CMLA Information

CMLA Technical Specification

Date: 2013-09-18

Version: V1.43-20131218

STATUS: APPROVED

CMLA - Contact Information

CMLA-Services @cm-la.com

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

NOTICE

CMLA TECHNICAL SPECIFICATION is the Proprietary CMLA, LLC., information that may be used for review and comment purposes only. ANY OTHER USE REQUIRES A SEPARATE LICENSE AGREEMENT BY AND BETWEEN USER AND CMLA, LLC AND WITHOUT SUCH LICENSE AGREEMENT ALL OTHER USE IS EXPRESSLY PROHIBITED.

DISCLAIMERS:

THIS APPROVED CMLA TECHNICAL SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY. CMLA, INTEL, NOKIA, PANASONIC AND SAMSUNG, DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO THE USE OF THE INFORMATION IN THIS APPROVED CMLA TECHNICAL SPECIFICATION AND TO THE IMPLEMENTATION OF INFORMATION IN THIS APPROVED CMLA TECHNICAL SPECIFICATION. CMLA, INTEL, NOKIA, MEI/PANASONIC AND SAMSUNG, DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

WITHOUT LIMITATION, CMLA, INTEL, NOKIA, PANASONIC AND SAMSUNG DISCLAIM ALL LIABILITY FOR COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY OR OTHERWISE, ARISING IN ANY WAY OUT OF USE OR RELIANCE UPON THIS SPECIFICATION OR ANY INFORMATION HEREIN.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

Copyright © 2013 CMLA, LLC.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

INDEX

1. INTRODUCTION 5

2. REFERENCES 5

3. DEFINITIONS 6

4. ABBREVIATIONS 7

5. OVERVIEW 9

5.1 PKI System Deliverables 9
5.2 Future Extensions 11

6. CERTIFICATES 12

6.1 CMLA Root CA Certificate 13
6.2 Device CA Certificate 14
6.3 Rights Issuer CA Certificate 15
6.4 OCSP Responder Certificate 16
6.5 Device Certificate 17
6.6 Rights Issuer Certificate 19

7. OCSP PROFILE 20

7.1 OCSP Requests 20
7.2 OCSP Responses 21

8. DEVICE CRL, RIGHTS ISSUER CRLS AND CMLA ARLS 22

8.1 Device CRL 22
8.1.1 CertificateRange CRL Entry Extension 23
8.1.2 Certificate Issuer CRL Entry Extension 23
8.2 Rights Issuer CRL 24
8.3 CMLA ARL 24

9. DEVICE CRL,CMLA ARL AND DEVICE CA CERTIFICATE DELIVERY MECHANISM 25

10. DEVICE CREDENTIALS DELIVERY 26

10.1 Key and Certificate Delivery Procedure 26
10.2 Device Credentials DVD-R 27
10.3 Rights Issuer CA Certificates file 28
10.4 PKCS#12 File Format 28

11. RIGHTS ISSUER CERTIFICATION 31

12. CMLA IP 32

12.1 Overview 33
12.2 CMLA Data-Dependent Transformation 34
12.2.1 CMLA_DDT_Perm 34
12.2.2 CMLA_DDT_Perm_Inv 35
12.2.3 CMLA_DDT_Exp 36
12.2.4 CMLA_DDT_Exp_Inv 36
12.3 CMLA Key Derivation Function 37
12.3.1 CMLA_KDF 37
12.4 CMLA Key Wrap Scheme 37
12.4.1 CMLA_WRAP 37
12.4.2 CMLA_UNWRAP 37
12.5 CMLA RSA Encryption Scheme 38
12.5.1 CMLA_RSA_ENCRYPT 38
12.5.2 CMLA_RSA_DECRYPT 38

13. CMLA PARAMETERS 40

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

14. DEVELOPMENT KEYS AND CERTIFICATES 40

15. CMLA MOBILE BROADCAST (SIGNED ADDENDUM REQUIRED) 41

15.1 OVERVIEW 41
15.2 ROT ALGORITHM DESCRIPTOR / TRUST AUTHORITY ALGORITHM DESCRIPTOR 41
15.2.1 Generic Syntax 41
15.2.2 CMLA-specific Values 42
15.3 KEYSET PROTECTION DETAILS 43

16. CONTENT PROTECTION TECHNOLOGY SYSTEM RENEWABILITY MESSAGE (CPT-SRM) SUPPORT 44

17. CMLA DECE IMPLEMENTATION (SIGNED ADDENDUM REQUIRED) 45

17.1 OVERVIEW 45
17.2 OMA COMPLIANCE REQUIREMENTS 46
17.3 DEVICE 47
17.3.1 DECE CMLA OMA Domain Manager Interaction 48
17.3.1.1 Device Attestation 48
17.3.1.2 ROAP Identification Protocol 48
17.3.1.3 Device Join Procedure 50
17.3.1.4 Domain Leave Procedure 51
17.3.2 Device DSP Interaction 51
17.4 CMLA DOWNLOAD SERVICE PROVIDER 52
17.4.1 Rights Issuer Certificate for Download Service Providers 53
17.5 CMLA DRM DOMAIN MANAGER 54
17.5.1 Rights Issuer Certificate for Domain Managers 54
17.6 DECE DOMAIN ROS 55
17.6.1 Domain RO with Key for both Audio and Video Tracks 55
17.6.2 Domain RO with separate keys, one for Audio and one for Video Track 57
17.7 CMLA-PSSH ELEMENTS 59
17.7.1 CMLA-PSSH Box 59
17.7.2 CMLA-PSSH UUID 59
17.7.3 CMLA-PSSH Contents 59
17.8 CMLA OBJECT IDENTIFIERS FOR DECE 60
17.9 CMLA XML SCHEMA FOR DECE 60

A1. CMLA DDT EXP 61

A2. CMLA DDT PERM 63

B1. CMLA IP TEST VECTOR 66

B2. CMLA IP TEST VECTOR 67

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

1. Introduction

OMA DRM Release 2 specifications [OMADRM-v2] define the end-to-end protocol for protected content distribution from Rights Issuers to Devices in a secure manner. These specifications rely on the existence of a Public Key Infrastructure to provide certain security services such as key and certificate provisioning mechanisms, certificate status checking, and the overall PKI hierarchy. The OMA specification builds on top of the assumption that every Device and Rights Issuer has one or more private keys, certificates and trust anchors. This specification defines the details of the CMLA deployment of OMA DRM technology.

The target audience is Client Adopters and Service Providers implementing CMLA based products and services. The reader is recommended to familiarize herself with the overall CMLA trust model that this technical specification is only one part of.

This specification supplements but does not replace OMA DRM Release 2 specifications.

2. References

[18Crypt] 18Crypt Profile specified in ETSI TS 102474 V1.1.1: Digital Video Broadcasting (DVB): IP Datacast over DVB-H: Service Purchase and Protection or most recent version.

[CMLAUVXSD] cmlauv schema cmla-uv-1_0.xsd at www.cm-la.com/tech/cmla-xml/cmla-uv-1_0.xsd

[DECECFF] Common File Format & Media Formats Specification Public Version 1.0.7r2 30 October 2013.

[DECECOR] DECE Coordinator API Specification 1.0.7r1 30 October 2013

[DECEDEV] DECE Device Specification Public 1.0.7r2 30 October 2013.

[DECESYS] DECE System Specification Public 1.0.7r1 30 October 2013.

[IEC62455] IEC 62455 First Edition 2007-06: Internet protocol (IP) and transport stream (TS) based service access or most recent version.

[OMABCAST-SCPv1] DRM Profile specified in OMA BCAST specification OMA-TS-BCAST_SvcCntProtection-V1_0: Service and Content Protection for Mobile Broadcast Services.

[OMADRM-XBS] OMA-TS-DRM_XBS-V1_0: OMA DRM V2.0 Extensions for Broadcast Support.

[FIPS180-2] "FIPS 180-2 - Secure Hash Standard", NIST, August 2002, URL: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[FIPS197] "FIPS 197 – Advanced Encryption Standard", NIST, November 2001, URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[ISO18033-2D] "ISO/IEC 18033-2: Information technology – Security techniques - Encryption algorithms - Part 2: Asymmetric Ciphers", Draft, Victor Shoup, January, 2004.

[OMADRM-v2] "Digital Rights Management", Open Mobile Alliance™, Version 2.0, OMA-DRM-DRM-v2_0, URL: http://www.openmobilealliance.org

[OMADRM-V2.1] "Digital Rights Management", Open Mobile Alliance™, Version 2.1, OMA-DRM-DRM-V2.1, URL:http://www.openmobilealliance.org

[OMAOCSP] "Online Certificate Status Protocol Mobile Profile", Version v1.0, OMA-WAP-OCSP-V1_0-20040127-C, URL: http://www.openmobilealliance.org

[PKCS#1] "PKCS #1 v2.1: RSA Cryptography Standard", RSA Laboratories, June 2002

[PKCS#8] "PKCS #8: Private-Key Information Syntax Standard", RSA Laboratories, November 1993

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

[PKCS#9] "PKCS #9 v2.0: Selected Object Classes and Attribute Types", RSA Laboratories, February 2000

[PKCS#10] "PKCS #10 v1.7: Certification Request Syntax Standard", RSA Laboratories, May 2000

[PKCS#12] "PKCS #12 v1.0: Personal Information Exchange Syntax", RSA Laboratories, June 1999

[RFC2560] "X.509 Internet Public Key Infrastructure – Online Certificate Status Protocol – OCSP", M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, June 1999. URL: http://www.ietf.org/rfc/rfc2560.txt

[RFC2616] "RFC 2616 – Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et al, June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] "RFC 2617 – HTTP Authentication: Basic and Digest Access Authentication", J Franks et al, June 1999. URL: http://www.ietf.org/rfc/rfc2617.txt

[RFC3279] "RFC 3279 - Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", R. Housley, W. Ford, W. Polk, D.Solo, April 2002. URL: http://www.ietf.org/rfc/rfc3279.txt

[RFC3280] "RFC 3280 - Internet X.509 Public Key Infrastructure – Certificate and CRL Profile", R. Housley, W. Ford, W. Polk, D.Solo, April 2002. URL: http://www.ietf.org/rfc/rfc3280.txt

[RFC3369] "Cryptographic Message Syntax (CMS)", R. Housley, August 2002. URL: http://www.ietf.org/rfc/rfc3369.txt

[RFC1738] "Uniform Resource Locators", T. Berners-Lee *et al*, December 1993. URL: http://www.ietf.org/rfc/rfc1738

[XMLDSIG] Schema for XML Signatures, Rev 1.1, www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd

3. Definitions

Client Adopter	As defined in the CMLA agreements
Client Adopter Transport Key	RSA key pair that is used to protect the confidentiality of Device Credentials DVD-R
CMLA DECE Domain	OMA DRM domain that has been established by the CMLA DECE DRMDM and for which Domain ROs are issued by CMLA DECE DSPs.
CMLA-DM	An OMA DRM Rights Issuer that manages CMLA domains in the DECE ecosystem. The RI certificate for the CMLA DRMDM has the ExtKeyUsage extension that contains key purposes *cmla-kp-dece-ricd* and *cmla-kp-dece-dm*.
CMLA-DSP	An OMA DRM Rights Issuer that manages CMLA domains in the DECE ecosystem. The RI certificate for the CMLA DRMDM has the ExtKeyUsage extension that contains key purposes *cmla-kp-dece-ricd* and *cmla-kp-dece-dsp*.
CMLA Transport Key	RSA key pair that is used to protect the integrity and authenticity of Device Credentials Delivery
CPT-SRM	Content Protection Technology System Renewability Message generally referred to as System Renewability Message (SRM) but within this specification is defined as CPT-SRM so as not to be confused with OMA SRM terminology

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

DECE RIs	Refers the CMLA-DM and CMLA-DSPs
Device	As defined in [OMADRM-v2]
Device CA	A CA that issues certificates for Devices
Device Credentials DVD-R	A DVD-R that contains security credentials for Client Adopters such as Device certificates and private keys
DRM Time	As defined in [OMADRM-v2]
Key Transport Algorithm	An algorithm used to transport key material to a Device, protected with the public key of the Device.
OCSP Responder	An entity producing OCSP responses
Rights Issuer	As defined in [OMADRM-v2]
Rights Issuer CA	A CA that issues certificates for Rights Issuers
Root CA	A trusted CA that is topologically on the top of the CMLA PKI hierarchy
Service Provider	As defined in the CMLA agreements
Tag Length Format	A syntax defined in the references [18Crypt], [IEC62455], and [OMABCAST-SCPv1] used to hold keyset blocks.

