을 확인한다. Verified OK가 표시되지 않거나 어떠한 에러지가 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $openssl dgst -sha256 -verify <(openssl x509 -in 3.pem -pubkey -noout) -signature <(xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -v "//x:Signature/x:SignatureValue" META-INF/signatures.xml | openssl base64 -d) <(xmlstarlet c14n <(xmlstarlet sel -N x="http://www.w3.org/2000/09/xmldsig#" -t -c "//x:Signature/x:SignedInfo" META-INF/signatures.xml))  Verified OK |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - xmlstarlet  - openssl |  |

### 인증서 검증

#### 인증서의 존재 확인

* **목적**

signatures.xml과 license.lcpl의 인증서 체인이 존재하는지 확인한다.

* **검증절차**

1. 검증 대상 전자책에 포함된 license.lcpl과 signatures.xml에서 추출한 인증서들을 준비한다.
2. 2.1.6. 인증서 검증 시나리오는 하나의 체인에 대해 검사가 진행되어야 하며, license.lcpl과 signatures.xml이 함께 있는 경우 2.1.6. 인증서 검증의 테스트 항목들을 별도로 수행하여야 한다.
3. license.lcpl / signatures.xml이 존재하지 않는다면 이 테스트는 진행하지 않고, signatures.xml이 있는 경우 2.1.5.8. 검증 항목에서 인증서 추출에 실패하였다면 이 검증은 실패이다.
4. license.lcpl의 인증서 체인은 jq툴과 다음의 명령문들을 통해 추출한다. leaf인증서는 ‘.signature.certificate’ 에서, 상위인증서들은 ‘.signature.ca\_certificates’ 의 길이를 확인한후 차례대로 추출한다. 만약 인증서 추출과정에서 어떠한 에러 메시지가 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $jq -r '.signature.certificate' META-INF/license.lcpl | openssl base64 -d -A -nopad | openssl x509 -inform DER -out leaf.pem |

|  |
| --- |
| $jq -r '.signature.ca\_certificates| length' META-INF/license.lcpl  1  $ jq -r '.signature.ca\_certificates| length[0]' META-INF/license.lcpl | openssl base64 -d -A -nopad | openssl x509 -inform DER -out signer1.pem |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명  - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.2.3.  - 4.8. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| jq |  |

#### 인증서 표현방식 확인

* **목적**

인증서에 대해서 ITU-T.509 v3 표준에 기술된 ASN.1 DER 방식의 표현을 준수하는지, 저장 포맷으로 DER 또는 PEM 방식을 사용하는지 검증한다.

* **사전조건**
* **검증절차**

1. 모든 인증서에 대해 다음의 명령문을 통해 파일 형식을 검증한다. 만약 인증서가 이진 형태라면 -inform DER 옵션을 추가하여 검증을 수행한다.

|  |
| --- |
| $openssl x509 -text -noout -in <certificate\_filename>  $openssl x509 -text -noout -inform DER -in <certificate\_filename> |

1. 어떠한 오류가 표시된다면 이 검증은 실패이다.
2. 다음의 명령문을 통해 출력된 결과의 version 항목이 3(0x2)이 아니라면 이 검증은 실패이다.

|  |
| --- |
| $ openssl x509 -text -noout -in cert.pem | grep Version  Version: 3 (0x2) ... |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | -4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.2.1 |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 시리얼번호 양식 확인

* **목적**

인증서의 시리얼번호 항목에 8바이트보다 크지 않은 양의 정수를 사용해야하는 표준 준수여부를 검증한다.

* **사전조건**
* **검증절차**

1. 2.1.6.2. 항목의 명령문을 통해 시리얼 넘버 항목의 존재 및 표준을 준수하는지를 확인한다. serial number 항목이 존재하지 않거나 64bit보다 크지 않은 양의 정수가 아닐 경우 이 검증은 실패이다.

|  |
| --- |
| $openssl x509 -text -noout -in <certificate\_filename> |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 전자서명 알고리즘 확인

* **목적**

인증서의 알고리즘 항목이 sha256WithRSAEncryption 이어야 하는 표준 준수여부를 검증한다.

* **사전조건**
* **검증절차**

1. 다음의 명령문을 통해 인증서 전자서명 알고리즘 표준을 준수하는지를 확인한다. 알고리즘이 "sha256WithRSAEncryption"이 아니거나, 어떠한 에러 메시지가 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $openssl x509 -text -noout -in <certificate\_filename> | grep "Signature Algorithm"  Signature Algorithm: sha256WithRSAEncryption  Signature Algorithm: sha256WithRSAEncryption |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 인증서의 체인 검증 확인

* **목적**

사용된 인증서 체인의 유효함을 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서 체인을 준비한다.
2. 인증서들의 발급순서를 가지고 인증서 체인을 검증한다. Root 인증서는 맨 마지막에 들어가며, 나머지 인증서들은 cat 명령 뒤로 상위 인증서부터 입력한다. stdin: OK가 표시되지 않거나, 어떠한 에러 메시지가 표시되는 경우 이 검증은 실패이다.

|  |
| --- |
| $cat 2.pem 3.pem | openssl verify -CAfile 1.pem  stdin: OK |

1. 만약 라이선스 문서의 경우 2.1.4.8. 항목, 전자서명 문서의 경우 2.1.5.8. 항목의 테스트 결과과 있다면, 그 결과에 그대로 사용하는 것도 가능하다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [ITU-T X.509] |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 인증서 공개키의 무인 검증

* **목적**

인증서의 issuer 및 subject 정보에 해당 인증서 공개키의 무인이 삽입되었는지 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서 체인을 준비한다.
2. 다음의 명령문을 통해 인증서의 issuer DN 값과 subject DN 값을 구한다.

|  |
| --- |
| $openssl x509 -issuer -subject -in cert.pem -text –noout  issuer= /dnQualifier=gymaNNiH8RTruzGvaPeIBMQDXog=/CN=inc auth/OU=ra/O=au  subject= /dnQualifier=vkHUvyFj1z28vPzHKKLyQTLuec0=/CN=device/OU=ra/O=au |

1. 다음의 명령문을 통해 인증서의 subject DN의 dnQualifier 값을 구한다.

|  |
| --- |
| $openssl x509 -pubkey -noout -in cert.pem -noout | openssl base64 -d | dd bs=1 skip=24 2>null | openssl sha1 -binary | openssl base64  vkHUvyFj1z28vPzHKKLyQTLuec0= |

1. root 인증서의 경우 issuer DN의 dnQualifier도 같은 값이다. 나머지 인증서의 경우는 issuer 인증서의 dnQualifier값과 issuer DN의 dnQualifier 값과 비교한다.
2. DN 값에 dnQualifier 항목이 존재하지 않거나, 계산된 값과 다른 dnQualifier값을 가지는 인증서가 존재하는 경우 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.4, 4.2.6. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### issuer, subject 정보 확인

* **목적**

인증서의 issuer 및 subject 정보에 조직이름(OrganizationName, O), 조직부서이름(OrganizationUnit, OU), 역할이름(CommonName, CN)이 명시되어 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서들을 준비한다.
2. 다음의 명령문을 통해 인증서의 issuer DN 값과 subject DN 값을 구한다.

|  |
| --- |
| $openssl x509 -issuer -subject -in cert.pem -text –noout  issuer= /dnQualifier=gymaNNiH8RTruzGvaPeIBMQDXog=/CN=inc auth/OU=ra/O=au  subject= /dnQualifier=vkHUvyFj1z28vPzHKKLyQTLuec0=/CN=device/OU=ra/O=au |

1. DN 값에 'O', 'OU', 'CN' 항목이 존재함을 확인하고, 값의 내용을 확인한다.  
   - O : 인증서를 발급한 기관 또는 회사이름  
   - OU : 인증서를 발급한 기관 또는 회사의 부서이름  
   - CN : 발급한 기기에 대한 식별자  
   해당되는 항목이 2개 이상이거나 존재하지 않을 경우 또는 값이 없을 경우 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.4, 4.2.6. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 기간 검증 확인

* **목적**

인증서의 기간 표기 방법이 표준을 준수하는지 검증한다. 기간 표기법은 2050년 이후는 GMT 방식, 그 이전은 UTC 방식이 표준이다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서를 준비한다.
2. 다음의 명령문을 통해 표시된 기간 정보의 값이 2050년 이전인 경우 UTC(YYMMDDhhmmssZ), 2050년 이후인 경우 GMT(YYYYMMDDhhmmssZ) 방식인지 확인한다.

|  |
| --- |
| 165:d=3 hl=2 l= 13 prim: UTCTIME :160801081835Z 180:d=3 hl=2 l= 15 prim: GENERALIZEDTIME :21160708081835Z |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 공개키 비도 확인

* **목적**

인증서에 포함된 공개키 정보의 내용이 표준을 준수하는지 검증한다.

* **검증절차**

1. 이전 테스트에서 추출한 인증서를 준비한다.
2. 다음의 명령문을 통해 Public Key Algorithm 항목의 값이 rsaEncryption인지 확인한다.

|  |
| --- |
| $openssl x509 -noout -in cert.pem -text | grep "Public Key Algorithm:"  Public Key Algorithm: rsaEncryption |

1. 다음의 명령문을 통해 Public-key 항목의 값이 (2048 bit)인지 확인한다.

|  |
| --- |
| $openssl x509 -noout -in cert.pem -text | grep Public-Key:  Public-Key: (2048 bit) |

1. Exponent 항목의 값이 65537 (0x10001)인지 확인한다.

|  |
| --- |
| $openssl x509 -noout -in cert.pem -text | grep Exponent  Exponent: 65537 (0x10001) |

1. 모든 인증서에 대해 일치되지 않는 값이 표시되거나 어떤 오류 메시지가 나타나면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.7. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### AuthorityKey 식별자 정보 확인

* **목적**

인증서의 AuthorityKey 식별자 정보가 표준에 맞게 존재하는지 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서를 준비한다.
2. 다음의 명령문을 통해 인증서의 정보를 확인한다.

|  |
| --- |
| $openssl x509 -in cert.pem -text -noout |

1. 표시된 인증서의 내용에서 X509v3 extensions 이하의 내용중 X509v3 Authority Key Identifier 항목이 존재하는지 확인한다. 만약 존재하지 않는 경우 이 검증은 실패이다.
2. Authority Key Identifier의 keyid값 항목을 확인한다. 항목이 존재하지 않는 경우 이 검증은 실패이다.
3. 검증대상 인증서의 상위인증서와 다음의 명령문을 통해 상위인증서의 Key Identifier의 값을 확인한다. 이 값과 3.에서 확인한Authority Key Identifier 의 key Id값과 일치하지 않는 경우 이 검증은 실패이다.

|  |
| --- |
| $openssl x509 -pubkey -noout -in cert.pem -noout | openssl base64 -d | dd bs=1 skip=24 2>null | openssl sha1 -binary | xxd -p -u  BE41D4BF2163D73DBCBCFCC728A2F24132EE79CD |

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.10. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 상위 인증서 정보 검증

* **목적**

상위 인증서의 extensions 항목 정보에 대해 표준을 준수하는지 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서를 준비한다.
2. 다음의 명령문을 통해 인증서의 정보를 확인한다. 확인의 대상은 leaf인증서를 제외한 인증서들이다.

|  |
| --- |
| $openssl x509 -in cert.pem -text -noout |

1. 표시된 인증서의 내용에서 X509v3 extensions 이하의 내용중 다음을 확인한다.
   1. Key Usage 항목의 값이 KeyCertSign임을 확인한다.
   2. Basic Constraints 항목에 대해, critical 옵션이 포함되어 있고, CA:TRUE임을 확인한다.
   3. Basic Constraints 항목에 대해, pathlen이 존재하는지 확인하고, 그 값이 0또는 양의 정수임을 확인한다.
2. 위의 내용중 다르게 적용된 항목이 있다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.10. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

#### 말단 인증서 정보 검증

* **목적**

말단(leaf) 인증서의 extensions 항목 정보에 대해 표준을 준수하는지 검증한다.