4. Abbreviations

AES	Advanced Encryption Standard
AES-WRAP	As defined in [OMADRM-v2] section 7.1.3
ARL	Authority Revocation List
BCRO	Broadcast Rights Object
CA	Certificate Authority
CFF	Common File Format [DECECFF]
CRL	Certificate Revocation List
CMLA_DDT_Exp	CMLA Data-dependent Transformation with Exponentiation
CMLA_DDT_Perm	CMLA Data-dependent Transformation with Permutation
CMLA-DECE-Domain	CMLA DECE Domain
CMLA_KDF	CMLA Key Derivation Function
CMLA_RSA_Decrypt	CMLA RSA Decryption
CMLA_RSA_Encrypt	CMLA RSA Encryption
CMLA_UNWRAP	CMLA Key Unwrap Scheme
CMLA_WRAP	CMLA Key Wrap Scheme
CMLA-DM	CMLA OMA DRM specific version of the DECE Domain Manager
CMLA-DSP	CMLA OMA DRM specific version of the DECE Download Service Provider
DECE	Digital Entertainment Content Ecosystem
DP	Device Public Key
DRM	Digital Rights Management

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

DVD-R	Digital Versatile Disc - Recordable
IETF	Internet Engineering Task Force
KDF	Key Derivation Function
OCSP	Online Certificate Status Protocol
OMA	Open Mobile Alliance
OMA_KDF	Key Derivation Function as defined in [OMADRM-v2] section 7.1.2
MAC	Message Authentication Code
MIME	Multipurpose Internet Mail Extensions
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
PSSH	Protection System Specific Header
REK	Rights Encryption Key
RFC	Request For Comments
ROAP	Rights Object Acquisition Protocol
ROT	Root Of Trust
RSA	Rivest, Shamir, Adleman
RSA.DECRYPT	As defined in [OMADRM-v2]
RSA.ENCRYPT	As defined in [OMADRM-v2]
SK	Session Key
TAA	Trust Authority Algorithm
TLF	Tag Length Format

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

5. Overview

5.1 PKI System Deliverables

The figure below illustrates the different entities producing output deliverables from the CMLA PKI system such as certificates, keys, CRLs and OCSP responses.

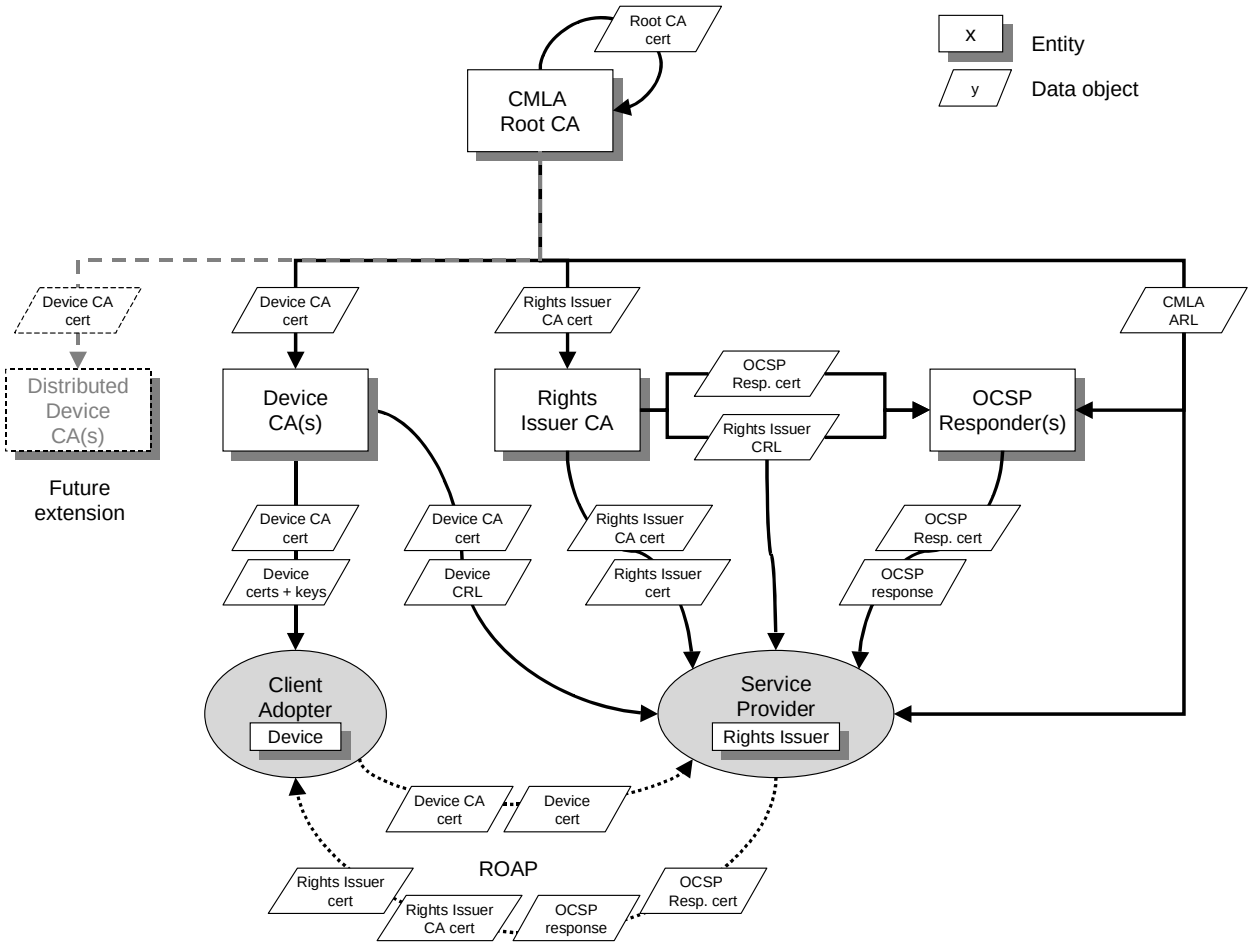


Figure 1. CMLA PKI System

Outputs from the CMLA PKI system are used by different entities according to the table below.

CMLA PKI output	Usage
Root CA certificate	Root CA certificate is a self-signed CA certificate that is used by Devices and Rights Issuers as a trust anchor to validate other CMLA certificates and the CMLA ARL. Root CA certificate is delivered to Client Adopters and Service Providers out-of-band.
Device certificate and private key	Device certificates are issued by a Device CA. The certificates and corresponding private keys are delivered to the Client Adopters in a Device Credentials DVD-R. Client Adopter stores them inside the Device. Device certificate is sent, upon request, from the Device to the Rights Issuer as part of the Device's certificate chain. Device certificate is validated and used by the Rights Issuer.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

Device CA certificate	Device CA certificate is issued by the CMLA Root CA and stored inside the Device as part of the Device's own certificate chain. Device CA certificate is sent, upon request, from the Device to the Rights Issuer as part of the Device's certificate chain. Device CA certificate is validated and used by the Rights Issuer.
Rights Issuer certificate	Rights Issuer certificate is issued by the Rights Issuer CA. Rights Issuer certificate is sent, upon request, from the Rights Issuer to the Device as a part of the Rights Issuer's certificate chain.
Rights Issuer CA certificate	Rights Issuer CA certificate is issued by the CMLA Root CA and stored inside the Rights Issuer as a part of Rights Issuer's certificate chain. Rights Issuer CA certificate is sent, upon request, from the Rights Issuer to the Device as a part of the Rights Issuer's certificate chain.
OCSP response	OCSP response is created and digitally signed by the OCSP Responder. The OCSP response is delivered to the Device through the Rights Issuer as a proof for the Rights Issuer's certificate's revocation status and for the purpose of time synchronization of the Device.
OCSP Responder certificate	The OCSP Responder certificate is issued by the Rights Issuer CA and stored inside the OCSP Responder. The OCSP Responder certificate is delivered to the Device through the Rights Issuer with the OCSP response, upon request by Device.
Device CRL	Device CRL is issued by the Device CA and it contains revocation status information for Device certificates issued by that Device CA. The Device CRL may, in the future, also be an indirect CRL that contains revocation status information for other Device CAs than the one issuing the Device CRL. The Device CRL is delivered to and used by the Rights Issuer to verify the revocation status of a Device certificate.
Rights Issuer CRL	Rights Issuer CRL is issued by the Rights Issuer CA and it contains revocation status information for Rights Issuer certificates issued by that CA. It is delivered to and used by the OCSP Responder to produce OCSP responses and Rights Issuers for verifying the revocation status of OCSP Responder certificates.
CMLA ARL	CMLA ARL is issued by the CMLA Root CA and delivered to and used by the Rights Issuer (to verify the revocation status of the Device CA).

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

5.2 Future Extensions

CMLA may start to support in the future a more distributed PKI model where Client Adopters are allowed to host Distributed Device CAs by themselves. However, as CMLA needs to have full control over the Device revocation process it is expected that all CRL issuing is done centrally by CMLA. In that setting one of the existing CMLA hosted Device CAs will be issuing CRLs for distributed Device CAs.

In order to make transition to distributed model as smooth as possible Service Providers are required to support the notion of indirect CRLs.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6. Certificates

The structure of all CMLA certificates is based on RFC 3280. All CMLA certificates SHALL be DER encoded. All attributes of the issuer and subject fields SHALL be encoded as type UTF8String.