* **사전조건**
* **검증절차**

1. 이전 테스트에서 추출한 인증서를 준비한다.
2. 다음의 명령문을 통해 인증서의 정보를 확인한다. 확인의 대상은 leaf인증서이다.

|  |
| --- |
| $openssl x509 -in cert.pem -text -noout |

1. 표시된 인증서의 내용에서 X509v3 extensions 이하의 내용중 다음을 확인한다.
   1. Key Usage 항목의 값이 DigitalSignature임을 확인한다.
   2. Basic Constraints 항목에 대해, CA:FALSE임을 확인한다. critical 옵션은 선택적으로 사용될 수 있다.
   3. Basic Constraints 항목에 대해, 만약 pathlen이 존재한다면 그 값이 0인지 확인한다.
2. 위의 내용중 다르게 적용된 항목이 있다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 제4부 인증서  - [ITU-T X.509] | - 4.2.10. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
| - openssl |  |

## 기능 정합성 검증

### 기본 문서 검증

#### 정합 데이터 검증

##### 전자서명 문서가 없는 전자책 열람 확인

* **목적**

signatures.xml이 없는 DRM 콘텐츠에 대해 정상 열람이 가능한지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(no\_sig.epub)를 입력한다.
2. 검증자료는 signatures.xml 문서가 없는 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | no\_sig.epub |

#### 비정합 데이터 검증

##### 라이선스 문서가 없는 전자책 오류 처리

* **목적**

license.lcpl 문서가 삭제된 전자책에 대해서 정상적인 오류 처리를 하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(no\_license.epub)를 입력한다.
2. 검증자료는 license.lcpl 문서가 없는 전자책 DRM 콘텐츠이다. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [전자책 DRM Open Container Format(OCF) 3.0] | - 2.5.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | no\_license.epub |

##### 암호화 문서가 없는 전자책 오류 처리

* **목적**

encryption.xml 문서가 삭제된 전자책에 대해서 정상적인 오류 처리를 하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(no\_enc\_doc.epub)를 입력한다.
2. 검증자료는 encryption.xml 문서가 없는 전자책 DRM 콘텐츠이다.검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [전자책 DRM Open Container Format(OCF) 3.0] | - 2.5.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | no\_enc\_doc.epub |

### 복호화 처리 기능 검증

#### 정합 데이터 검증

##### 암호화된 리소스를 포함한 전자책 열람 확인

* **목적**

검사대상 전자책 처리기가 암호화된 리소스를 포함한 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_res.epub)를 입력한다.
2. 검증자료는 암호화된 리소스를 포함하며 "http://kats.org/lcp/profile-1.0" 프로파일을 준수하는 라이선스 문서를 가지는 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_res.epub |

##### 압축 암호화된 리소스를 포함한 전자책 열람 확인

* **목적**

검사대상 전자책 처리기가 압축되고 암호화된 리소스를 포함한 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(comp\_enc\_res.epub)를 입력한다.
2. 검증자료는 압축이되고 암호화된 리소스를 포함하는 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | comp\_enc\_res.epub |

##### 압축/비압축 암호화된 리소스들을 포함한 전자책 열람 확인

* **목적**

검사대상 전자책 처리기가 비압축 암호화된 리소스와 압축 암호화된 리소스를 모두 포함한 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(comp\_enc\_res2.epub)를 입력한다.
2. 검증자료는 압축되고 암호화된 리소스와 무압축으로 암호화된 리소스가 다 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | comp\_enc\_res2.epub |

##### 평문/암호화된 리소스를 포함한 전자책 열람 확인

* **목적**

검사대상 전자책 처리기가 암호화된 리소스와 암호화되지 않은 리소스를 모두 포함한 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_res2.epub)를 입력한다.
2. 검증자료는 암호화된 리소스와 평문 리소스가 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_res2.epub |

##### 암호화된 리소스 파일이 없는 암호화문서를 포함하는 전자책 열람 확인

* **목적**

검사대상 전자책 처리기가 encryption.xml과 license.lcpl이 모두 존재하지만 모든 콘텐츠가 암호화되지 않은 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(no\_enc\_res.epub)를 입력한다.
2. 검증자료는 encryption.xml과 license.lcpl이 정상적으로 존재하지만 encryption.xml의 암호화된 리소스의 리스트가 없는 평문 전자책이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | no\_enc\_res.epub |

##### User key 방식의 DRM이 적용된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 User Key 방식의 전자책 DRM 콘텐츠를 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(userkey\_type.epub)를 입력한다.
2. 검증자료는 password로 DRM 패키징된 User key 방식의 전자책 DRM 콘텐츠이다.
3. 오픈시 입력할 비밀번호는 "password"이다.
4. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | userkey\_type.epub |

##### 콘텐츠 암호화 알고리즘이 AES-256-GCM 방식인 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 AES-256-GCM 알고리즘으로 리소스를 암호화한 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(gcm256.epub)를 입력한다.
2. 검증자료는 콘텐츠 암호화 알고리즘이 AES-256-GCM으로 암호화된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 8.3 |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | gcm256.epub |

##### 암호화된 이미지 파일이 포함된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 암호화된 이미지 리소스 파일이 포함된 전자책을 열람할 수 있는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_img.epub)를 입력한다.
2. 검증자료는 암호화된 이미지 리소스 파일이 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_img.epub |

##### 암호화된 음악 파일이 포함된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 암호화된 음악 리소스 파일이 포함된 전자책을 열람할 수 있는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_mp3.epub)를 입력한다.
2. 검증자료는 암호화된 음악 리소스 파일이 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_mp3.epub |

##### 암호화된 스크립트 파일이 포함된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 암호화된 javascript 파일이 포함된 전자책을 열람할 수 있는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_script.epub)를 입력한다.
2. 검증자료는 암호화된 javascript 리소스 파일이 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_script.epub |

##### 암호화된 동영상 파일이 포함된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 암호화된 동영상 파일이 포함된 전자책을 열람할 수 있는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(enc\_mp4.epub)를 입력한다.
2. 검증자료는 암호화된 동영상 리소스 파일이 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | enc\_mp4.epub |

#### 비정합 데이터 검증

##### 올바르지 않은 암호화 문서를 포함한 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 표준 스키마를 준수하지 않는 encryption.xml 문서가 포함된 전자책에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_enc1.epub, invalid\_enc2.epub)를 입력한다.
2. 검증자료는 표준 스키마를 준수하지 않는 encryption.xml 문서를 포함하는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제2부 암호화 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_enc1.epub  invalid\_enc2.epub |

##### Key의 URI, type 정보가 비표준인 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 표준이 아닌 콘텐츠 암호화 키의 URI와 type을 사용하는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_key\_info1.epub, invalid\_key\_info2.epub)를 입력한다.
2. 검증자료는 encryption.xml 문서의 KeyInfo/RetrievalMethod의 @URI가 license.lcpl#/encryption/content\_key가 아니거나 @Type이 "http://readium.org/2014/01/lcp#EncryptedContentKey"가 아닌 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제2부 암호화 | - 4.4.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_key\_info1.epub invalid\_key\_info2.epub |

##### 라이선스 문서의 content\_key 영역에 ECK가 없는 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 license.lcpl의 encryption 영역에 user\_key 또는 device Key를 포함하지 않는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(no\_eck.epub)를 입력한다.
2. 검증자료는 license.lcpl 문서의 encryption 항목에 user\_key 또는 device\_key를 포함하지 않는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4 |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | no\_eck.epub |

##### 검증대상 기기의 device key가 아닌 key로 암호화된 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 device key로 암호화되지 않은 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_dev\_key.epub)를 입력한다.
2. 검증자료는 검사대상 전자책 처리기의 디바이스 키가 아닌 다른키를 사용해 암호화한 리소스 파일을 가지는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4.3. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_dev\_key.epub |

##### User key 방식의 전자책에 틀린 비밀번호를 입력하는 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 user key로 암호화된 전자책에 대해 패스워드가 틀렸을 경우 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(userkey\_type.epub)를 입력한다.
2. 검증자료는 password로 DRM 패키징된 User key 방식의 전자책 DRM 콘텐츠이다.
3. 오픈시 입력할 비밀번호는 "password"이다.
4. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 - 제6부 라이선스 | - 4.4.4. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | userkey\_type.epub |

##### 암호화문서에 등록된 암호화 대상 리소스가 존재하는 않는 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 encryption.xml 문서에 암호화된 리소스의 정보가 있지만, 그 파일이 전자책 컨테이너에 존재하지 않는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(not\_ex\_res.epub)를 입력한다.
2. 검증자료는 encryption.xml 문서에 암호화된 리소스 파일이 있지만, 그 파일 리소스가 없는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 정상적으로 열람 가능한지 확인하되, 해당 리소스의 페이지에서 정상적인 오류메시지를 출력해야한다. (하거나 다음 페이지를 표시해야 한다.) 만약 비정상적인 동작이나 종료된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | not\_ex\_res.epub |

##### 암호화된 리소스 파일이 손상된 전자책의 오류 처리

* **목적**

검증대상 전자책 처리기가 암호화된 파일이 훼손되어 복호화되지 않는 리소스에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(damaged\_res.epub)를 입력한다.
2. 검증자료는 encryption.xml 문서에 암호화된 리소스 파일이 암호화 내용이 손상된 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 정상적으로 열람 가능한지 확인하되, 해당 리소스의 페이지에서 정상적인 오류메시지를 출력해야 한다. 만약 비정상적인 동작이나 종료된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
|  |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | damaged\_res.epub |

### 전자서명 확인 기능 검증

#### 정합 데이터 검증

##### 전자서명 문서가 포함된 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 전자서명 문서가 포함된 전자책 DRM 콘텐츠에 대해 정상적인 열람이 가능한지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(with\_sig.epub)를 입력한다.
2. 검증자료는 signatures.xml 파일이 포함된 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을 때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - [전자책 DRM Open Container Format(OCF) 3.0] | - 2.5.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | with\_sig.epub |

#### 비정합 데이터 검증

##### 올바르지 않은 전자서명 문서를 포함한 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 표준 스키마를 준수하지 않는 signatures.xml 문서를 포함하는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_sig1.epub, invalid\_sig2.epub)를 입력한다.
2. 검증자료는 표준 스키마를 준수하지 않는 signatures.xml 문서를 포함하는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_sig1.epub  invalid\_sig2.epub |

##### 리소스의 해시값이 유효하지 않은 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 해시값으로 등록된 리소스 파일이 변형 및 훼손된 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_res\_hash.epub)를 입력한다.
2. 검증자료는 signatures.xml 문서에 등록된 리소스 파일의 해쉬값(Object/Manifest/Reference/DigestValue)이 실제 리소스의 해쉬값과 다른 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.3.1. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_res\_hash.epub |

##### 인증서 체인이 유효하지 않은 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 유효하지 않은 인증서 체인으로 signatures.xml 문서를 전자서명한 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_sig\_chain1.epub, invalid\_sig\_chain2.epub, invalid\_sig\_chain3.epub)를 입력한다.
2. 검증자료는 유효하지않은 인증서 체인으로 signatures.xml 문서가 전자서명된 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제4부 인증서  - [ITU-T X.509] |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_sig\_chain1.epub  invalid\_sig\_chain2.epub  invalid\_sig\_chain3.epub |

##### 전자서명이 유효하지 않은 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 전자서명이 훼손된 signatures.xml을 포함하는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_signature.epub)를 입력한다.
2. 검증자료는 signatures.xml 문서의 내용이 무단으로 수정된 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제3부 전자서명 | - 4.2. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_signature.epub |

### 라이선스 확인 기능 검증

#### 정합 데이터 검증

##### 기간 권한이 유효한 전자책 열람 확인

* **목적**

검증대상 전자책 처리기가 검사일 기준 유효한 기간 권한을 가진 전자책을 열람할 수 있는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(valid\_license\_period.epub)를 입력한다.
2. 검증자료는 기간 권한이 검사일에 유효한 전자책 DRM 콘텐츠이다.
3. 전자책을 오픈하고, 페이지를 이동시켰을때 정상적인 콘텐츠가 표시되는지 확인한다. 전자책이 오픈되지 않는다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | valid\_license\_period.epub |

##### copy 권한 제어 확인

* **목적**

검증대상 전자책 처리기가 copy 권한에 대해 제어가 가능한지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기가 라이선스 권한의 copy 권한의 제어 기능이 포함된 전자책 처리기인지 확인한다. 만약 그렇지 않다면 이 검증은 수행하지 않는다.
2. 검증대상 전자책 처리기와 검증자료를 준비한다.
   1. copy 리미트가 무제한인 전자책(unlimited\_copy\_right.epub)의 copy 동작을 확인한다. 만약 copy 기능이 동작하지 않는다면 이 검증은 실패이다.
   2. copy 리미트가 100인 전자책(limited\_copy\_right\_100.epub)의 copy 동작을 확인하고, 100 초과의 copy 기능을 수행시 적절한 에러 메시지를 출력하는지 확인한다. 만약 100이하의 size의 copy기능이 동작하지 않거나, 100 초과의 size의 copy가 가능하다면 이 검증은 실패이다.
   3. copy 리미트가 0인 전자책(limited\_copy\_right\_0.epub)의 copy 동작 불가를 확인하고 적절한 에러 메시지를 출력하는지 확인한다. 만약 copy 기능이 동작한다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | unlimited\_copy\_right.epub  limited\_copy\_right\_100.epub  limited\_copy\_right\_0.epub |

##### print 권한 제어 확인

* **목적**

검증대상 전자책 처리기가 print 권한에 대해 제어가 가능한지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기가 라이선스 권한의 print 권한의 제어 기능이 포함된 전자책 처리기인지 확인한다. 만약 그렇지 않다면 이 검증은 수행하지 않는다.
2. 검증대상 전자책 처리기와 검증자료를 준비한다.
   1. print 리미트가 무제한인 전자책(unlimited\_print\_right.epub)의 print 동작을 확인한다. 만약 print 기능이 동작하지 않는다면 이 검증은 실패이다.
   2. print 리미트가 2인 전자책(limited\_print\_right\_2.epub)의 print 동작을 확인하고, 2 초과의 print 기능을 수행시 적절한 에러 메시지를 출력하는지 확인한다. 만약 2 이하의 size의 print 기능이 동작하지 않거나, 2 초과의 size의 print가 가능하다면 이 검증은 실패이다.
   3. print 리미트가 0인 전자책(limited\_print\_right\_0.epub)의 print 동작 불가를 확인하고 적절한 에러 메시지를 출력하는지 확인한다. 만약 print 기능이 동작한다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | unlimited\_print\_right.epub  limited\_print\_right\_2.epub  limited\_print\_right\_0.epub |

#### 비정합 데이터 검증

##### 올바르지 않은 라이선스 문서를 포함한 전자책 오류 처리

* **목적**

검증대상 전자책 처리기가 표준 스키마를 준수하지 않는 license.lcpl 문서를 포함하는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_license1.epub, invalid\_license2.epub)를 입력한다.
2. 검증자료는 표준 스키마를 준수하지 않는 license.lcpl 문서를 포함하는 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_license1.epub  invalid\_license2.epub |

##### 인증서 체인이 유효하지 않은 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 유효하지 않은 인증서 체인으로 license.lcpl 문서를 전자서명한 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_sig\_chain\_lic.epub)를 입력한다.
2. 검증자료는 유효하지 않은 인증서 체인으로 license.lcpl 문서가 전자서명된 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제4부 인증서  - [ITU-T X.509] |  |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_sig\_chain\_lic.epub |

##### 전자서명이 유효하지 않은 경우의 오류 처리

* **목적**

검증대상 전자책 처리기가 전자서명이 훼손된 license.lcpl 문서를 포함하는 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(invalid\_sig\_value\_lic.epub)를 입력한다.
2. 검증자료는 license.lcpl 문서의 내용이 무단으로 수정된 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 7.6 |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | invalid\_sig\_value\_lic.epub |

##### 라이선스 문서의 시작일이 검사일 이후인 전자책의 오류 처리

* **목적**

검증대상 전자책 처리기가 검사일이 시작일보다 앞선 기간 권한을 가진 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* **사전조건**
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(before\_start\_period.epub)를 입력한다.
2. 검증자료는 라이선스 문서의 시작일(start)이 검사일보다 늦은 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | before\_start\_period.epub |

##### 라이선스 문서의 종료일이 검사일 이전인 전자책의 오류 처리

* **목적**

검증대상 전자책 처리기가 검사일이 종료일보다 늦은 기간 권한을 가진 전자책 DRM 콘텐츠에 대해 정상적인 오류 처리를 하는지 검증한다.

* 사전조건
* **검증절차**

1. 검증대상 전자책 처리기를 준비하고, 검증자료(after\_end\_period.epub)를 입력한다.
2. 검증자료는 검사일이 라이선스 문서의 종료일(rights/end)보다 늦은 전자책 DRM 콘텐츠이다.
3. 검증대상 전자책 처리기가 오류 메시지를 출력하며 열람 불가 상태인지 확인한다. 만약 전자책이 오픈된다면 이 검증은 실패이다.

* **참조 표준 문서**

|  |  |
| --- | --- |
| 참조문서 번호 | 참조문서 장 |
| - 한국산업표준 EPUB DRM 상호운용 기술 명세 – 제6부 라이선스 | - 4.5. |

* **검증환경**

|  |  |
| --- | --- |
| 검증 도구 | 검증 자료 |
|  | after\_end\_period.epub |

# 정합성 검증 방법

## 데이터 정합성 검증

데이터 정합성 검증은 P전자책 DRM 표준에 따라 제작된 전자책 DRM 콘텐츠에 대하여 DRM을 위한 데이터 검증을 실시하는 것으로 검증 대상 전자책 DRM 콘텐츠를 2.1(데이터 정합성 검증)의 형식에 따라 검증한다. 검증은 전자책 DRM 콘텐츠 내부에 존재하는 encryption.xml과 license.lcpl, signatures.xml의 유효성 검증을 기본으로 하고 있으며 각 검증 소항목에 대한 검증 내용과 검증 결과를 보고서 형태로 작성하여 결과로 제시하여야 한다.

## 기능 정합성 검증

기능 정합성 검증은 전자책 DRM 표준에 따라 제작된 EPUB관련 툴에 대하여 DRM 처리를 위한 기능 검증을 실시하는 것으로 검증 대상 툴을 2.2(기능 정합성 검증)의 형식에 따라 검증한다. 검증은 검증 대상 뷰어 시스템에 대하여 각 기능을 검증하기 위한 검증 자료를 입력하여 검증을 수행하여야 하며, 각 검증 소항목에 대한 검증 내용과 검증 결과를 보고서 형태로 작성하여 결과로 제시하여야 한다.

기능 검증을 위한 검증 자료는 검증 전에 검증 대상 툴에서 사용하는 인증서를 기반으로 제작이 이루어져야 하는데, 검증 자료 제작에 대한 과정은 4.2(검증자료 제작방법) 항목을 참조하기 바란다.

## 보고서

데이터, 기능, 프로토콜 정합성 검증에 대한 검증 보고서는 각각의 검증 소항목에 따라 아래와 같은 형식의 보고서를 작성하여야 한다.

|  |  |  |
| --- | --- | --- |
| 항목 | 내용 | 예 |
| 검증 번호 | 검증 시나리오 번호 | 2.1.1.1. |
| 검증 종류 | 검증 시나리오 종류 | 데이터 정합성 검증 |
| 검증 제목 | 검증 시나리오 제목 | 암호화 문서의 존재 확인 |
| 검증 목적 | 검증 시나리오 목적 | 검증 대상 전자책에 암호화 문서가 “META-INF/encryption.xml” 경로에 존재하는지 검증한다. |
| 기준 표준 | 정합성 검증에 사용된 표준 문서 | [전자책 DRM Open Container Format(OCF) 3.0] 2.5.2. |
| 검증 결과 | 검증 결과 | 성공 |
| 상세 결과 | 검증 단계별 결과 | 1. encryption.xml 추출 : 성공 2. encryption.xml의 charset 확인 : 성공 |
| 검증 자료 | 검증에 사용된 자료 | no\_sig.epub |

# 정합성 검증을 위한 자료 및 도구

## 개요

전자책 DRM 콘텐츠 및 DRM 뷰어시스템의 표준정합성 검사를 위해서는 검증도구 및 검증자료들이 필요하다. 검증도구들은 누구나 이용이 가능한 오픈소스 도구들과 본 표준을 위해 작성된 프로그램과 스크립트가 제공된다.

## 검증도구

표준정합성 검사에는 데이터 검증에 여러 검증도구들이 사용된다.

### 검증 툴

|  |  |  |  |
| --- | --- | --- | --- |
| 종류 | 번호 | 검증도구 | 설명 |
| public  tools | 4.2.2.1. | json-schema-validator-master-lib.jar | JSON 스키마 validator. open source.  https://github.com/fge/json-schema-validator |
| 4.2.2.2. | jq | JSON parser. open source.  http://stedolan.github.io/jq/ |
| 4.2.2.3. | OpenSSL | 암복호화 tool. open source.  https://www.openssl.org/ |
| 4.2.2.4. | unzip | 압축해제 tool.  http://gnuwin32.sourceforge.net/packages/unzip.htm |
| 4.2.2.5. | xmllint | XML validator. open source.  http://xmlsoft.org/ |
| 4.2.2.6. | xmlstartlet | XML parser. open source.  http://xmlstar.sourceforge.net/ |
| 4.2.2.7. | bcprov-jdk15on-152.jar | Bouncycastle 암복호화 라이브러리(MIT 라이선스).  https://www.bouncycastle.org/latest\_releases.html |
| 4.2.2.8. | gson-2.3.1.jar | JSON parser. open source.  https://github.com/google/gson |
| 4.2.2.9. | Python 3.5 | Python interpreter.  https://www.python.org/ |
| shell  scripts | 4.2.3.1. | compression\_checker | 미디어 타입 리소스의 압축 여부를 확인하는 스크립트. |
| 4.2.3.2. | key\_checker | encryption/user\_key/key\_check의 값을 확인하는 스크립트. |
| 4.2.3.3. | resource\_decrypter | 암호화된 리소스를 복호화하는 스크립트. |
| 4.2.3.4. | user\_info\_decrypter | 암호화된 유저 정보를 복호화하는 스크립트. |
| program | 4.2.4.1. | signatureValidator.java | 라이선스 문서의 전자서명을 확인하는 자바 프로그램. |
| 4.2.4.2. | u\_checker.py | 라이선스 문서의 key 중복을 검사하는 파이썬 프로그램. |