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.1 CMLA Root CA Certificate

CMLA Root CA certificate is a self-signed CA certificate that is used by Devices and Rights Issuers as a trust anchor to validate other CMLA certificates and the CMLA ARL. Root CA certificate is delivered to Client Adopters and Service Providers out-of-band.

CMLA Root CA Certificates SHALL be based on OMA DRM specification appendix E.3 “CA Certificates” [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the *subject* field
Validity	Expires 31st December 2034. CMLA may in the future create a new Root CA or recertify the existing Root CA.
Subject	Contains attributes: - *countryName* (e.g. “US”) - *organizationName* (e.g. “CMLA”) - *commonName* (e.g. “Root CA”)
SubjectPublicKeyInfo	Carries a 2048 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Mandatory, critical. The boolean *cA* field is set to TRUE. The *pathLenConstraint* field is not used.
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bits *keyCertSign* and *cRLSign* are set.
AuthorityKeyIdentifier extension	Not used
SubjectKeyIdentifier extension	Mandatory, non-critical. The key identifier method (1) is used as defined in [RFC3280] (160-bit SHA-1 hash of the *subjectPublicKey* field, excluding the tag, length, and number of unused bits).
CertificatePolicies extension	Not used
ExtKeyUsage extension	Not used
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.2 Device CA Certificate

Device CA certificate is issued by the CMLA Root CA and stored inside the Device as part of the Device's own certificate chain. Device CA certificate is sent, upon request, from the Device to the Rights Issuer as part of the Device's certificate chain. Device CA certificate is validated and used by the Rights Issuer.

Device CA Certificates SHALL be based on OMA DRM specification appendix E.3 "CA Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the corresponding CMLA Root CA *subject* field.
Validity	Expires 30 years from the issuance date but not later than the certificate of the corresponding CMLA Root CA Certificate.
Subject	Contains attributes: - *countryName* (e.g. "US") - *organizationName* (e.g. "CMLA") - *commonName* (e.g. "Device CA 1")
SubjectPublicKeyInfo	Carries a 2048 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Mandatory, critical. The boolean *cA* field is set to TRUE. The *pathLenConstraint* field is set to zero.
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bits *keyCertSign* and *cRLSign* are set for CMLA hosted Device CAs. For distributed Device CAs, CMLA may decide not to use cRLSign
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding CMLA Root CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Mandatory, non-critical. The key identifier method (1) is used as defined in [RFC3280] (160-bit SHA-1 hash of the *subjectPublicKey* field, excluding the tag, length, and number of unused bits).
CertificatePolicies extension	Not used
ExtKeyUsage extension	Not used
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.3 Rights Issuer CA Certificate

Rights Issuer CA certificate is issued by the CMLA Root CA and stored inside the Rights Issuer as a part of Rights Issuer's certificate chain. Rights Issuer CA certificate is sent, upon request, from the Rights Issuer to the Device as a part of the Rights Issuer's certificate chain.

Rights Issuer CA Certificates SHALL be based on OMA DRM specification appendix E.3 "CA Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the corresponding CMLA Root CA *subject* field.
Validity	Expires 30 years from the issuance date but not later than the certificate of the corresponding CMLA Root CA Certificate.
Subject	Contains attributes: - *countryName* (e.g. "US") - *organizationName* (e.g. "CMLA") - *commonName* (e.g. "Rights Issuer CA 1")
SubjectPublicKeyInfo	Carries a 2048 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Mandatory, critical. The boolean *cA* field is set to TRUE. The *pathLenConstraint* field is set to zero.
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bits *keyCertSign and cRLSign* are set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding CMLA Root CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Mandatory, non-critical. The key identifier method (1) is used as defined in [RFC3280] (160-bit SHA-1 hash of the *subjectPublicKey* field, excluding the tag, length, and number of unused bits).
CertificatePolicies extension	Not used
ExtKeyUsage extension	Not used
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.4 OCSP Responder Certificate

The OCSP Responder certificate is issued by the Rights Issuer CA and stored inside the OCSP Responder. The OCSP Responder certificate is delivered to the Device through the Rights Issuer with the OCSP response, upon request by Device.

OCSP Responder Certificates SHALL be based on OMA DRM specification appendix E.4 "OCSP Responder Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*V3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the corresponding Rights Issuer CA *subject* field.
Validity	Expires 3 months from the issuance date but not later than the certificate of the corresponding Rights Issuer CA Certificate.
Subject	Contains attributes: - *countryName* (e.g. "US") - *organizationName* (e.g. "CMLA") - *commonName* (e.g. "OCSP Responder 1")
SubjectPublicKeyInfo	Earries a 1024 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Not used
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bit *digitalSignature* is set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding Rights Issuer CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Mandatory, non-critical. The key identifier method (1) is used as defined in [RFC3280] (160-bit SHA-1 hash of the *subjectPublicKey* field, excluding the tag, length, and number of unused bits).
CertificatePolicies extension	Not used
ExtKeyUsage extension	Mandatory, critical. The extension contains only the *id-kp-OCSPSigning* object identifier.
Id-pkix-ocsp-nocheck extension	Mandatory, non-critical. The extension value is NULL.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.5 Device Certificate

Device certificates are issued by a Device CA. The certificates and corresponding private keys are delivered to the Client Adopters in a Device Credentials DVD-R. Client Adopter stores them inside the Device. Device certificate is sent, upon request, from the Device to the Rights Issuer as part of the Device's certificate chain. Device certificate is validated and used by the Rights Issuer.

Device certificates SHALL be based on OMA DRM specification appendix E.1 "DRM Agent Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
SerialNumber	Non-negative integers that are unique within the set of all CMLA issued Device certificates. The DER encoded bytes of the *serialNumber* field are structured as follows: *Bytes 1-2:* CA Identifier. The first byte must not be 00h or have its most significant bit set. *Bytes 3-6:* Certificate batch identifier that is unique within the specific Device CA. *Bytes 7-20:* 14 least significant bytes of SHA-1 hash calculated over the certified public key and current date and time. The serial numbering scheme may be used in the Device CRLs to revoke a batch delivery of keys and certificates using the CRL mechanism defined in chapter 8.
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the issuing Device CA Certificate's *subject* field.
Validity	Expires 20 years from the issuance date but not later than the certificate of the corresponding Device CA Certificate.
Subject	Contains always attributes: - *organizationName* as assigned by CMLA for the specific Client Adopter - *serialNumber* as assigned by CMLA by calculating SHA-1 hash over the certified public key May contain additional attributes as defined in [OMADRM-v2]. The values of those fields are assigned by the Client Adopter.
SubjectPublicKeyInfo	Carries a 1024 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Not used
CRLDistributionPoints extension	(a) Certificates issued by a CMLA managed Device CA: Not used. (b) Certificates issued by a distributed Device CAs (future extension): Mandatory, non-critical. The extension contains only one *cRLIssuer* field that defines the distinguished name of the Device CRL Issuer being responsible for producing the CRLs governing this certificate.
KeyUsage extension	Mandatory, critical. Only key usage bits *digitalSignature* and *keyEncipherment* are set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding Device CA

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

	Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Not used
CertificatePolicies extension	Not used
ExtKeyUsage extension	Mandatory, critical. The extension contains only the *oma-kp-drmAgent* object identifier.
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

6.6 Rights Issuer Certificate

Rights Issuer certificate is issued by the Rights Issuer CA. Rights Issuer certificate is sent, upon request, from the Rights Issuer to the Device as a part of the Rights Issuer's certificate chain.

Rights Issuer Certificates SHALL be based on OMA DRM specification appendix E.2 "Rights Issuer Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the issuing Rights Issuer CA Certificate's *subject* field.
Validity	Expires 5 years from the issuance date but not later than the certificate of the corresponding Rights Issuer CA Certificate.
Subject	Contains always attributes: - *organizationName* (e.g. "CoolService Ltd.") as assigned by CMLA for the specific Service Provider May contain additional attributes as defined in [OMADRM-v2]. The values of those fields are assigned by the Service Provider.
SubjectPublicKeyInfo	Carries a 1024 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Not used
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bit *digitalSignature* is set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding Rights Issuer CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Not used
CertificatePolicies extension	Not used
ExtKeyUsage extension	Mandatory, critical. The extension contains only the *oma-kp-rightsIssuer* object identifier.
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

7. OCSP Profile

The OCSP service shall be based on and compliant with the OMA OCSP profile [OMAOCSP]. Whilst that document takes a device-centric view, this section specifies the service that shall be provided. The OCSP transport binding to HTTP, as specified in Appendix A of [RFC2560], SHALL be used. The OCSP Responder SHALL authenticate the OCSP requestor using HTTP digest authentication method defined in [RFC2617]. The username and password are provided to the Service Provider by CMLA.

A Rights Issuer certificate chain consists of the Rights Issuer leaf certificate and the Rights Issuer CA certificate. The Device MUST check, as specified in [OMADRM-v2], the revocation status of the Rights Issuer certificate. The Device is not required to check the revocation status of the Rights Issuer CA certificate. In other words, the complete set of OCSP responses required to check the revocation status of the whole Rights Issuer certificate chain consists of only one OCSP response vouching for the revocation status of the Rights Issuer leaf certificate.

7.1 OCSP Requests

OCSP requests are made by Rights Issuers on behalf of the Devices.

OCSP Request Field	Value
TbsRequest	Fields as defined below
Version	v1 (integer value 0)
RequestorName	Optional. May be included in request, but will be ignored by the OCSP responder.
RequestList	SHOULD contain only one *Request* ASN.1 structure
Request	SHOULD contain only one *ReqCert* ASN.1 structure
ReqCert	CertID fields as defined below
Hash-Algorithm	*id-sha1* as defined in [RFC3279].
IssuerName-Hash	Hash of Issuer's DN. The hash value SHALL NOT be truncated.
IssuerKey-Hash	Hash of Issuer's public key. The hash value SHALL NOT be truncated.
Serialnumber	Certificate serial number
SingleRequest-Extension	Not used.
RequestExtensions	Extensions that have not been defined below SHOULD NOT be used.
Nonce (id-pkix-ocsp-nonce)	Optional. Non-critical. If given, the nonce value will be repeated in the response. The value of the nonce originates from the device as defined in [OMADRM-v2].
OptionalSignature	Optional May be included in request, but will be ignored by OCSP responder.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

7.2 OCSP Responses

OCSP Response Field	Value
ResponseStatus	The status for the response (*successful, malformedRequest, internalError, tryLater, sigRequired, unauthorized* as appropriate).
ResponseBytes	*ResponseBytes* fields as defined below
ResponseType	*id-pkix-ocsp-basic*
Response	OCTET STRING consisting of *BasicOCSPResponse* as defined below
BasicOCSPResponse	
TbsResponseData	ResponseData fields as defined below
Version	v1 (integer value 0)
ResponderID	The KeyHash method shall be used (and not the Name) method. The KeyHash SHALL be calculated as specified in [RFC2560], so that it matches the subjectKeyIdentifier extension of the OCSP responder's certificate.
ProducedAt	Time of producing this OCSP Response
Responses	*SingleResponse* fields as defined below
CertID	*CertID* fields as defined below
HashAlgorithm	*id-sha1* as defined in [RFC3279]. The hash values SHALL NOT be truncated.
IssuerName-Hash	Hash of Issuer's DN
IssuerKeyHash	Hash of Issuer's public key
Serialnumber	Certificate serial number
CertStatus	Choice of *Good*, *revoked*, *unknown* ASN.1 structures as appropriate.
ThisUpdate	These will be taken from the *thisUpdate* field of the CRL on which the response is based.
NextUpdate	These will be taken from the *nextUpdate* field of the CRL on which the response is based.
SingleExtensions	Not used.
Response-Extensions	Extensions that have not been defined below SHOULD NOT be used.
Nonce (id-pkix-ocsp-nonce)	Optional. Non-critical. If given, the nonce value will be repeated in the response. The value of the nonce originates from the device as defined in [OMADRM-v2].
SignatureAlgorithm	sha1WithRSAEncryption as defined in [RFC3279].
Signature	Siganture bits as calculated according to the signature algorithm.
Certs	The OCSP responder certificate will be provided with each response given by the OCSP responder to the Rights Issuer. However, this certificate may be stripped from the OCSP response before the response is passed to the device. The RI CA and root CA certificates are not provided, since they must be present at the device already.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

8. Device CRL, Rights Issuer CRLs and CMLA ARLs

CMLA ARL is issued by the CMLA Root CA and delivered to and used by the Rights Issuer (to verify the revocation status of the Device CA).

The structure of CMLA CRLs and ARLs is based on [RFC3280]. All CRLs and ARLs SHALL be DER encoded. All attributes of the issuer and subject fields SHALL be encoded as type UTF8String.

```
CertificateList  ::=  SEQUENCE  {
     tbsCertList          TBSCertList,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertList  ::=  SEQUENCE  {
     version                 Version OPTIONAL,
                                  -- if present, MUST be v2
     signature               AlgorithmIdentifier,
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time OPTIONAL,
     revokedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          revocationDate          Time,
          crlEntryExtensions      Extensions OPTIONAL
                                        -- if present, MUST be v2
                                }  OPTIONAL,
     crlExtensions           [0]  EXPLICIT Extensions OPTIONAL
                                        -- if present, MUST be v2
                                }
```

8.1 Device CRL

Device CRL is issued by the Device CA and it contains revocation status information for Device certificates issued by that Device CA or Distributed Device CA(s). It is delivered to and used by the Rights Issuer to verify the revocation status of a Device certificate.

The Device CRL SHALL be based on IETF's [RFC3280].

CRL field name		Value
Version		*v2* (integer value 1)
Signature		*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer		Same as the CMLA managed Device CA Certificate's *subject* field.
ThisUpdate		The date of issuance
NextUpdate		180 hours. CMLA may modify this value based on operational considerations.
RevokedCertificates entries:		
	UserCertificate	Revoked certificate serial number
	RevocationDate	Date of revocation decision.
	CertificateRange CRL entry extension	Optional, critical. Used as defined below when batch revocation is needed. Rights Issuers MUST be able to support this extension.
AuthorityKeyIdentifier extension		Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the issuing Device CA Certificate's *subjectKeyIdentifier* extension value.
CRLNumber extension		Mandatory, non-critical. The value is a monotonically increasing sequence number.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

IssuingDistributionPoint	Optional, critical This extension is used by distributed Device CAs. The boolean *indirectCRL* field is set to TRUE. Rights Issuers MUST be able to support this extension.

8.1.1 CertificateRange CRL Entry Extension

The *CertificateRange* extension is a CMLA defined non-standard CRL entry extension that MAY be used in CRLs issued by CMLA. This extension can be used to revoke a batch of certificates identified with a certificate serial number range. The usefulness of this extension depends on the certificate serial numbering scheme and it is expected to be used in cases when a CMLA Device Credentials DVD-R has been compromised for some reason.

The extension defines a range of certificate serial numbers starting from the value of the *userCertificate* field of the specific CRL entry and ending in the value of the *lastRevokedCertificate* field value of the *CertificateRange* extension. All certificate serial numbers that are within these two integers (both ends inclusive) MUST be considered to be revoked.

The value of the *lastRevokedCertificate* field MUST be larger than the value of the *userCertificate* field.

The extension SHALL be marked as critical.

```
id-xx-CMLA-revokedCertificateRange OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1)
     private(4) enterprise(1) content-management-license-administrator(21888) 1 }

lastRevokedCertificate ::= CertificateSerialNumber
```

Value of CMLA OID	Name	Description
1.3.6.1.4.1.21888.1	*crl*	*Certificate revocation list*

8.1.2 Certificate Issuer CRL Entry Extension

The Certificate Issuer CRL Entry Extension SHALL be used for CRLs that have revoked certificates not issued by the CA issuing the CRL, as defined in [RFC3280]. This will be the case when certificates issued by a Distributed Device CA have been revoked.

The extension SHALL be marked as critical. Rights Issuers MUST be able to support this extension.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

8.2 Rights Issuer CRL

Rights Issuer CRL is issued by the Rights Issuer CA and it contains revocation status information for Rights Issuer certificates issued by that CA. It is delivered to and used by the OCSP Responder to produce OCSP responses and Rights Issuers for verifying the revocation status of OCSP Responder certificates.

The Rights Issuer CRL SHALL be based on IETF's [RFC3280].

CRL field name		Value
Version		*v2* (integer value 1)
Signature		*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer		Same as the issuing Rights Issuer CA Certificate's *subject* field.
ThisUpdate		The date of issuance
NextUpdate		180 hours. CMLA may modify this value based on operational considerations.
RevokedCertificates entries:		
	UserCertificate	Revoked certificate serial number
	RevocationDate	Date of revocation decision.
AuthorityKeyIdentifier extension		Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the issuing Rights Issuer CA Certificate's *subjectKeyIdentifier* extension value.
CRLNumber extension		Mandatory, non-critical. The value is a monotonically increasing sequence number.

8.3 CMLA ARL

The CMLA ARL SHALL be based on IETF's [RFC3280].

CRL field name		Value
Version		*v2* (integer value 1)
Signature		*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer		Same as the issuing CMLA Root CA Certificate's *subject* field.
ThisUpdate		The date of issuance
NextUpdate		8 months. CMLA may modify this value based on operational considerations.
RevokedCertificates entries:		
	UserCertificate	Revoked certificate serial number
	RevocationDate	Date of revocation decision.
AuthorityKeyIdentifier extension		Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the issuing CMLA Root CA Certificate's *subjectKeyIdentifier* extension value.
CRLNumber extension		Mandatory, non-critical. The value is a monotonically increasing sequence number.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

9. Device CRL,CMLA ARL and Device CA Certificate delivery mechanism

Device CRLs, Rights Issuer CRLs and CMLA ARLs are used in the CMLA context for verifying the revocation status of Device certificate chains and OCSP Responder Certificates. Device CA Certificates are used to verify the signature of Device CRLs. The same delivery mechanism defined in this chapter applies for all of them.

Service Providers can fetch CRLs, ARLs and Device CA Certificates by making an HTTP GET request to URLs that can be constructed as described below.

```
URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]
```

The following rules SHALL apply for the URLs:

The *host* name is a parameter defined by CMLA

The *abs_path* for a Device CRL SHALL be "/cmla/DeviceCRL/download.aspx" and the query string SHALL be "CN=*name*", where *name* is the URL [RFC1738] encoding of the *commonName* attribute of the Device CA's distinguished name.

- Example: "*http://examplehost.cm-la.com/cmla/DeviceCrl/download.aspx?CN=Device%20CA%201*"

- The *abs_path* for a Rights Issuer CRL SHALL be "/cmla/RightsCrl/download.aspx" and the query string SHALL be "CN=*name*", where *name* is the URL[RFC1738] encoding of the *commonName* attribute of the Rights Issuer CA's distinguished name. Example: "*http://examplehost.cm-la.com/cmla/RightsCrl/download.aspx?CN=Rights%20Issuer%20CA%201*" The *abs_path* for an ARL SHALL be "/cmla/ARL/download.aspx" and the query string shall be "CN=*name*", where *name* is the URL [RFC1738] encoding of the commonName attribute of the Root CA's distinguished name.

  - Example: "*http://examplehost.cm-la.com/cmla/ARL/download.aspx?CN=Root%20CA*"

The *abs_path* for Device CA Certificate SHALL be "/cmla/DeviceCACertificate/download.aspx" and the query string SHALL be "CN=*name*", where *name* is the URL [RFC1738] encoding of the commonName attribute of the Device CA's distinguished name.

  - Example: "*http://examplehost.cm-la.com/cmla/DeviceCACertificate/download.aspx?CN=Device%20CA%201*"

The URLs will be populated with the most recently issued CRL/ARL or Device CA Certificate.

The CRL/ARL and Device CA Certificate SHALL be delivered to the Service Provider in the body part of the HTTP response message for the HTTP GET request made to the specified URL. The MIME-type of the CRL/ARL SHALL be "`application/pkix-crl`". The MIME-type of the Device CA Certificate SHALL be "`application/pkix-cert`".

As defined in the [RFC3280], the CRL Issuer will issue a new CRL/ARL and make it available from the specified URL before the NextUpdate value of the most recent CRL/ARL has been passed. In addition to the periodic updates the CRL Issuer will issue a new CRL/ARL when new certificates have been revoked. Service Providers are recommended to attempt to pull the new CRL/ARL well in advance of the NextUpdate time of the most recent CRL/ARL, even though that may result in downloading the same CRL/ARL several times. The Service Provider is recommended to use the *If-Modified-Since* request header as defined in [RFC2616] chapter 14.25 to avoid unnecessary downloads of the same CRL/ARL.

The HTTP server SHALL authenticate the requesting entity using HTTP digest authentication method defined in [RFC2617]. The username and password are provided to the Service Provider by CMLA. In case of invalid username and/or password the HTTP error code 401 "Unauthorized" will be returned.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

10. Device Credentials Delivery

CMLA mechanism and format for the secure transfer of Client Adopter keys and certificates is based on the PKCS#12 standard. The Client Adopter keys and corresponding certificates are generated and certified by Device CA, and distributed securely from CMLA to the Client Adopter in a DVD-R.

10.1 Key and Certificate Delivery Procedure

The first two process steps are required to be completed before any Client Adopter key orders can be made. Both development and production keying material is sent to adopter on 4 DVDs: Production CMLA Root CA Certificate, Production Transport Keys, Development CMLA Root CA Certificate, and Development CMLA Transports Keys.