### 검증용 스키마 파일

|  |  |  |
| --- | --- | --- |
| 번호 | 스키마 파일 | 설명 |
| 4.2.2.1. | epub-container.xsd | 전자책 컨테이너 포맷 스키마 |
| 4.2.2.2. | w3c\_xml\_encryption\_profile.xsd | 암호화 문서 스키마 |
| 4.2.2.3. | w3c\_xml\_signature\_profile.xsd | 전자서명 문서 스키마 |
| 4.2.2.4. | readium-ep.xsd | 리소스 압축 정보에 대한 확장 스키마 |
| 4.2.2.5. | license\_schema.json | 라이선스 문서 스키마 |

#### epub-container.xsd

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <schema xmlns:ocf="urn:oasis:names:tc:opendocument:xmlns:container"  xmlns="http://www.w3.org/2001/XMLSchema"  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  xmlns:ns="http://www.idpf.org/2016/encryption#compression"  targetNamespace="urn:oasis:names:tc:opendocument:xmlns:container"  elementFormDefault="qualified"  attributeFormDefault="unqualified"  version="1.0">  <import namespace="http://www.w3.org/2000/09/xmldsig#"  schemaLocation="w3c\_xml\_signature\_profile.xsd"/>  <import namespace="http://www.w3.org/2001/04/xmlenc#"  schemaLocation="w3c\_xml\_encryption\_profile.xsd"/>  <import namespace="http://www.idpf.org/2016/encryption#compression"  schemaLocation="readium-ep.xsd"/>  <element name="encryption"  type="ocf:encryptionType"/>  <element name="signatures"  type="ocf:signaturesType"/>  <complexType name="encryptionType"  mixed="false">  <choice minOccurs="0"  maxOccurs="unbounded">  <element ref="xenc:EncryptedData"  minOccurs="0"/>  </choice>  </complexType>  <complexType name="signaturesType"  mixed="false">  <choice minOccurs="0"  maxOccurs="unbounded">  <element ref="ds:Signature"  minOccurs="0"  maxOccurs="unbounded"/>  </choice>  </complexType>  </schema> |

#### w3c\_xml\_encryption\_profile.xsd

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <schema xmlns="http://www.w3.org/2001/XMLSchema"  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  targetNamespace="http://www.w3.org/2001/04/xmlenc#"  elementFormDefault="qualified"  attributeFormDefault="unqualified"  version="1.0">  <import namespace="http://www.w3.org/2000/09/xmldsig#"  schemaLocation="w3c\_xml\_signature\_profile.xsd" />  <complexType name="EncryptedType"  abstract="true"  mixed="false">  <sequence>  <element name="EncryptionMethod"  type="xenc:EncryptionMethodType" />  <element ref="ds:KeyInfo"  minOccurs="0"/>  <element ref="xenc:CipherData" />  <element ref="xenc:EncryptionProperties"  minOccurs="0" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  <attribute name="MimeType"  type="string"  use="optional" />  </complexType>  <complexType name="EncryptionMethodType"  mixed="true">  <attribute name="Algorithm"  type="anyURI"  use="required" />  </complexType>  <element name="CipherData"  type="xenc:CipherDataType" />  <complexType name="CipherDataType"  mixed="false">  <choice>  <element ref="xenc:CipherReference" />  </choice>  </complexType>  <element name="CipherReference"  type="xenc:CipherReferenceType" />  <complexType name="CipherReferenceType"  mixed="false">  <attribute name="URI"  type="anyURI"  use="required" />  </complexType>  <element name="EncryptedData"  type="xenc:EncryptedDataType" />  <complexType name="EncryptedDataType"  mixed="false">  <complexContent mixed="false">  <extension base="xenc:EncryptedType" />  </complexContent>  </complexType>  <element name="EncryptionProperties"  type="xenc:EncryptionPropertiesType" />  <complexType name="EncryptionPropertiesType"  mixed="false">  <sequence>  <element ref="xenc:EncryptionProperty"  maxOccurs="unbounded" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  <element name="EncryptionProperty"  type="xenc:EncryptionPropertyType" />  <complexType name="EncryptionPropertyType"  mixed="true">  <choice maxOccurs="unbounded">  <any namespace="##other"  processContents="strict" />  </choice>  </complexType>  </schema> |

#### w3c\_xml\_signature\_profile.xsd

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <schema xmlns="http://www.w3.org/2001/XMLSchema"  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  targetNamespace="http://www.w3.org/2000/09/xmldsig#"  elementFormDefault="qualified"  attributeFormDefault="unqualified"  version="0.1">  <simpleType name="CryptoBinary">  <restriction base="base64Binary" />  </simpleType>  <element name="Signature"  type="ds:SignatureType" />  <complexType name="SignatureType"  mixed="false">  <sequence>  <element ref="ds:SignedInfo" />  <element ref="ds:SignatureValue" />  <element ref="ds:KeyInfo"  minOccurs="0" />  <element ref="ds:Object"  minOccurs="0" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  <element name="SignatureValue"  type="ds:SignatureValueType" />  <complexType name="SignatureValueType"  mixed="false">  <simpleContent>  <extension base="base64Binary">  <attribute name="Id"  type="ID"  use="optional" />  </extension>  </simpleContent>  </complexType>  <element name="SignedInfo"  type="ds:SignedInfoType" />  <complexType name="SignedInfoType"  mixed="false">  <sequence>  <element ref="ds:CanonicalizationMethod" />  <element ref="ds:SignatureMethod" />  <element ref="ds:Reference"  maxOccurs="unbounded" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  <element name="CanonicalizationMethod"  type="ds:CanonicalizationMethodType" />  <complexType name="CanonicalizationMethodType"  mixed="true">  <attribute name="Algorithm"  type="anyURI"  use="required" />  </complexType>  <element name="SignatureMethod"  type="ds:SignatureMethodType" />  <complexType name="SignatureMethodType"  mixed="true">  <attribute name="Algorithm"  type="anyURI"  use="required" />  </complexType>  <element name="Reference"  type="ds:ReferenceType" />  <complexType name="ReferenceType"  mixed="false">  <sequence>  <element ref="ds:Transforms"  minOccurs="0" />  <element ref="ds:DigestMethod" />  <element ref="ds:DigestValue" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  <attribute name="URI"  type="anyURI"  use="optional" />  <attribute name="Type"  type="anyURI"  use="optional" />  </complexType>  <element name="Transforms"  type="ds:TransformsType" />  <complexType name="TransformsType"  mixed="false">  <sequence>  <element ref="ds:Transform" />  </sequence>  </complexType>  <element name="Transform"  type="ds:TransformType" />  <complexType name="TransformType"  mixed="true">  <attribute name="Algorithm"  type="anyURI"  use="required" />  </complexType>  <!-- End Reference -->  <element name="DigestMethod"  type="ds:DigestMethodType" />  <complexType name="DigestMethodType"  mixed="true">  <attribute name="Algorithm"  type="anyURI"  use="required" />  </complexType>  <element name="DigestValue"  type="ds:DigestValueType" />  <simpleType name="DigestValueType">  <restriction base="base64Binary" />  </simpleType>  <element name="KeyInfo"  type="ds:KeyInfoType" />  <complexType name="KeyInfoType"  mixed="true">  <sequence minOccurs="0"  maxOccurs="unbounded">  <element ref="ds:KeyName"  minOccurs="0" />  <element ref="ds:RetrievalMethod"  minOccurs="0" />  <element ref="ds:X509Data"  minOccurs="0" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  <element name="KeyName"  type="string" />  <element name="RetrievalMethod"  type="ds:RetrievalMethodType" />  <complexType name="RetrievalMethodType"  mixed="false">  <attribute name="URI"  type="anyURI"  use="required" />  <attribute name="Type"  type="anyURI"  use="optional" />  </complexType>  <element name="X509Data"  type="ds:X509DataType" />  <complexType name="X509DataType"  mixed="false">  <sequence maxOccurs="unbounded">  <choice>  <element name="X509IssuerSerial"  type="ds:X509IssuerSerialType" />  <element name="X509Certificate"  type="base64Binary" />  </choice>  </sequence>  </complexType>  <complexType name="X509IssuerSerialType"  mixed="false">  <sequence>  <element name="X509IssuerName"  type="string" />  <element name="X509SerialNumber"  type="integer" />  </sequence>  </complexType>  <element name="Object"  type="ds:ObjectType" />  <complexType name="ObjectType"  mixed="true">  <sequence minOccurs="0"  maxOccurs="unbounded">  <any namespace="##any"  processContents="lax" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  <element name="Manifest"  type="ds:ManifestType" />  <complexType name="ManifestType"  mixed="false">  <sequence>  <element ref="ds:Reference"  maxOccurs="unbounded" />  </sequence>  <attribute name="Id"  type="ID"  use="optional" />  </complexType>  </schema> |

#### readium-ep.xsd

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <schema xmlns="http://www.w3.org/2001/XMLSchema"  xmlns:ns="http://www.idpf.org/2016/encryption#compression"  targetNamespace="http://www.idpf.org/2016/encryption#compression"  elementFormDefault="qualified"  attributeFormDefault="unqualified">  <element name="Compression"  type="ns:CompressionType" />  <complexType name="CompressionType"  mixed="false">  <attribute name="Method"  type="ns:methodType"  use="optional" />  <attribute name="OriginalLength"  type="positiveInteger"  use="optional" />  </complexType>  <simpleType name="methodType">  <restriction base="string">  <enumeration value="0" />  <enumeration value="8" />  </restriction>  </simpleType>  </schema> |