1. CMLA generates a CMLA Transport Key that is a 2048 bit RSA key pair and communicates the public key part to the Client Adopter during the process of agreement execution between CMLA and the Client Adopter. The CMLA Transport Key is used to protect the integrity and authenticity of the Device Credentials DVD-R. CMLA will not use the same CMLA Transport Key indefinitely and thus the Client Adopter must be able to replace the old CMLA Transport Key with a newer one, after notice from CMLA.
   The public parts of CMLA Transport Keys are delivered to the Client Adopter in a DVD-R that contains one file per each CMLA Transport Key. The file naming convention is "`CMLA_Transport_Key_##.der`", where ## is replaced with a two digit number. Each file contains the public key part of the 2048 bit RSA key pair of the CMLA Transport key as a DER encoding of the `RSAPublicKey` structure defined in [PKCS#1].
   CMLA delivers to the Client Adopter, in a secure out-of-band mechanism, a list of public key file names and corresponding SHA-1 hash values calculated over the file (20 bytes in hexadecimal representation). Client Adopter must verify that each hash value matches the file on the DVD-R.

2. CMLA delivers valid CMLA Root CA certificates to Client Adopters using a similar mechanism as above.
   The same DVD-R contains one file for each CMLA Root CA certificate. The file naming convention is "CMLA_Root_CA_Certificate_##.der", where ## is replaced with a two digit number. Each file contains the DER encoded CMLA Root CA certificate.
   CMLA delivers to the Client Adopter, in a secure out-of-band mechanism, a list of CMLA Root CA certificate file names and corresponding SHA-1 hash values calculated over the file (20 bytes in hexadecimal representation). Client Adopter must verify that each hash value matches the file on the DVD-R.

3. Client Adopter signs the Client Adopter Agreement Exhibit G that defines (a) the persons that are authorized to generate key orders, (b) one delivery address per authorized person and (c) one Client Adopter Transport Key per authorized person (the same key may be used by multiple authorized persons).
   The Client Adopter Transport Key is a 1024 bit RSA key pair and the Exhibit G includes the hexadecimal representation of the SHA-1 hash calculated over the DER encoding of the public key as stored in the file.
   The public keys are delivered to CMLA in a DVD-R that contains one file per each Client Adopter Transport Key. The file naming convention is:
   "`Client_Adopter_Transport_Key_*_##.der`", where * is replaced with the Client Adopter name and ## is replaced with a two digit number. Each file contains the DER encoding of the RSAPublicKey structure defined in [PKCS#1].

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

The ASN.1 definition for the RSAPublicKey structure from [PKCS#1] is given below.

```
RSAPublicKey ::= SEQUENCE {
   modulus           INTEGER,  -- n
   publicExponent    INTEGER   -- e
}
```

The subsequent steps define how the actual key order and delivery process.

4. Client Adopter sends its key order to CMLA:
   4.1 An authorized person of the Client Adopter submits key order to CMLA via Fax and a scanned email attachment or by signed email.
   4.2 CMLA will review and verify each order submitted, and if complete & accurate, generate an invoice.
   4.3 The Invoice is sent via email and hard copy to the Client Adopter.
   4.4 Upon receipt of payment on invoice, the key order is then processed and responded.
5. According to the key order, Device CA generates requested number of key pairs and certificates, packs these data and relevant authority certificates into one or more files based on PKCS#12 standard as described in the next sections. The files are burned to a DVD-R that is delivered to the Client Adopter. The Client Adopter validates the signature and decrypts the Device Credentials DVD-R files.
6. CMLA registers into certificate database necessary information such as the keys and certificates issued by Device CA, request data, and key and certificate delivery data.

10.2 Device Credentials DVD-R

The Device Credentials DVD-R contains the set of security credentials that is needed to manufacture CMLA Compliant Devices. The credentials are stored in the DVD-R in two types of files:

- Rights Issuer CA Certificates file
- Device Credentials files

The table below describes the file types in more detail.

File type	Contains	File naming convention	Comment
Rights Issuer CA Certificates	One or more Rights Issuer CA certificates.	“Rights_Issuer_CA_Certs.der”	There will be only one file of this type in the DVD-R. Typically there will be only one Rights Issuer CA certificate in this file, but the Client Adopter must not assume that. Client Adopters may choose not to store the Rights Issuer CA certificates to manufactured Devices i.e. the use of this file is optional.
Device Credentials	One Device CA certificate, and 1000 Device private key and certificate pairs (issued by the corresponding Device CA)	“Device_Credentials_*_####.p12” * is replaced with the hexadecimal encoding of the CA Identifier (2 bytes) and Batch number (4 bytes) #### is replaced with a four digit sequence number differentiating the files from each other. First file is identified with “0001” and the number is incremented by one for subsequent files. Example: “Device_Credentials_ 0002003F270C_0019.p12”	There will be one or more files of this type in the DVD-R. The maximum number of files is 1000. As each file contains 1000 Device private keys and certificates one DVD-R contains security credentials for a maximum of 1.000.000 Devices.

The format of each file is defined in the next sections.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

10.3 Rights Issuer CA Certificates file

The Rights Issuer CA certificates file contains one or more DER encoded Rights Issuer CA Certificates. Before using the Rights Issuer CA certificates on the DVD-R the Client Adopter must validate that each certificate chains back to the CMLA Root CA. Using the Rights Issuer CA certificates is optional for the Client Adopter.

10.4 PKCS#12 File Format

Device Credentials files are based on [PKCS#12] and [RFC3369].

Cryptographic Message Syntax (CMS) [RFC3369] stores all data using ContentInfo structures. Three ContentInfo types, `Data, SignedData` and `EnvelopedData` are used in the key and certificate delivery format.

- `Data` is just a wrapper for raw, unauthenticated, unencrypted arbitrary data.
- `SignedData` is a wrapper allowing a digital signature to be applied to arbitrary data, thereby giving authentication.
- `EnvelopedData` is a wrapper allowing the data to be encrypted using a symmetric message key, which in turn is encrypted under an RSA key. This gives confidentiality.

The cryptographic algorithms that are used are listed below.

- Message digest algorithm: FIPS 180-1 defines the SHA-1 message digest algorithm [FIPS180-2]. The algorithm identifier assigned by ISO is 1.3.14.3.2.26
- Asymmetric signature algorithm: PKCS#1v1.5 RSA signature with SHA1 hash has algorithm identifier = 1.2.840.113549.1.1.5 [PKCS#1]
- Asymmetric encryption algorithm: PKCS#1v1.5 RSA encryption has algorithm identifier = 1.2.840.113549.1.1.1 [PKCS#1]
- Symmetric encryption algorithm: AES with 128-bit keys, 128-bit block length, in CBC mode has algorithm identifier = 2.16.840.1.101.3.4.1.2 [FIPS197]

The top-level structure from PKCS#12 is a ‘PFX’.

```
PFX ::= SEQUENCE {
    version Integer,
    authSafe ContentInfo }
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

The overall PKCS #12 structure is illustrated in the figure below.

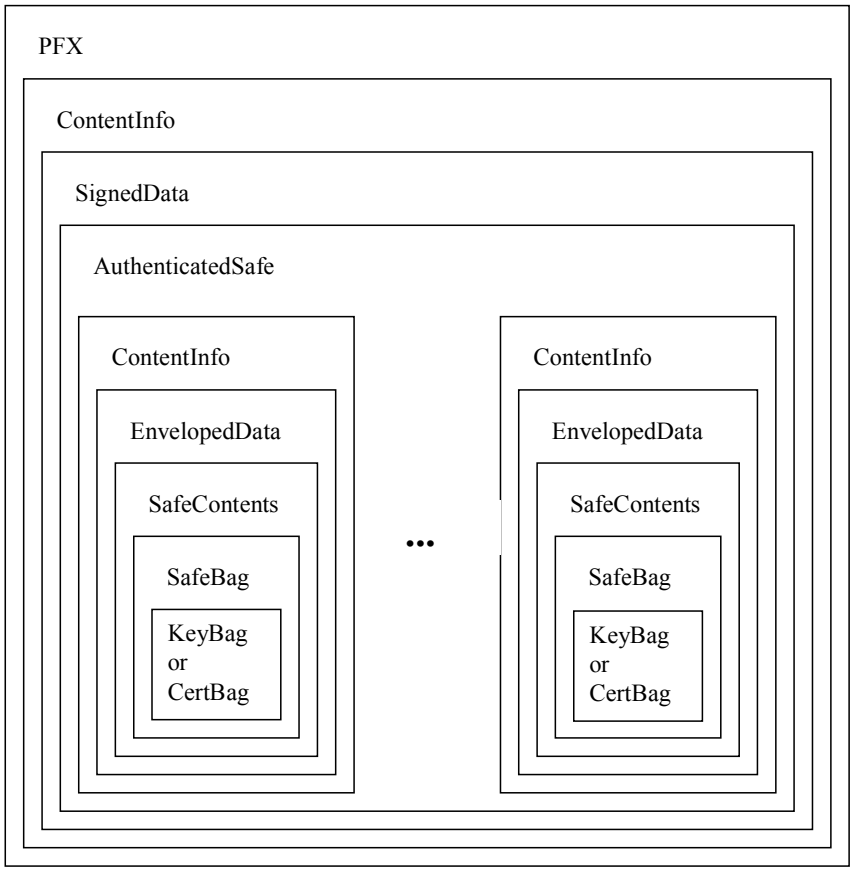


Figure 2. PKCS #12 Structure

The `ContentInfo` in the PFX comes from CMS [RFC3369], and SHALL contain a `SignedData`, which contains the encoding of an `AuthenticatedSafe` (below) as well as a digital signature and reference to the CMLA Transport Key used for a SignerIdentifier. SignerIdentifier is a SubjectKeyIdentifier which in turn is a SHA-1 hash over SubjectPublicKeyInfo. SubjectPublicKeyInfo shall be constructed from the CMLA Transport Key which has the RSAPublicKey structure (refer to section 4.1.2.7 of [RFC3280]). This top-level signature provides integrity and authenticity protection for all the data in the file.

The signed data is an `AuthenticatedSafe` structure:

```
AuthenticatedSafe ::= SEQUENCE OF ContentInfo
```

This SEQUENCE OF contains a number of ContentInfo structures, each containing an EnvelopedData structure. These in turn contain the encryption of a SafeContents, together with key identifier information referencing the Client Adopter Transport Key that is used for a RecipientIdentifier. RecipientIdentifier is a SubjectKeyIdentifier which in turn is a SHA-1 hash over SubjectPublicKeyInfo. SubjectPublicKeyInfo shall be constructed from the Client Adopter Transport Key which has the RSAPublicKey structure.

```
SafeContents ::= SEQUENCE OF SafeBag
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

Only one `SafeBag` structure will be present in each `SafeContents` structure.

```
SafeBag :: = SEQUENCE {
   bagId ObjectIdentifier,
   bagValue [0] EXPLICIT ANY DEFINED BY bagId,
   bagAttributes SET OF PKCS12Attribute OPTIONAL
}
```

The `bagValue` can contain either a `KeyBag` or a `CertBag` as identified by the `bagId`:

```
KeyBag ::= PrivateKeyInfo

CertBag ::= Sequnce {
  certId ObjectIdentifier,
  certValue  EXPLICIT ANY DEFINED BY certId }
```

where `PrivateKeyInfo` is as defined in [PKCS#8] and `certValue` is an octet string containing the DER encoding of an X.509 certificate.