#### license\_schema.json

|  |
| --- |
| {  "title": "EPUB DRM license v1.0 schema",  "description": "Created by DRM inside Taehyun. Kim on 2015-07-02.",  "type": "object",  "definitions": {  "base64String": {  "type": "string",  "pattern": "^(?:[A-Za-z0-9+/]{4})\*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=|[A-Za-z0-9+/]{4})$"  },  "dateString": {  "type": "string",  "format": "date-time"  },  "uriString": {  "type": "string",  "format": "uri"  },  "positiveInteger": {  "type": "integer",  "minimum": 0  },  "percentInteger": {  "type": "integer",  "minimum": 0,  "maximum": 100  },  "stringArray": {  "type": "array",  "items": { "type": "string" },  "minItems": 1,  "uniqueItems": true  },  "base64StringArray": {  "type": "array",  "items": { "$ref": "#/definitions/base64String" },  "minItems": 1,  "uniqueItems": true  },  "deviceKeyObject": {  "type": "object",  "properties": {  "key\_name": { "type": "string" },  "key\_check": { "$ref": "#/definitions/base64String" }  },  "required": ["key\_check", "key\_name" ],  "additionalProperties": false  },  "userKeyObject": {  "type": "object",  "properties": {  "algorithm": { "$ref": "#/definitions/uriString" },  "key\_check": { "$ref": "#/definitions/base64String" },  "text\_hint": { "type": "string" }  },  "required": [ "algorithm", "key\_check", "text\_hint" ],  "additionalProperties": false  },  "contentKeyObject": {  "type": "object",  "properties": {  "algorithm": { "$ref": "#/definitions/uriString" },  "encrypted\_value": { "$ref": "#/definitions/base64String" }  },  "required": [ "algorithm", "encrypted\_value" ],  "additionalProperties": false  },  "encryptionObject": {  "type": "object",  "properties": {  "profile": { "$ref": "#/definitions/uriString" },  "content\_key": { "$ref": "#/definitions/contentKeyObject" },  "device\_key": { "$ref": "#/definitions/deviceKeyObject" },  "user\_key": { "$ref": "#/definitions/userKeyObject" }  },  "required": [ "profile", "content\_key" ],  "additionalProperties": false  },  "rightsObject": {  "type": "object",  "properties": {  "start": { "$ref": "#/definitions/dateString" },  "end": { "$ref": "#/definitions/dateString" },  "copy": { "$ref": "#/definitions/positiveInteger" },  "print": { "$ref": "#/definitions/positiveInteger" }  }  },  "linksPropertyObject": {  "type": "object",  "properties": {  "href": { "$ref": "#/definitions/uriString" },  "title": { "type": "string" },  "type": { "type": "string" },  "templated": { "type": "boolean" },  "length": { "$ref": "#/definitions/positiveInteger" },  "hash": { "$ref": "#/definitions/base64String" }  },  "additionalProperties": false,  "required": [ "href" ]  },  "linksObject": {  "type": "object",  "properties": {  "publication": { "$ref": "#/definitions/linksPropertyObject" },  "license": { "$ref": "#/definitions/linksPropertyObject" },  "hint": { "$ref": "#/definitions/linksPropertyObject" },  "self": { "$ref": "#/definitions/linksPropertyObject" }  },  "additionalProperties": {  "anyOf": [  {  "$ref": "#/definitions/linksPropertyObject"  },  {  "type": "array",  "items": {  "$ref": "#/definitions/linksPropertyObject"  },  "uniqueItems": true  }  ]  }  },  "informationObject": {  "type": "object",  "properties": {  "id": { "type": "string" },  "email": { "type": "string" },  "name": { "type": "string"},  "encrypted": { "$ref": "#/definitions/stringArray" }  }  },  "signatureObject": {  "type": "object",  "properties": {  "algorithm": { "$ref": "#/definitions/uriString" },  "certificate": { "$ref": "#/definitions/base64String" },  "ca\_certificates": {"$ref": "#/definitions/base64StringArray" },  "value": { "$ref": "#/definitions/base64String" }  },  "additionalProperties": false,  "required": [ "algorithm", "certificate", "value" ]  }  },  "properties": {  "id": { "type": "string" },  "issued": { "$ref": "#/definitions/dateString" },  "provider": { "$ref": "#/definitions/uriString" },  "updated": { "$ref": "#/definitions/dateString" },  "encryption": { "$ref": "#/definitions/encryptionObject" },  "rights": { "$ref": "#/definitions/rightsObject" },  "links": { "$ref": "#/definitions/linksObject" },  "user": { "$ref": "#/definitions/informationObject" },  "signature": { "$ref": "#/definitions/signatureObject" }  },  "additionalProperties": false,  "required": [  "id",  "issued",  "provider",  "encryption",  "signature"  ]  } |

## 검증자료

### 개요

검증자료는 기능 정합성 검증을 위해 사용되는 정합 또는 비정합 전자책 DRM 콘텐츠를 의미한다. 디바이스 키 방식의 검증자료는 검증 대상이 되는 뷰어 시스템에 종속적인 키로만 사용이 가능하기에 검증 대상 뷰어 시스템의 디바이스 인증서를 통해 패키징 되어야 한다. 시나리오별 검증자료의 제작 방법은 다음 4.3.2. ~ 4.3.5. 항목을 참고한다.

### 기본 문서 검증자료 제작 방법

#### 전자서명 문서가 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.1.1.2. 전자서명 문서가 없는 전자책 열람 확인 |
| 파일명 | no\_sig.epub |
| 설명 | 전자서명 문서가 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 검증 대상 뷰어 시스템의 디바이스 인증서와 평문 epub을 준비한다. 2. 디바이스 인증서를 사용하여 device\_key 방식의 DRM으로 평문 epub을 패키징한다. 이때 signatures.xml의 생성은 제외한다. |

#### 라이선스 문서가 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.1.2.1. 라이선스 문서가 없는 전자책 오류 처리 |
| 파일명 | no\_license.epub |
| 설명 | 라이선스 문서가 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 검증 대상 뷰어 시스템의 디바이스 인증서와 평문 epub을 준비한다. 2. 디바이스 인증서를 사용하여 device\_key 방식의 DRM으로 평문 epub을 패키징한다. 이때 license.lcpl 의 생성은 제외한다. |

#### 암호화 문서가 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.1.2.2. 암호화 문서가 없는 전자책 오류 처리 |
| 파일명 | no\_enc\_doc.epub |
| 설명 | 암호화 문서가 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. 3. 패키징된 epub 내부의 META-INF/encryption.xml을 삭제한다. |

### 복호화 처리 기능 검증 자료 제작 방법

#### 암호화된 리소스를 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.1. 암호화된 리소스를 포함한 전자책 열람 확인 |
| 파일명 | enc\_res.epub |
| 설명 | 암호화된 리소스를 가지는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub의 모든 리소스를 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 압축 암호화된 리소스를 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.2. 압축 암호화된 리소스를 포함한 전자책 열람 확인 |
| 파일명 | comp\_enc\_res.epub |
| 설명 | 압축되고 암호화된 리소스를 가지는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub의 모든 리소스를 압축하고, 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 압축/비압축 암호화된 리소스들을 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.3. 압축/비압축 암호화된 리소스들을 포함한 전자책 열람 확인 |
| 파일명 | comp\_enc\_res2.epub |
| 설명 | 압축/비압축 암호화된 리소스들을 함께 가지는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub의 리소스중 일부는 압축을 하고 암호화, 나머지는 그대로 암호화하여 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 평문/암호화된 리소스들을 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.4. 평문/암호화된 리소스들을 포함한 전자책 열람 확인 |
| 파일명 | enc\_res2.epub |
| 설명 | 평문/암호화된 리소스들을 함께 가지는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub의 리소스중 일부만 암호화하여 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화된 리소스 파일이 없는 암호화문서를 포함하는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.5. 암호화된 리소스 파일이 없는 암호화문서를 포함하는 전자책 열람 확인 |
| 파일명 | no\_enc\_res.epub |
| 설명 | 암호화된 리소스가 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub의 모든 리소스를 암호화하지 않고 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### User key 방식의 DRM이 적용된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.6. User Key 방식의 DRM이 적용된 전자책 열람 확인 |
| 파일명 | userkey\_type.epub |
| 설명 | 비밀번호로 보호된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 패스워드(password)를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 콘텐츠 암호화 알고리즘이 AES-256-GCM 방식인 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.7. 콘텐츠 암호화 알고리즘이 AES-256-GCM 방식인 전자책 열람 확인 |
| 파일명 | gcm256.epub |
| 설명 | AES-256-GCM 알고리즘으로 리소스가 암호화된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 이때 리소스 암호화 알고리즘은 AES-256-GCM을 사용한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화된 이미지 파일이 포함된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.8. 암호화된 이미지 파일이 포함된 전자책 열람 확인 |
| 파일명 | enc\_img.epub |
| 설명 | 암호화된 이미지 리소스가 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 이미지 리소스가 포함된 평문 epub의 모든 리소스를 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화된 음악 파일이 포함된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.9. 암호화된 음악 파일이 포함된 전자책 열람 확인 |
| 파일명 | enc\_mp3.epub |
| 설명 | 암호화된 음악 리소스가 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 음악 리소스가 포함된 평문 epub의 모든 리소스를 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화된 스크립트 파일이 포함된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.10. 암호화된 스크립트 파일이 포함된 전자책 열람 확인 |
| 파일명 | enc\_script.epub |
| 설명 | 암호화된 스크립트 리소스가 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 스크립트 리소스가 포함된 평문 epub의 모든 리소스를 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화된 동영상 파일이 포함된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.1.11. 암호화된 동영상 파일이 포함된 전자책 열람 확인 |
| 파일명 | enc\_mp4.epub |
| 설명 | 암호화된 동영상 리소스가 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 동영상 리소스가 포함된 평문 epub의 모든 리소스를 암호화하여, 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 올바르지 않은 암호화 문서를 포함한 전자책1

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.1. 올바르지 않은 암호화 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_enc1.epub |
| 설명 | 표준 스키마를 준수하지 않는 encryption.xml이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 encryption.xml의 기본 네임스페이스를 ""urn:oasis:names:tc:opendocument:xmlns:content""로 설정한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 올바르지 않은 암호화 문서를 포함한 전자책2

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.1. 올바르지 않은 암호화 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_enc2.epub |
| 설명 | 표준 스키마를 준수하지 않는 encryption.xml이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 encryption.xml의 EncryptedData의 네임 스페이스를 ""http://www.w3.org/2001/04/encryption""으로 설정한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### key의 URI 정보가 비표준인 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.2. key의 URI, type 정보가 비표준인 전자책 오류 처리 |
| 파일명 | invalid\_key\_info1.epub |
| 설명 | 표준 keyInfo URI를 가지지 않는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 encryption.xml의 ""EncryptedData/KeyInfo/RetrievalMethod/@URI""의 값이 ""license.lcpl#/encryption/contentKey""로 설정한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### key의 type 정보가 비표준인 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.2. key의 URI, type 정보가 비표준인 전자책 오류 처리 |
| 파일명 | invalid\_key\_info2.epub |
| 설명 | 표준 keyInfo type을 가지지 않는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 encryption.xml의 ""EncryptedData/KeyInfo/RetrievalMethod/@Type""의 값은 ""http://readium.org/2014/01/lcp#ContentKey""로 설정한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 라이선스 문서의 content\_key 영역에 ECK가 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.3. 라이선스 문서의 content\_key 영역에 ECK가 없는 전자책 오류 처리 |
| 파일명 | no\_eck.epub |
| 설명 | 라이선스 문서에 ECK가 포함되지 않은 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 license.lcpl에서 encryption/device\_key 항목을 삭제하여 진행한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 검증대상 기기의 device key가 아닌 key로 암호화된 경우

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.4. 검증대상 기기의 device key가 아닌 key로 암호화된 경우 오류 처리 |
| 파일명 | invalid\_dev\_key.epub |
| 설명 | 라이선스 문서에 등록된 key와 다른 key로 패키징된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, 단 cek의 암호화시에는 다른 비밀키를 통해 암호화하고 content\_key/encrypted\_value에 저장한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 암호화문서에 등록된 암호화 대상 리소스가 존재하지 않는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.6. 암호화문서에 등록된 암호화 대상 리소스가 존재하는 않는 경우의 오류 처리 |
| 파일명 | not\_ex\_res.epub |
| 설명 | 암호화대상 리소스의 일부가 존재하지 않는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. 3. 첫페이지의 리소스를 전자책 컨테이너에서 삭제한다. |

#### 암호화된 리소스 파일이 손상된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.2.2.7. 암호화된 리소스 파일이 손상된 경우의 오류 처리 |
| 파일명 | damaged\_res.epub |
| 설명 | 암호화대상 리소스의 일부가 손상된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스의 권한은 전부 유효한 상태이다. 3. 첫페이지의 암호화된 리소스 파일을 랜덤한 바이트의 파일로 변경한다. |

### 전자서명 확인 기능 검증 자료 제작 방법

#### 전자서명 문서를 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.1.1. 전자서명 문서를 포함한 전자책 열람 확인 |
| 파일명 | with\_sig.epub |
| 설명 | signatures.xml이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 올바르지 않은 전자서명 문서를 포함한 전자책1

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.1. 올바르지 않은 전자서명 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_sig1.epub |
| 설명 | 표준 스키마를 준수하지 않는 signatures.xml이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 이때 signatures.xml의 signatures/Signature의 네임 스페이스가 ""http://www.w3.org/2000/09/signature""로 설정한다. 3. 라이선스의 권한은 전부 유효한 상태이다. |

#### 올바르지 않은 전자서명 문서를 포함한 전자책2

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.1. 올바르지 않은 전자서명 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_sig2.epub |
| 설명 | 표준 스키마를 준수하지 않는 signatures.xml이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 이때 signautres.xml의 루트 엘리먼트는 <root-signatures.xml>이다. 3. 라이선스의 권한은 전부 유효한 상태이다. |

#### 리소스의 해시값이 유효하지 않은 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.2. 리소스의 해시값이 유효하지 않은 전자책 오류 처리 |
| 파일명 | invalid\_res\_hash.epub |
| 설명 | 전자서명 문서에 기록된 리소스의 해시값과 실제 해시값이 다른 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 라이선스의 권한은 전부 유효한 상태이다. |

#### 인증서 체인이 유효하지 않은 signatures.xml을 포함하는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.3. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain1.epub |
| 설명 | 유효하지 않은 인증서 체인으로 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. signatures.xml의 전자서명은 유효하지 않은 CA에서 발급된 인증서를 사용한다. 3. 라이선스의 권한은 전부 유효한 상태이다. |

#### 만료된 인증서로 signatures.xml 문서가 전자서명된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.3. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain2.epub |
| 설명 | 유효하지 않은 인증서 체인으로 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 이때 signatures.xml 문서의 전자서명은 만료된 인증서로를 사용한다. 3. 라이선스의 권한은 전부 유효한 상태이다. |

#### 비도가 낮은 인증서가 포함된 인증서 체인으로 signatures.xml 문서가 전자서명된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.3. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain3.epub |
| 설명 | 유효하지 않은 인증서 체인으로 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. 이때 signatures.xml 문서의 전자서명에는 인증서의 비도가 1024bit인 인증서가 포함된 인증서 체인을 사용한다. 3. 라이선스의 권한은 전부 유효한 상태이다. |

#### 전자서명이 유효하지 않은 signautres.xml을 포함하는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.3.2.4. 전자서명이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_signature.epub |
| 설명 | 전자서명이 훼손된 signatures.xml을 포함하는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징하되, signatures.xml을 추가한다. 2. signatures.xml을 전자서명하고, signedInfo/Reference/DigestValue의 값을 다른 임의의값으로 수정한다. 3. 라이선스의 권한은 전부유효한 상태이다. |

### 라이선스 확인 기능 검증 자료 제작 방법

#### 기간 권한이 유효한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.1. 기간 권한이 유효한 전자책 |
| 파일명 | valid\_license\_period.epub |
| 설명 | 검사일 기준 유효한 기간 권한이 있는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 검사 기간 내에 한하여 유효한 기간 권한을 주고 패키징한다. |

#### copy 권한이 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.2. copy 권한 제어 확인 |
| 파일명 | limited\_copy\_right\_0.epub |
| 설명 | 라이선스 문서에 copy 권한이 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 copy 권한에 0을 설정하여 패키징한다. |

#### 제한된 copy 권한이 있는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.2. copy 권한 제어 확인 |
| 파일명 | limited\_copy\_right\_100.epub |
| 설명 | 라이선스 문서에 한정적으로 콘텐츠 내용 복사가 가능한 copy 권한이 있는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 일정량의 콘텐츠 내용 복사가 가능한 copy 권한 (100)을 설정하여 패키징한다. |

#### 무제한의 copy 권한이 있는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.2. copy 권한 제어 확인 |
| 파일명 | unlimited\_copy\_right.epub |
| 설명 | 라이선스 문서에 무제한 copy 권한이 있는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 copy 권한을 제외하고 패키징한다. copy 권한이 제외된 경우 무제한 copy 권한을 가진다. |

#### print 권한이 없는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.3. print 권한 제어 확인 |
| 파일명 | limited\_print\_right\_0.epub |
| 설명 | 라이선스 문서에 print 권한이 없는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 print 권한에 0을 설정하여 패키징한다. |

#### 제한이 있는 print 권한이 있는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.3. print 권한 제어 확인 |
| 파일명 | limited\_print\_right\_2.epub |
| 설명 | 라이선스 문서에 한정적으로 콘텐츠 내용을 프린트 할수 있는 권한이 있는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 일정량의 콘텐츠 내용 프린트가 가능한 print 권한 (2)을 설정하여 패키징한다. |

#### 무제한의 print 권한이 있는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.1.3. print 권한 제어 확인 |
| 파일명 | unlimited\_print\_right.epub |
| 설명 | 라이선스 문서에 무제한 print 권한이 있는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 print 권한을 제외하고 패키징한다. print 권한이 제외된 경우 무제한 print 권한을 가진다. |

#### 올바르지 않은 라이선스 문서를 포함한 전자책1

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.1. 올바르지 않은 라이선스 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_license1.epub |
| 설명 | 표준 스키마를 준수하지 않는 license.lcpl이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징을 하되, 라이선스 문서를 생성할 때 "signature" 항목을 제거하고 만든다. |

#### 올바르지 않은 라이선스 문서를 포함한 전자책2

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.1. 올바르지 않은 라이선스 문서를 포함한 전자책 오류 처리 |
| 파일명 | invalid\_license2.epub |
| 설명 | 표준 스키마를 준수하지 않는 license.lcpl이 포함된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징을 하되, 라이선스 문서를 생성할 때 "issued" 항목을 제거하고 만든다. |

#### 인증서 체인이 유효하지 않은 license.lcpl을 포함한 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.2. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain\_lic1.epub |
| 설명 | 유효하지 않은 인증서 체인으로 라이선스 문서가 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징을 하되, 라이선스 문서를 생성할 때 인증서 체인 중 하나만 다른 인증서를 넣어 만든다. |

#### 만료된 인증서로 license.lcpl 문서가 전자서명된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.2. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain\_lic2.epub |
| 설명 | 유효하지 않은 인증서 체인으로 라이선스 문서가 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징을 하되, 라이선스 문서를 생성할 때 인증서 체인 중 하나만 다른 인증서를 넣어 만든다. 2. 이때 라이선스 문서의 전자서명은 만료된 인증서로를 사용한다. |

#### 비도가 낮은 인증서가 포함된 인증서체인으로 license.lcpl 문서가 전자서명된 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.2. 인증서 체인이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_chain\_lic3.epub |
| 설명 | 유효하지 않은 인증서 체인으로 라이선스 문서가 전자서명된 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징을 하되, 라이선스 문서를 생성할 때 인증서 체인 중 하나만 다른 인증서를 넣어 만든다. 2. 이때 라이선스 문서의 전자서명에는 인증서의 비도가 1024bit인 인증서가 포함된 인증서 체인을 사용한다. |

#### 전자서명이 유효하지 않은 license.lcpl를 포함하는 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.3. 전자서명이 유효하지 않은 경우의 오류 처리 |
| 파일명 | invalid\_sig\_value\_lic.epub |
| 설명 | 전자서명이 훼손된 license.lcpl을 포함하는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 라이선스 문서의 전자서명 값에 추가 문자열 ""ABCD""를 붙인다. |

#### 라이선스 문서의 시작일이 검사일 이후인 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.4. 라이선스 문서의 시작일이 검사일 이후인 전자책의 오류 처리 |
| 파일명 | before\_start\_period.epub |
| 설명 | 시작일이 검사일보다 늦은 날짜를 가지는 라이선스를 포함하는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 기간 권한의 시작 날짜를 검사 기간보다 이후로 하여 설정한다. |

#### 라이선스 문서의 종료일이 검사일 이전인 전자책

|  |  |
| --- | --- |
| 테스트 | 2.2.4.2.5. 라이선스 문서의 종료일이 검사일 이전인 전자책의 오류 처리 |
| 파일명 | after\_end\_period.epub |
| 설명 | 종료일이 검사일보다 이른 날짜를 가지는 라이선스를 포함하는 전자책 DRM 콘텐츠 |
| 제작방법 | 1. 평문 epub을 표준 명세서에 맞게 디바이스 인증서를 통해 패키징한다. 2. 단, 라이선스의 권한 중 기간 권한의 끝 날짜를 검사 기간보다 이전으로 하여 설정한다. |

## 소스코드

### compression\_checker

|  |
| --- |
| #!/bin/bash  #  # compression\_checker for the Readium LCP license v1.0 license  # ver1.0  #  # Created by Readium Foundation in 2015  # Copyright (c) 2015 The Readium Foundation and contributors.  #  #  # Usage:: compression\_checker $1 $2  # $1 : encrypted\_resource name  # $2 : encryption.xml file name  #  declare -r TRUE=1  declare -r FALSE=0  declare -r NS\_ENC="http://www.w3.org/2001/04/xmlenc#"  declare -r NS\_EXT="http://www.idpf.org/2016/encryption#compression"  ret=$TRUE  if [ "$2" == "" ]; then echo "Resource name is required"; ret=$FALSE ; fi  if [ ! -f "$1" ]; then echo "$2 not found"; ret=$FALSE ; fi  if [ $ret != $TRUE ] ; then  echo "Usage:: compression\_checker \$1 \$2"  echo " \$1 : Encrypted Resource name"  echo " \$2 : encryption.xml file name"  exit  fi  value=$(xmlstarlet sel -N enc=$NS\_ENC -N ns=$NS\_EXT -t -v "//enc:CipherData/enc:CipherReference[@URI='$2']/../..//enc:EncryptionProperties/enc:EncryptionProperty/ns:Compression/@Method" $1)  if [ [$value=="8"] ]; then  echo "Resource is compressed"  elif [ [$value=="0"] ]; then  echo "Resource is uncompressed"  else  echo "Compression check error.."  fi |

### key\_checker

|  |
| --- |
| #!/bin/bash  #  # Key\_checker for the Readium LCP license v1.0 license  # ver1.0  #  # Created by Readium Foundation in 2015  # Copyright (c) 2015 The Readium Foundation and contributors.  #  # Usage:: key\_checker $1 $2  # $1 : lcp\_license\_file\_name  # $2 : passphrase  #  declare -r TRUE=1  declare -r FALSE=0  declare -r CHECK\_VALUE=".encryption.user\_key.key\_check"  declare -r Base64Decoding="openssl enc -base64 -d -A"  declare -r SelectIV="dd bs=16 count=1"  declare -r SelectRecourse="dd bs=1 skip=16"  declare -r Bin2Hex="od -A n -t x1"  trim() {  echo $1 | tr -d '\040\011\012\014\015'  }  getPaddingSize() {  padSize=$(echo -n "$1" | tail -c 1 | od -An -vtu1)  echo -n $padSize  }  ret=$TRUE  if [ ! -f "$1" ]; then echo "$1 not found";ret=$FALSE ; fi  if [ "$2" == "" ]; then echo "Need for passphrase"; ret=$FALSE ; fi    if [ $ret != $TRUE ] ; then  echo "Usage:: key\_checker \$1 \$2"  echo " \$1 : lcp license file name"  echo " \$2 : passphrase"  exit  fi  #Get Resource IV from Encrypted Resource file  iv=$(jq -r $CHECK\_VALUE "$1"| $Base64Decoding | $SelectIV 2>/dev/null| $Bin2Hex )  key=$(echo -n "$2" | openssl dgst -sha256 -binary | od -A n -v -t x1 )  id=$(jq -r .id $1)  echo ""  echo " iv::" `echo -n $(trim "$iv")`  echo " key::" `echo -n $(trim "$key")`  echo ""  echo " original\_id ::" `echo -n $id`  dec\_data=$(jq -r $CHECK\_VALUE "$1" | $Base64Decoding | $SelectRecourse 2>/dev/null | openssl enc -d -aes-256-cbc -K `echo $(trim "$key")` -iv `echo $(trim "$iv")` -nopad)  paddCnt=$(getPaddingSize "$dec\_data")  if [ $paddCnt -gt 16 ]; then  echo "Bad padding"  fi  actual\_size=$(($(echo "$dec\_data" | wc -c | cut -d " " -f -1) - $paddCnt))  dec\_data=$(echo -n $dec\_data | head -c $actual\_size)  echo "decrypted\_key\_check ::" $dec\_data  if [ $(trim "$id") == $(trim "$dec\_data") ]; then  echo "passphrase is correct"  else  echo "passphrase is incorrect"  fi |