For Device Credentials files: the first `ContentInfo` structure contains the Device CA certificate and subsequent `ContentInfos` contain Device private key and certificate pairs, each private key and certificate in separate `ContentInfo` structures.

For example, with two Device private keys and two corresponding Device certificates, there will be five `ContentInfo` structures, each containing an `EnvelopedData`, each in turn containing an encrypted `SafeContents` that contains a single `SafeBag`. These five SafeBags will contain, in sequence:

1. The Device CA Certificate issuing all Device certificates in this file
2. The first Device private key
3. The Device certificate corresponding to the first Device private key
4. The second Device private key
5. The Device certificate corresponding to the second Device private key

To assist with matching keys to certificates, the optional `bagAttributes` field of each `SafeBag` structure SHALL be used for each private key and end-entity certificate, containing identifiers unique to each key and certificate pair. The localKeyId mechanism from [PKCS#12] and [PKCS#9] shall be used, using an octet string formed from the SHA-1 hash of the public key (the computation method is irrelevant to the party receiving the file, they just need to match the localKeyId from the KeyBag to the localKeyId on the CertBag).

In general, a new session key could be used with each EnvelopedData in the PKCS#12 file. As a performance optimization, only a single session key shall be used, i.e. the same session key shall be re-used for each EnvelopedData in a single file (but shall never be re-used between two separate files). This gives a potential performance optimization, in that only one RSA decryption is required to obtain the session key, but existing tools not supporting this optimization are still able to process the PKCS#12 file without modification.

Each PKCS#12 file shall contain exactly 1000 pairs of Device private key and certificate. All key orders must therefore be made in multiples of 1000. The maximum order size shall be 1.000.000 key and certificate pairs. The same certificate batch number shall be used for all keys in the batch (and so shall be shared across all the certificates in all the files in that batch).

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

11. Rights Issuer Certification

As a prerequisite for issuing a Rights Issuer certificate for the Service Provider, it is assumed that the Service Provider has already generated a Rights Issuer RSA key pair in an environment that satisfies the security requirements stipulated by the Rights Issuer Robustness Rules, another prerequisite is for the Service Provider to have fulfilled the Exhibit G of the Service Provider Agreement and to have communicated it to CMLA.. After that has been done, the first time certification and subsequent re-certifications are done as defined below.

1. Service Provider creates self-signed DER encoded PKCS#10 certificate request [PKCS#10], burns a DVD-R containing the request, and sends the DVD-R to CMLA. This request SHALL be signed using sha-1WithRSAEncryption as defined in [RFC3279].
   The file naming convention is "`Rights_Issuer_*_##.p10`", where * is replaced with the Service Provider's name and ## is replaced with a two digit number.
   The Service Provider also submits the Rights Issuer Certificate ordering form to CMLA via signed email and the original via regular courier. . This ordering form will include the SHA1 hash of the public key (PKCS#1 RSAPublicKey structure in the subjectPublicKey field) inside the PKCS#10 request PKCS#10 file (20 bytes, encoded in hexadecimal).
2. CMLA will review and verify each request submitted. After verification, CMLA generates an invoice.
3. The Invoice is sent via email and hard copy to the Service Provider.
4. Service Provider makes payment on invoice.
5. Upon receipt of payment on invoice, CMLA processes order and responds. Order processing includes the verification of the hash value provided in the Rights Issuer Certificate ordering form to the hash value calculated over the to be certified public key. If none of the verifications fail CMLA creates, according to the certificate request, a Rights Issuer Certificate and delivers it with the corresponding Rights Issuer CA Certificate to the Service Provider in a DVD-R.
   The file naming convention for the Rights Issuer Certificate is "`Rights_Issuer_*_##.der`", where * is replaced with the Service Provider's name and ## is replaced with the two digit number. The file contains the DER encoding of the Rights Issuer Certificate
   The file naming convention for the Rights Issuer CA Certificate is "Rights_Issuer_CA_Certs.der". The file contains the DER encoding of the Rights Issuer CA Certificate.
6. CMLA registers into certificate database necessary information such as the certificate issued by the RI CA, request data, and certificate delivery data.

CMLA Root CA Certificates are delivered to Service Providers in the same way as to the Client Adopters (check chapter 10.1 step 2 for details).

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12. CMLA IP

This chapter defines seven CMLA Key Transport Algorithms that all Devices and Rights Issuers MUST support, in addition to the mandatory key transport algorithms defined in [OMADRM-v2].

Each CMLA Key Transport Algorithm contains patent pending CMLA IP technology that is owned by CMLA and licensed according to the CMLA agreements. An entity that has not acquired a license for the CMLA IP cannot use CMLA Digital Content without infringing CMLA IP patents. As a result, CMLA IP prevents such entities from participating in the CMLA ecosystem.

Each CMLA Key Transport Algorithm is identified with an algorithm identifier. The identifiers are used in the ROAP PDUs defined [OMADRM-v2] sections 5.4.2.1 Device Hello, 5.4.2.2 RI Hello and as the value of the `Algorithm` attribute of the `<xenc:EncryptionMethod>` element in a similar way as the OMA mandatory Key Transport Algorithm defined in [OMADRM-v2] section 7.2 Key Transport Mechanisms.

The CMLA Key Transport Algorithms are parameterized by the choice of an Asymmetric Encryption Scheme, a Key Derivation Function, and a Key Wrapping Scheme. For each of these either the scheme defined in [OMADRM-v2] or the scheme defined in this specification is used, according to the table below. See also section 7.2 of [OMADRM-v2].

Algorithm Identifier	Asymmetric Encryption Scheme	Key Derivation Function	Key Wrapping Scheme
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-1	CMLA_RSA_Encrypt and CMLA_RSA_Decrypt	CMLA_KDF	CMLA_WRAP and CMLA_UNWRAP
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-2	CMLA_RSA_Encrypt and CMLA_RSA_Decrypt	CMLA_KDF	AES-Wrap
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-3	CMLA_RSA_Encrypt and CMLA_RSA_Decrypt	OMA_KDF	CMLA_WRAP and CMLA_UNWRAP
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-4	CMLA_RSA_Encrypt and CMLA_RSA_Decrypt	OMA_KDF	AES-Wrap
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-5	RSA.ENCRYPT and RSA.DECRYPT	CMLA_KDF	CMLA_WRAP and CMLA_UNWRAP
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-6	RSA.ENCRYPT and RSA.DECRYPT	CMLA_KDF	AES-Wrap
http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-7	RSA.ENCRYPT and RSA.DECRYPT	OMA_KDF	CMLA_WRAP and CMLA_UNWRAP

Both the Device and RI implementations MUST support all seven algorithms defined above.

To notify RIs that Device can use CMLA IP, Device MUST send the seven CMLA IP algorithm URIs in the `<supportedAlgorithm>` parameter in ROAP-DeviceHello message. If the Device indicated support of CMLA IP algorithms, RI MUST select the following Key Transport Algorithm identifier in the `<selectedAlgorithm>` parameter in ROAP-RIHello message.

"http://www.cm-la.com/tech/cmlaip/cmlaip#cmlaip-1"

The usage of other algorithms is not specified in the current specification.

Rights Issuer MUST use the Key Transport Algorithm identified above to transport key materials to the Device (MAC-key, REK, Domain Key). The used algorithm must be expressed correctly in the `Algorithm` attribute of the `<xenc:EncryptionMethod>` child element of the `<encKey>` element of ROAP-ROResponse and ROAP-JoinDomainResponse messages.

In case there is a dispute, the `Algorithm` attribute of the `<xenc:EncryptionMethod>` element takes precedence over the algorithm identifier stored in the Device's RI context.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12.1 Overview

CMLA IP consists of the following cryptographic primitives:

- Data-dependent Transformation with Permutation (CMLA_DDT_Perm),
- Data-dependent Transformation with Exponentiation (CMLA_DDT_Exp),
- CMLA Key Derivation Function (CMLA_KDF),
- CMLA Key Wrap Scheme (CMLA _WRAP and CMLA_UNWRAP) and
- CMLA RSA Encryption Scheme (CMLA_RSA_ENCRYPT and CMLA_RSA_DECRYPT).

CMLA Data-dependent Transformation with Permutation operates on a 64-bit string. Before transformation, the input is divided into 9 blocks. The first block consists of 8 bits. The remaining 8 blocks consist of 7 bits. The first block is used to select a transformation, which is then applied to the remaining part of the input. See Section 12.2.1 and 12.2.2 for details.

CMLA Data-dependent Transformation with Exponentiation operates on a 24-bit string. Before transformation, the input is divided into 2 blocks. The first block consists of 8 bits. The remaining block consists of 16 bits. The first block is used in index of exponentiation of an integer derived from the second block. The exponentiation is calculated in modulo the prime $p=2^{16}+1=65537$. See Section 12.2.3 and 12.2.4 for details.

CMLA Key Derivation Function operates on an octet string of length 128. The input splits in two blocks of equal length. A constant value is concatenated to the first block. Then SHA-1 is applied to the preceding result. The result and two integers derived from the second block are used in a modulus calculation to produce the Key Encrypting Key. The output shall be 16 bytes. See Section 12.3 for details.

CMLA Key Wrap Scheme wraps key data of length 32-byte using a 16-byte key encryption key. The key data splits in two blocks and CMLA_DDT_Perm is applied to each block. The AES Key Wrap Algorithm is applied to the preceding result. The output shall be 40 bytes. See Section 12.4 for details.

CMLA RSA Encryption Scheme produces a cipher text from a plaintext of length 128 bytes using a device public key. The input splits into two blocks and CMLA_DDT_Exp is applied to each block. The RSA Encryption without padding scheme is applied to the preceding result. The output shall be 128 bytes. See Section 12.5 for details.

For octet string (respectively, bit string) and integer conversion, big-endian format (known also as network byte order format) is used. See the section 5.2.5 (respectively, 5.2.4) in [ISO18033-2D].

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12.2 CMLA Data-Dependent Transformation

In this section, we will define two CMLA_DDTs and its inverse transformations; CMLA_DDT_Perm and its inverse CMLA_DDT_Perm_Inv, as well as CMLA_DDT_Exp and its inverse CMLA_DDT_Exp_Inv.

12.2.1 CMLA_DDT_Perm

CMLA_DDT_Perm is the CMLA Data-dependent Transformation with Permutation. It permutes last 56 bits of 64-bit input according to the first 8 bits.

CMLA_DDT_Perm (*b*)

Input:	***b***	**Bit string of length 64**
Output:	***B***	**Bit string of length 64**

Steps:

1. Let $b = (b_0, b_1, \ldots, b_{63})$.
2. For $0 \le i \le 7$, set $G_i = (b_{8+7i}, b_{9+7i}, \ldots, b_{14+7i})$, bit string of length 7.
3. The transformation works as follows for i=0, 1, … , 7.
   - If $b_i = 0$, let $H_i = G_i$.
   - If $b_i = 1$, let $H_i = P_i (G_i)$, where P_i is defined below.
4. Let $B = (b_0, b_1, \ldots, b_7) \| H_0 \| H_1 \| \ldots \| H_7$.
5. Output B.