### resource\_decrypter

|  |
| --- |
| #!/bin/bash  #  # resource\_decrypter for the Readium LCP license v1.0 license  # ver1.0  #  # Created by Readium Foundation in 2015  # Copyright (c) 2015 The Readium Foundation and contributors.  #  # Usage:: resource\_decrypter $1 $2 $3 $4  # $1 : encrypted\_resource\_file  # $2 : lcp\_license\_file\_name  # $3 : encryption.xml file name  # $4 : passphrase or device\_key  #  declare -r TRUE=1  declare -r FALSE=0  declare -r CEK\_VALUE=".encryption.content\_key.encrypted\_value"  declare -r Base64Decoding="openssl enc -base64 -d -A"  declare -r SelectIV="dd bs=16 count=1"  declare -r SelectRecourse="dd bs=1 skip=16"  declare -r Bin2Hex="od -A n -t x1"  declare -r NS\_ENC="http://www.w3.org/2001/04/xmlenc#"  declare -r NS\_EXT="http://www.idpf.org/2016/encryption#compression"  declare -r ALGO\_256GCM="http://www.w3.org/2009/xmlenc11#aes256-gcm"  declare -r ALGO\_256CBC="http://www.w3.org/2001/04/xmlenc#aes256-cbc"  print\_usage() {  echo "Usage:: resource\_decrypter \$1 \$2 \$3 \$4"  echo " \$1 : encrypted\_resource\_file"  echo " \$2 : lcp\_license\_file\_name"  echo " \$3 : encryption.xml file name"  echo " \$4 : passphrase or device\_key"  }  trim() {  echo $1 | tr -d '\040\011\012\015'  }  getPaddingSize() {  padSize=$(tail -c 1 "$1"| od -An -vtu1)  echo -n $padSize  }  ret=$TRUE  if [ ! -f "$1" ]; then echo "$1 not found";ret=$FALSE ; fi  if [ ! -f "$2" ]; then echo "$2 not found";ret=$FALSE ; fi  if [ ! -f "$3" ]; then echo "$3 not found";ret=$FALSE ; fi  if [ "$4" == "" ]; then echo "passphrase or device\_key missing"; ret=$FALSE ; fi    if [ $ret != $TRUE ] ; then  $print\_usage;  exit  fi  #Get CEK IV from license.lcpl  cek\_iv=$(jq -r $CEK\_VALUE "$2" | $Base64Decoding | $SelectIV 2>null | $Bin2Hex )  key\_type=$(jq -r ".encryption" "$2")  if [ `echo $key\_type | awk '{print match($0, "user\_key")}'` -gt 0 ]; then  key\_type="user"  fi  if [ `echo $key\_type | awk '{print match($0, "device\_key")}'` -gt 0 ]; then  key\_type="device"  fi  if [ "$key\_type" == "user" ]; then  #Drive user\_key  userkey=$(echo -n "$4" | openssl dgst -sha256 -binary | $Bin2Hex )  #Get CEK from license.lcpl  cek=$(jq -r $CEK\_VALUE "$2" | $Base64Decoding | $SelectRecourse 2>null | openssl enc -d -aes-256-cbc -nopad -K `echo $(trim "$userkey")` -iv `echo $(trim "$cek\_iv")`| $Bin2Hex )  tmp\_cek=$(echo -n $(trim "$cek") | head -c 64)  cek=$tmp\_cek  elif [ "$key\_type" == "device" ]; then  if [ ! -f "$4" ]; then echo "$4 not found";  $print\_usage;  exit  fi  cek=$(jq -r $CEK\_VALUE "$2" | $Base64Decoding | openssl rsautl -decrypt -oaep -inkey $4 | $Bin2Hex )  fi    echo ""  echo "license ::" `echo -n $2`  if [ "$key\_type" == "user" ]; then  echo "passphrase ::" `echo -n $4`  echo "userkey ::" `echo $(trim "$userkey")`  elif [ "$key\_type" == "device" ]; then  echo "device key ::" `echo -n $4`  fi  rsc\_enc\_algo=$(xmlstarlet sel -N enc=$NS\_ENC -t -v "//enc:CipherData/enc:CipherReference[@URI='$1']/../..//enc:EncryptionMethod/@Algorithm" $3)    echo "resource encryption algorithm= " $rsc\_enc\_algo  echo ""  echo "cek\_iv ::" `echo $(trim "$cek\_iv")`  echo "cek ::" `echo $(trim "$cek")`  echo ""  paddCnt=0  if [ "$rsc\_enc\_algo" == "$ALGO\_256CBC" ]; then  #Get Resource IV from encrypted resource file  resource\_iv=$(head -c 16 "$1"| $Bin2Hex )  echo "resource\_iv::" `echo $(trim "$resource\_iv")`  #Decrypt the encrypted resource part  filesize=$(($(wc -c "$1" | cut -d " " -f -1) - 16))  tail -c $filesize "$1" | openssl enc -d -aes-256-cbc -out "$1.tmp2" -K `echo $(trim "$cek")` -iv `echo $(trim "$resource\_iv")` -nopad  paddCnt=$(getPaddingSize "$1.tmp2")  if [ $paddCnt -gt 16 ]; then  echo ""  echo "Padding invalid"  exit  fi  elif [ "$rsc\_enc\_algo" == "$ALGO\_256GCM" ]; then  #Get Resource IV from encrypted resource file  resource\_iv=$(head -c 12 "$1"| $Bin2Hex )  echo "resource\_iv::" `echo $(trim "$resource\_iv")`  #Decrypt the encrypted resource part  filesize=$(($(wc -c "$1" | cut -d " " -f -1) - 12))  tail -c $filesize "$1" >> "$1.tmp3"  filesize=`expr $filesize - 16`  head -c $filesize "$1.tmp3" | openssl enc -d -aes-256-gcm -out "$1.tmp2" -K `echo $(trim "$cek")` -iv `echo $(trim "$resource\_iv")` 2>null  rm "$1.tmp3"  fi  actual\_size=$(($(wc -c "$1.tmp2" | cut -d " " -f -1) - $paddCnt))  head -c $actual\_size "$1.tmp2" > "$1".tmp  is\_comp\_rsc=$(xmlstarlet sel -N enc=$NS\_ENC -N ns=$NS\_EXT -t -v "//enc:CipherData/enc:CipherReference[@URI='$1']/../..//enc:EncryptionProperties/enc:EncryptionProperty/ns:Compression/@Method" $3)  if [ "$is\_comp\_rsc" == "8" ]; then  openssl zlib -d -in "$1.tmp" -out "$1.out"  else  cp "$1.tmp2" "$1.out"  fi  rm "$1.tmp"  rm "$1.tmp2"  if [ -e "null" ]; then  rm "null"  fi  echo "encryped\_resource :: $1"  echo "decrypted\_resource :: $1.out" |

### user\_info\_decrypter

|  |
| --- |
| #!/bin/bash  #  # user\_info\_decrypter for the Readium LCP license v1.0 license  # ver1.0  #  #  # Created by Readium Foundation in 2015  # Copyright (c) 2015 The Readium Foundation and contributors.  #  # To run on windows environment, remove the carriage return characters from the file  # with dos2unix tool before running the script  #  # Usage:: user\_info\_decrypter $1 $2 $3  # $1 : user info item (e.g, "id" or "email")  # $2 : lcp\_license\_file\_name  # $3 : passphrase  #  declare -r TRUE=1  declare -r FALSE=0  declare -r CEK\_VALUE=".encryption.content\_key.encrypted\_value"  declare -r Base64Decoding="openssl enc -base64 -d -A"  declare -r SelectIV="dd bs=16 count=1"  declare -r SelectRecourse="dd bs=1 skip=16"  declare -r Bin2Hex="od -A n -t x1"  trim() {  echo $1 | tr -d '\040\011\012\014\015'  }  getPaddingSize() {  padSize=$(echo -n "$1" | tail -c 1 | od -An -vtu1)  echo -n $padSize  }  ret=$TRUE  if [ "$1" == "" ]; then echo "Need for user info item";ret=$FALSE ; fi  if [ ! -f "$2" ]; then echo "$2 not found";ret=$FALSE ; fi  if [ "$3" == "" ]; then echo "Need for passphrase"; ret=$FALSE ; fi    if [ $ret != $TRUE ] ; then  echo "Usage:: user\_info\_decrypter \$1 \$2 \$3"  echo " \$1 : user info item (e.g, "id" or "email")"  echo " \$2 : lcp\_license\_file\_name"  echo " \$3 : passphrase"  exit  fi  check=$(jq .user.encrypted "$2")  if [[ `echo $(trim "$check")` == "null" ]]; then  echo $1 is not encrypted  exit  fi    check=$(jq -r '.user.encrypted[]' "$2" | grep $1 | wc -l)  if [ "$check" -ne 1 ]; then  echo $1 is not encrypted  exit  fi  #Get Resource IV from Encrypted Resource file  iv=$(jq -r .user.$1 "$2"| $Base64Decoding | $SelectIV 2>/dev/null| $Bin2Hex )  key=$(echo -n "$3" | openssl dgst -sha256 -binary | od -A n -v -t x1 )  echo ""  echo " iv::" `echo -n $(trim "$iv")`  echo " key::" `echo -n $(trim "$key")`  echo ""  dec\_data=$(jq -r .user.$1 "$2" | $Base64Decoding | $SelectRecourse 2>/dev/null | openssl enc -d -aes-256-cbc -K `echo $(trim "$key")` -iv `echo $(trim "$iv")` -nopad)  paddCnt=$(getPaddingSize "$dec\_data")  if [ $paddCnt -gt 16 ]; then  echo "Padding invalid"  exit  fi  actual\_size=$(($(echo "$dec\_data" | wc -c | cut -d " " -f -1) - $paddCnt))  echo "encryped\_user\_info ::" `jq -r .user.$1 "$2"`  echo "decrypted\_user\_info::" `echo -n $dec\_data | head -c $actual\_size` |

### u\_checker.py

|  |
| --- |
| # -\*- encoding: utf-8 -\*-  ###  # u\_checker.py  # license.lcpl 중복 키 체크 스크립트  #  # created by Ahram (08-03-2016)  ###  import sys  encryption\_path = '//encryption/'  def check\_duplicated(route, val):  """  :param route: license.lcpl json문서 안에서 val 값 까지의 경로  :param val: json문서 안에서 {} 내부의 substring  :return 발생한 오류에 대한 메시지(중복값, user\_key - device\_key간의 상호배재 여부) 오류가 없으면 빈 string을 반환함.  """  route = route.lower()  idx\_list = list()  idx = 0  while idx < len(val):  if val[idx] == '{':  idx\_list.append(idx)  if val[idx] == '}':  start\_bracket = idx\_list.pop()  val = val.replace(val[start\_bracket:idx + 1], '"null"')  idx = 0  idx\_list.clear()  idx += 1  json\_object = val.split(',')  keys = list()  res = ""  for o in json\_object:  key = o.split(':')[0]  if not key in keys:  keys.append(key )  else:  res += route + ": " + key + " is duplicated.\r\n"  # ECK(user\_key or device\_key)를 체크하기 위해 예외로 생성한 스크립트  if route == encryption\_path:  has\_user\_key = '"user\_key"' in keys  has\_dev\_key = '"device\_key"' in keys  if has\_user\_key and has\_dev\_key:  res += route + ": ECK is duplicated.\r\n"  elif not has\_user\_key and not has\_dev\_key:  res += route + ": ECK is not found.\r\n"  return res  def document\_walker(root, json\_str):  """  :param root: json문서 안에서 json\_str의 경로  :param json\_str: json object string  :returns: substring val 안에 있는 '}'의 위치 index값, check\_duplicated 함수에서 발생한 오류 메세지.  """  idx = 0  key = ""  res = ""  while idx < len(json\_str):  if json\_str[idx] == '"':  idx += 1  end\_quote = json\_str.find('"', idx)  key = json\_str[idx:end\_quote]  idx = end\_quote + 1  elif json\_str[idx] == ',':  key = ""  idx += 1  elif json\_str[idx] == '{':  sub\_idx, err\_msg = document\_walker(root + "/" + key, json\_str[idx + 1:])  idx = idx + sub\_idx + 2  res += err\_msg  elif json\_str[idx] == '}':  return idx, res + check\_duplicated(root + '/', json\_str[0:idx])  else:  idx += 1  return 0, res  if \_\_name\_\_ == "\_\_main\_\_":  arg = sys.argv  if len(arg) != 2:  print("usage: python u\_checker.py [.lcpl file]")  pass  file = arg[1]  if not '.lcpl' == file[-5:]:  print("Target file extension must be 'lcpl'.")  else:  with open(file, encoding='utf-8') as f:  data = f.read()  data = ''.join([chr(ch) for ch in data.encode() if not chr(ch) in ['\r', '\n', '\t', ' ']])  \_, msg = document\_walker("", data)  # 키 중복에 대한 에러 메세지를 파일로 출력하고 싶은 경우에는 print 스크립트를 파일 출력으로 변경해야 한다.  if len(msg) == 0:  msg = "Validation clear"  print(msg) |

### signatureValidator.java

|  |
| --- |
| //  // signatureValidator for the Readium LCP license v1.0 license  // ver1.0  //  // Created by Readium Foundation in 2015  // Copyright (c) 2015 The Readium Foundation and contributors.  //  //  // Usage:: signatureValidator $1 $2  // $1 : root\_certificate\_file\_name  // $2 : lcp\_license\_file\_name  //  import java.io.ByteArrayInputStream;  import java.io.File;  import java.io.IOException;  import java.io.InputStream;  import java.lang.reflect.Type;  import java.math.BigDecimal;  import java.nio.charset.Charset;  import java.nio.charset.StandardCharsets;  import java.nio.file.Files;  import java.nio.file.Paths;  import java.security.InvalidKeyException;  import java.security.NoSuchAlgorithmException;  import java.security.PublicKey;  import java.security.Signature;  import java.security.SignatureException;  import java.security.cert.CertificateException;  import java.security.cert.CertificateFactory;  import java.security.cert.X509Certificate;  import java.util.ArrayList;  import java.util.Iterator;  import java.util.List;  import java.util.Map;  import java.util.TreeMap;  import org.bouncycastle.util.encoders.Base64;  import com.google.gson.Gson;  import com.google.gson.GsonBuilder;  import com.google.gson.JsonElement;  import com.google.gson.JsonPrimitive;  import com.google.gson.JsonSerializationContext;  import com.google.gson.JsonSerializer;  import com.google.gson.reflect.TypeToken;  public class Validator {  public static String usage = "Usage: signatureValidator $root\_certificate $license\_lcpl\_file \n";  public static Gson mGson = null;      public static void main(String[] args) {    mGson = getGsonInstance();    if(checkParameter(args) == false) {  return;  }    String rootCertificatePath = args[0];  String licenseDocumentPath = args[1];    String json;  try {  json = new String(Files.readAllBytes(Paths.get(licenseDocumentPath)), Charset.forName("UTF-8"));  } catch (IOException e1) {  System.err.println("signatureValidator:Error: Invalid IO.\n");  e1.printStackTrace();  return;  }    String sig\_algorithm = "";  String sig\_value = "";  String sig\_cert = "";  List<X509Certificate> ca\_cert\_list = new ArrayList<X509Certificate>();  X509Certificate self = null;  X509Certificate root = null;  Type type = new TypeToken<Map<String, Object>>(){}.getType();  Map<String, Object> lcplMap = mGson.fromJson(json, type);  for (String key : lcplMap.keySet()) {  if (key.equalsIgnoreCase("signature")) {  Map<String, Object> sigMap = mGson.fromJson(mGson.toJson(lcplMap.get(key)), type);  sig\_algorithm = (String)sigMap.get("algorithm");  sig\_value = (String)sigMap.get("value");    try {  sig\_cert = (String)sigMap.get("certificate");  @SuppressWarnings("unchecked")  List<String> certs = (List<String>)sigMap.get("ca\_certificates");  if (certs!=null) {  for (int i = 0; i < certs.size(); i++) {  X509Certificate cert = stringToCertificate(certs.get(i));  ca\_cert\_list.add(cert);  }  }  } catch (CertificateException e) {  System.err.println("signatureValidator:Error: Invalid certificates.\n");  return;  }  }  }    try {  root = stringToCertificate(new String(Files.readAllBytes(Paths.get(rootCertificatePath)), StandardCharsets.UTF\_8));  } catch (CertificateException e) {  System.err.println("signatureValidator:Error: Invalid root certificate.\n");  return;  } catch (IOException e) {  System.err.println("signatureValidator:Error: not found root certificate.\n");  }    try {  self = stringToCertificate(sig\_cert);  } catch (CertificateException e) {  System.err.println("signatureValidator:Error: Invalid leaf certificate.\n");  return;  }    List<X509Certificate> sortedList = getSortedItem(self, ca\_cert\_list, root);  if (sortedList == null) {  return;  }    String removedSigJson = removeSignature(json);  TreeMap<String, Object> removedMap = sortByTreeMap(removedSigJson);    String canonicaledJson = mGson.toJson(removedMap);    byte[] data = canonicaledJson.getBytes();  byte[] signedData = Base64.decode(sig\_value.getBytes());    boolean result = false;  try {  if (sig\_algorithm.equalsIgnoreCase("http://www.w3.org/2001/04/xmldsig-more#rsa-sha256")) {  result = verifyData(data, signedData, self.getPublicKey());  } else {  System.err.println("signatureValidator:Error: Signature Algorithm is not rsa-sha256.\n");  }    } catch (InvalidKeyException e) {  System.err.println("signatureValidator:Error: Invalid key.\n");  } catch (NoSuchAlgorithmException e) {  System.err.println("signatureValidator:Error: No Such Algorithm.\n");  } catch (SignatureException e) {  System.err.println("signatureValidator:Error: Invalid signature.\n");  }    if (result) {  System.out.println("signatureValidator:Result:Success: \n["+licenseDocumentPath+"]'s signature is correct.\n");  } else {  System.out.println("signatureValidator:Result:Fail: \n["+licenseDocumentPath+"]'s signature is not correct.\n");  }  }  public static boolean verifyData(byte[] data, byte[] sigBytes, PublicKey publicKey)  throws NoSuchAlgorithmException, InvalidKeyException, SignatureException {  Signature signature = Signature.getInstance("SHA256withRSA");  signature.initVerify(publicKey);  signature.update(data);  return signature.verify(sigBytes);  }    public static String removeSignature(String json) {  Type type = new TypeToken<Map<String, Object>>(){}.getType();  Map<String, Object> map = mGson.fromJson(json, type);    Iterator<Map.Entry<String, Object>> iter = map.entrySet().iterator();  while (iter.hasNext()) {  Map.Entry<String, Object> entry = iter.next();  if ("signature".equalsIgnoreCase(entry.getKey())) {  iter.remove();  }  }  return mGson.toJson(map);  }    public static TreeMap<String, Object> sortByTreeMap(String json) {  @SuppressWarnings("unchecked")  TreeMap<String, Object> map = mGson.fromJson(json, TreeMap.class);  for(String key : map.keySet()) {  if (map.get(key) instanceof Map) {  map.put(key, sortByTreeMap(mGson.toJson(map.get(key))));  }  }  return map;  }    public static Gson getGsonInstance() {  return new GsonBuilder()  .serializeNulls()  .disableHtmlEscaping()  .registerTypeAdapter(Double.class, new JsonSerializer<Double>() {  @Override  public JsonElement serialize(Double originalValue, Type typeOf, JsonSerializationContext context) {  BigDecimal bigValue = BigDecimal.valueOf(originalValue);  bigValue = new BigDecimal(bigValue.toBigIntegerExact());  return new JsonPrimitive(bigValue);  }  })  .create();  }    public static boolean checkParameter(String[] args) {  if (args.length != 2) {  System.err.println(usage);  return false;  }    if (!(args[1].toLowerCase().endsWith(".lcpl"))) {  System.err.println("signatureValidator:Error: Invalid License File Input.\n" + usage);  return false;  }    File root = new File(args[0]);  if (!root.exists()) {  System.err.println("signatureValidator:Error: Root Certificate File Not Found.\n" + usage);  return false;  }    File file = new File(args[1]);  if (!file.exists()) {  System.err.println("signatureValidator:Error: License File Not Found.\n" + usage);  return false;  }  return true;  }    public static X509Certificate stringToCertificate(String certString) throws CertificateException {  InputStream is = null;  CertificateFactory cf;  try {  cf = CertificateFactory.getInstance("X.509");  if (certString.startsWith("-----BEGIN CERTIFICATE-----")) {  is = new ByteArrayInputStream(certString.getBytes());  } else {  is = new ByteArrayInputStream(Base64.decode(certString.getBytes()));  }  X509Certificate cert = (X509Certificate)cf.generateCertificate(is);    return cert;    } catch (CertificateException e) {  return null;  } finally {  try {  is.close();  } catch (IOException e) {  return null;  }  }  }    private static X509Certificate temp = null;  public static List<X509Certificate> getSortedItem(X509Certificate self, List<X509Certificate> signers, X509Certificate root) {  List<X509Certificate> tempList = new ArrayList<>();  tempList.addAll(signers);    List<X509Certificate> itemList = new ArrayList<>();  temp = self;  itemList.add(temp);  int length = tempList.size();    for (int i = 0; i < length; i++) {  for (X509Certificate item : tempList) {  if (temp.getIssuerDN().equals(item.getSubjectDN())) {  temp = item;  itemList.add(temp);  tempList.remove(item);  break;  }  }  }  itemList.add(root);    boolean check = false;  try {  for (int i = 0; i < itemList.size(); i++) {  X509Certificate child = itemList.get(i);  X509Certificate signer = itemList.size()!=(i+1) ? itemList.get(i+1) : itemList.get(i);  child.verify(signer.getPublicKey());  }  check = true;  } catch (Exception e) {  System.err.println(e.getMessage());  }  if (!check) {  System.err.println("signatureValidator:Error: Certificate verifyed failed.\n");  return null;  }    if (length != (itemList.size() -2)) {  System.err.println("signatureValidator:Error: certificate chain error.");  return null;  }  return itemList;  }  } |

1. http://www.w3.org/2001/04/xmlenc# [↑](#footnote-ref-2)
2. http://www.w3.org/2000/09/xmldsig# [↑](#footnote-ref-3)