The 8 permutations $P_0, P_1, \ldots, P_7$ are defined by:

Input	a	b	c	d	e	f	g
P_0	f	a	e	b	d	g	c
P_1	g	f	d	a	b	c	e
P_2	c	g	b	f	a	e	d
P_3	e	c	a	g	f	d	b
P_4	d	e	f	c	g	b	a
P_5	b	d	g	e	c	a	f
P_6	e	c	a	g	f	d	b
P_7	c	g	b	f	a	e	d

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12.2.2 CMLA_DDT_Perm_Inv

CMLA_DDT_Perm_Inv is the inverse transformation of CMLA_DDT_Perm.

CMLA_DDT_Perm_Inv (*B*)

Input:	***B***	**Bit string of length 64**
Output:	***b***	**Bit string of length 64**

Steps:

1. Let $B = (B_0, B_1, \ldots, B_{63})$.
2. For $0 \le i \le 7$, set $S_i = (B_{8+7i}, B_{9+7i}, \ldots, B_{14+7i})$, bit string of length 7.
3. The transformation works as follows for each i=0, 1, … , 7.
   - If $B_i = 0$, let $T_i = S_i$.
   - If $B_i = 1$, let $T_i = P_i^{-1}(S_i)$, where P_i^{-1} is defined below.
4. Let $b = (B_0, B_1, \ldots, B_7) \| T_0 \| T_1 \| \ldots \| T_7$.
5. Output b.

The 8 permutations $P_0^{-1}, P_1^{-1}, \ldots, P_7^{-1}$ are defined by:

Input	a	b	c	d	e	f	g
P_0^{-1}	b	d	g	e	c	a	f
P_1^{-1}	d	e	f	c	g	b	a
P_2^{-1}	e	c	a	g	f	d	b
P_3^{-1}	c	g	b	f	a	e	d
P_4^{-1}	g	f	d	a	b	c	e
P_5^{-1}	f	a	e	b	d	g	c
P_6^{-1}	c	g	b	f	a	e	d
P_7^{-1}	e	c	a	g	f	d	b

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12.2.3 CMLA_DDT_Exp

CMLA_DDT_Exp is the CMLA Data-dependent Transformation with Exponentiation. It permutes last 16 bits of 24-bit input according to the first 8 bits.

CMLA_DDT_Exp (*b*)

Input:	***b***	**Bit string of length 24**
Output:	***B***	**Bit string of length 24**

Steps:

1. Let $b = (b_0, b_1, \ldots, b_{23})$.
2. Set L= $(b_0, b_1, \ldots, b_7)$, bit string of length 8, and set R = $(b_8, b_9, \ldots, b_{23})$, bit string of length 16.
3. Regarding R and L as integer, compute $Y = ((R+1)^{(2L+1)} \bmod 65537) - 1$.
4. Let $B = L \| Y$.
5. Output B.

12.2.4 CMLA_DDT_Exp_Inv

CMLA_DDT_Exp_Inv is the inverse transformation of CMLA_DDT_Exp.

CMLA_DDT_Exp_Inv (*B*)

Input:	***B***	**Bit string of length 24**
Output:	***b***	**Bit string of length 24**

Steps:

1. Let $B = (B_0, B_1, \ldots, B_{23})$.
2. Set L = $(B_0, B_1, \ldots, B_7)$, an integer, and set R = $(B_8, B_9, \ldots, B_{23})$, an integer.
3. Compute e as the inverse of 2L+1 modulo 2^{16}: e * (2L+1) = 1 modulo 2^{16}.
4. Compute $X = ((R+1)^e \bmod 65537) - 1$.
5. Let $b = L \| X$.
6. Output b.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

12.3 CMLA Key Derivation Function

In this section, we will define CMLA Key Derivation Function CMLA_KDF. See the section 7.1.2 in [OMADRM-v2] for the details of OMA_KDF.

12.3.1 CMLA_KDF

CMLA_KDF is the CMLA Key Derivation Function.

CMLA_KDF (*x*)

Input:	***X***	**Seed value, an octet string of length 128**
Output:	***KEK***	**Key Encryption Key, an octet string of length 16**

Steps:

1. Let $x = x_0 \| x_1$, where each x_i consists of 64 bytes octet for i=0,1.
2. Let C=(0x)00 00 00 01 (4 bytes).
3. Compute Y=SHA-1(x_0|| C) (160 bits).
4. Let A be the first 32 bytes and B be the last 32 bytes of x_1, respectively .
5. Regarding Y, A, and B as integers, derive key encryption key *KEK* taking the least significant 128 bits of A*Y+B mod p, where $p=2^{192}-2^{64}-1$.
6. Output *KEK*.

12.4 CMLA Key Wrap Scheme

In this section, we will define CMLA Key Wrap and Unwrap Schemes. CMLA Key Wrap Algorithm can be obtained by composing AES Key Wrap algorithm (AES-WRAP) and CMLA_DDT_Perm. CMLA Key Unwrap Algorithm can be obtained by composing AES Key Unwrap Algorithm and CMLA_DDT_Perm_Inv. See the section 7.1.3 in [OMADRM-v2] for the details of AES-WRAP.

12.4.1 CMLA_WRAP

CMLA_WRAP is the CMLA Key Wrap Algorithm.

CMLA_WRAP (*KEK, K*)

Input:	*KEK*	Key Encryption Key, an octet string of length 16
	K	key to be wrapped, an octet string of length 32
Output:	*KWrap*	An octet string of length 40

Steps:

1. Let $K = K_0 \| K_1$, where each K_i consists of 16 bytes octet for i=0,1.
2. Apply CMLA_DDT_Perm to the first 8 bytes of K_i, keeping the rest of K_i unchanged, to produce 16-byte octet string k_i for i=0,1.
3. Let $k = k_0 \| k_1$.
4. Compute AES-WRAP (*KEK*, *k*) to produce an octet string *KWrap* of 40 bytes.
5. Output *KWrap*.

12.4.2 CMLA_UNWRAP

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

CMLA_UNWRAP is the CMLA Key Unwrap Algorithm.

CMLA_UNWRAP (*KEK, KWrap*)

Input:	***KEK***	**Key Encryption Key, an octet string of length 16**
	KWrap	**key to be unwrapped, an octet string of length 40**
Output:	***K***	**An octet string of length 32**

Steps:

1. Compute AES-UNWRAP (*KEK*, *KWrap*) to produce an octet string *k* of length 32 octets.
2. Let $k = k_0||k_1$, where each k_i consists of 16 bytes for i=0,1.
3. Apply CMLA_DDT_Perm_Inv to the first 8 bytes of k_i, keeping the rest of k_i unchanged, to produce 16-byte octet string K_i for i=0,1.
4. Let $K = K_0 || K_1$.
5. Output *K*.

12.5 CMLA RSA Encryption Scheme

In this section, we will define RSA Encryption and Decryption Schemes. CMLA RSA Encryption Algorithm can be obtained by composing RSA Encryption and CMLA_DDT_Exp. CMLA RSA Decryption Algorithm can be obtained by composing RSA Decryption and CMLA_DDT_Exp_Inv.

12.5.1 CMLA_RSA_ENCRYPT

CMLA_RSA_ENCRYPT is the CMLA RSA Encryption Algorithm.

CMLA_RSA_ENCRYPT (*pubKey, M*)

Input:	***pubKey***	**Device's RSA public key**
	M	**Message to be encrypted, an octet string of length 128**
Output:	***C***	**Cipher text, an octet string of length 128**

Steps:

1. Let $M = M_0 || M_1$, where each M_i consists of 64 bytes octet for i=0,1.
2. Apply CMLA_DDT_Exp to the first 3 bytes of M_i, keeping the rest of M_i unchanged, to produce octet string m_i for i=0,1.
3. Let $m = m_0 || m_1$.
4. If the integer representative of *m* is not strictly less than the RSA modulus, output "integer too long" and stop.
5. Encrypt *m* under device public key *pubKey* with RSA.ENCRYPT to produce the octet string *c* of length 128. That is,

   c = RSA.ENCRYPT (*pubKey*, *m*).
6. Output *c*.

12.5.2 CMLA_RSA_DECRYPT

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

CMLA_RSA_DECRYPT is the CMLA RSA Decryption Algorithm.

CMLA_RSA_DECRYPT (*pKey*, *c*)

Input:	*pKey*	Device's RSA private key
	c	Message to be decrypted, an octet string of length 128
Output:	*M*	Plain text, an octet string of length 128

Steps:

1. Decrypt *c* under device private key *pKey* with RSA.DECRYPT to produce the octet string *m* of length 128. That is,
2. m = RSA.DECRYPT (*pKey*, *c*).
3. Let $m = m_0 \| m_1$, where each m_i consists of 64 bytes octet for i=0,1.
4. Apply CMLA_DDT_Exp_Inv to the first 3 bytes of m_i, keeping the rest of m_i unchanged, to produce octet string M_i for i=0,1.
5. Let $M = M_0 \| M_1$.
6. Output *M*.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

13. CMLA Parameters

In addition to what has been specified in this document the following parameters will be communicated to adopters separately:

CMLA Root CA certificate(s). (for both Client Adopters and Service Providers)

Host names and username/password for fetching the Device CA certificate(s), the most recent Device CRL(s) and CMLA ARL (for Service Providers)

URL(s) and username/password for OCSP Responder service (for Service Providers)

Certificate Practice Statement

14. Development Keys and Certificates

In order to facilitate the R&D efforts of Client Adopters and Service Providers, CMLA will provide a separate development PKI and associated services that can be used for R&D purposes. The development PKI is completely separated from the production PKI and thus the liabilities for production keys, as set forth in the CMLA agreements, do not apply for the development keys.

In order to differentiate between development and production credential all file names used for development credentials MUST be prefixed with "**DEV_**", as listed below.

- "DEV_CMLA_Transport_Key_##.der"
- "DEV_CMLA_Root_CA_Certificate_##.der"
- "DEV_Client_Adopter_Transport_Key_*_##.der"
- "DEV_Rights_Issuer_CA_Certs.der"
- "DEV_Device_Credentials_*_####.p12"
- "DEV_Rights_Issuer_*_##.p10"
- "DEV_Rights_Issuer_*_##.der"

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

15. CMLA Mobile Broadcast (Signed Addendum Required)

USE AND IMPLEMENTATION OF THE FUNCTIONALITY IN THIS CHAPTER REQUIRES A SEPARATE ADDENDUM TO THE CMLA TECHNOLOGY LICENSE AGREEMENT. CMLA Technology Licensees wishing to implement the functionality specified in this chapter may only do so after signing the relevant Mobile Broadcast Addendum to their CMLA Technology License. ALL OTHER USE IS STRICTLY PROHIBITED AND NO LICENSE IS HEREBY GRANTED BY ESTOPPEL OR OTHERWISE EXCEPT AS SET FORTH IN THE MOBILE BROADCAST ADDENDUM.

15.1 Overview

The keys used in Broadcast Mode are delivered to the device in device_registration_response or domain_registration_response messages, in a data structure called keyset. By default, the keyset is protected with a session key (SK) during delivery, and the SK and the encrypted keyset are further protected by encrypting with the public key of the device (DP).

The steps that differ from the default keyset protection at the service provider when a CMLA algorithm is used are the following:

- additionally prepend a descriptor (ROT_algorithm_descriptor or TAA_descriptor, as specified in section 15.2 to the keyset to indicate that the keyset has been protected by a CMLA algorithm, and to identify the particular CMLA algorithm
- before using the default protection method, use the CMLA specific protection method specified in section 15.3

At the receiving end, after AES_UNWRAP decryption with SK, if the device detects a TAA_descriptor / ROT_algorithm_descriptor, it checks whether the keyset has been further protected with a CMLA algorithm, and if it is, the device decrypts it with the indicated algorithm.

15.2 ROT ALGORITHM DESCRIPTOR / TRUST AUTHORITY ALGORITHM DESCRIPTOR

15.2.1 Generic Syntax

The syntax of the keyset relies on so-called Tag-Length Format (TLF), which is defined to identify the keyset_items in the keyset. A keyset_item is identified by following syntax:

<tag> [optional <clarifier>] <length> <keyset_item>

The syntax of the ROT_algorithm_descriptor or TAA_descriptor TLF fields is specified in following Table.

Field	Length	Value	Type
ROT_algorithm_descriptor() / TAA_descriptor {			
Tag	4	1110_b	Bslbf
Clarifier	10		Uimsbf
Length	3	111_b	Bslbf
ROT_ID / TAA_TA_ID	10		Uimsbf
ROT_parameter	8*clarifier - 10 – n		Bslbf
Padding	n	0_b (n times)	Bslbf
}			

tag – This is the tag for ROT_algorithm_descriptor, previously “reserved for future use”.

clarifier – This is the clarifier for ROT_algorithm_descriptor. It shall represent the length of the keyset_item in bytes (excluding the tag, clarifier and length fields).

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

length – This has the nil (111) value, as the actual length is specified by the clarifier

ROT_ID / TAA_TA_ID – This field contains, in binary format, the Identifier of the Root of Trust that defined the algorithms for extra encryption of the keyset block in which this descriptor is located.

ROT_parameter – This field contains the Root of Trust specific parameter(s). Note that in IEC 62455 TAA_descriptor, this field has a predefined structure consisting of TAA_report_value (56 bits) and TAA_algorithm (8 bits), followed by TAA_parameter.

padding – This field is an n-bit zero-valued padding field.

15.2.2 CMLA-specific Values

Clarifier: 26 (this results in 8*26-10-6 = 192 bits as the size of ROT_parameter)

ROT_ID: the ROT_ID value assigned by DVB to CMLA, 0x0002.

ROT_parameter:

Field	Length	Value	Type
TAA_report_value	56	0	Bslbf
CMLA_algorithm_id	8	5, 6 or 7	Uimsbf
CMLA_key_material	128		Bslbf

TAA_report_value: This field is not used by CMLA, but it needs to be there for [IEC 62455]. Therefore, it is filled with zeros, regardless of whether the service is based on [18Crypt], [IEC 62455] or [OMABCAST-SCPv1] ([OMADRM-XBS] is required to implement) specifications.

CMLA_algorithm_id: This is a 8-bit field that identifies the CMLA algorithm used to protect that particular keyset, the values of which correspond to the number following “cmlaip-“ in the algorithm identifier that is specified for <xenc:EncryptionMethod> fields in the Interactive Mode. (Note that CMLA_RSA is not used.)

CMLA_algorithm_id	**Key Derivation Function**	**Key Wrapping / Unwrapping**
5	CMLA_KDF	CMLA_WRAP and CMLA_UNWRAP
6	CMLA_KDF	AES-Wrap
7	OMA_KDF	CMLA_WRAP and CMLA_UNWRAP

CMLA_key_material: randomly generated number

Padding: 000000 (6 zero-valued bits)

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

15.3 KEYSET Protection Details

The CMLA ROT_algorithm (Trust Authority Algorithm) that is used for an additional encryption layer (before the standardized encryption steps, and after the standardized decryption steps) depends on the value of CMLA_algorithm_id as follows:

Encryption:

If CMLA_algorithm_id is 5, KEK = CMLA_KDF(CMLA_key_material).

If CMLA_algorithm_id is 6, KEK = CMLA_KDF(CMLA_key_material).

If CMLA_algorithm_id is 7, KEK = OMA_KDF(CMLA_key_material).

The concatenated keyset SHALL be padded with one bit with the value '1' and, after this 1-valued bit, 0 to 63 bits with the value '0', such that the length of the padded keyset is a multiple of 64 bits, see Appendix A of [NIST 800-38A]. Note that if the non-padded keyset was already a multiple of 64 bits in length, it is padded with 64 bits.

If CMLA_algorithm_id is 5, use CMLA_WRAP to encrypt the keyset with KEK, starting at the last bit of ROT_ algorithm_descriptor (or TAA_descriptor).

If CMLA_algorithm_id is 6, use AES_WRAP to encrypt the keyset with KEK starting at the last bit of ROT_algorithm_descriptor (or TAA_descriptor).

If CMLA_algorithm_id is 7, use CMLA_WRAP to encrypt the keyset with KEK starting at the last bit of ROT_ algorithm_descriptor (or TAA_descriptor).

Decryption:

If CMLA_algorithm_id is 5, KEK = CMLA_KDF(CMLA_key_material).

If CMLA_algorithm_id is 6, KEK = CMLA_KDF(CMLA_key_material).

If CMLA_algorithm_id is 7, KEK = OMA_KDF(CMLA_key_material).

If CMLA_algorithm_id is 5, use CMLA_UNWRAP to decrypt the keyset with KEK, starting at the last bit of ROT_ algorithm_descriptor (or TAA_descriptor).

If CMLA_algorithm_id is 6, use AES_UNWRAP to decrypt the keyset with KEK starting at the last bit of ROT_algorithm_descriptor (or TAA_descriptor).

If CMLA_algorithm_id is 7, use CMLA_UNWRAP to decrypt the keyset with KEK starting at the last bit of ROT_ algorithm_descriptor (or TAA_descriptor).

Remove padding.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

16. Content Protection Technology System Renewability Message (CPT-SRM) Support

This chapter describes CMLA provisions for the carriage of CPT-SRMs in the OMA DRM Content Format. Currently CMLA supports HDCP and DTCP content protection technologies. CPT-SRM support is Optional and is provided to support various business requirements.

The Mutable DRM information Box section 5.2.4 of [OMA-DCF] provides for the modification, extension, truncation, deletion or addition of the MutuableDRMInformation box. So for delivery of DTCP and HDCP system renewability messages into either a DCF or a PDCF the following new CPT-SRM boxes are defined:

- DTCP CPT-SRM Delivery Box, carries the DTCP SRM as defined in Chapter 7 of [DTCPV1]

```
Aligned(8) clase DTCPSrm extends FullBox ('dsrm', 0, 0) { byte Data[ ]; // DTCP SRM  }
```

- HDCP CPT-SRM Delivery Box carries the HDCP SRM as defined in section 5 of [HDCP-V1.3]

```
Aligned(8) clase HDCPSrm extends FullBox ('hsrm', 0, 0) { byte Data[ ]; // HDCP SRM }
```

Where:

- The SRM data is to be treated as binary data.
- Devices MUST NOT add or delete CPT-SRM delivery boxes to the MutuableDRMInformation box.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17. CMLA DECE Implementation (Signed Addendum Required)

USE AND IMPLEMENTATION OF THE FUNCTIONALITY IN THIS CHAPTER REQUIRES A SEPARATE ADDENDUM TO THE CMLA TECHNOLOGY LICENSE AGREEMENT. CMLA Technology Licensees wishing to implement the functionality specified in this chapter may only do so after signing the relevant DECE Addendum to their CMLA Technology License. ALL OTHER USE IS STRICTLY PROHIBITED AND NO LICENSE IS HEREBY GRANTED BY ESTOPPEL OR OTHERWISE EXCEPT AS SET FORTH IN THE DECE ADDENDUM.

This Chapter 17 describes the additional technical requirements for CMLA implementations that support Digital Entertainment Content Ecosystem (DECE) down load service. The requirements for DECE streaming services are to be defined later.

17.1 OVERVIEW

This Chapter is primarily concerned with Device, CMLA-DM and CMLA-DSP requirements.

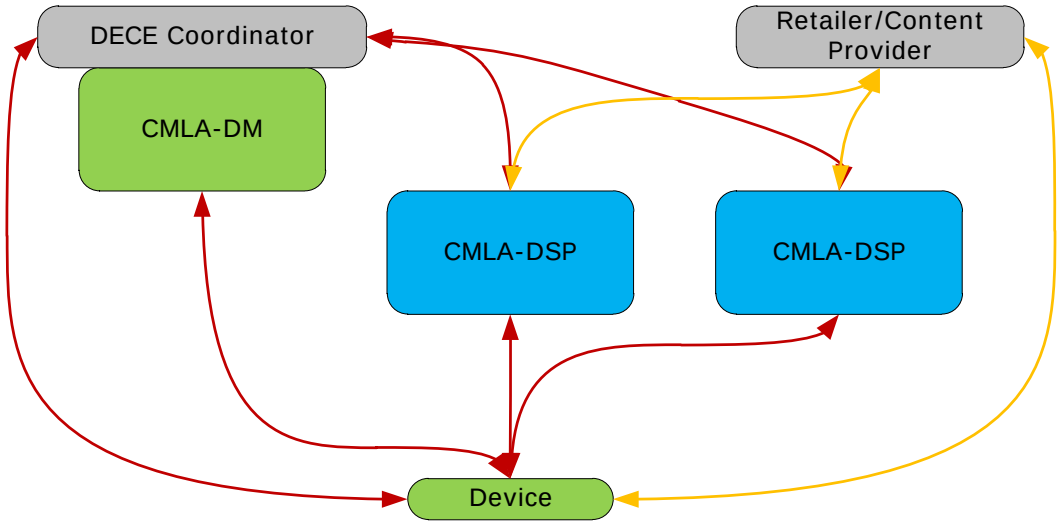


Figure 3 DECE Domain System Elements

The DECE Coordinator is the domain control authority for DECE Ecosystem. Each DECE account corresponds to a specific DECE domain which is referred to hereafter as CMLA DECE Domain for implementations meeting this chapter's requirements. The DECE Coordinator tracks the content and associated rights to that content as well as manages the number of devices for each CMLA DECE account (domain).

The CMLA Domain Manager CMLA-DM is used by the DECE Coordinator to add and remove Devices to each account using the OMA DRM Domain Join and Leave ROAP protocols. Note, that the OMA DRM Identification Protocol is used during the DECE join process to communicate Attestation data. CMLA-DM is prohibited from providing any type OMA DRM Rights Objects.

The only purpose of the CMLA Download Service Provider (CMLA-DSP) is to create CMLA DECE Domain Rights Objects in conjunction with Retailers. The CMLA-DSP is prohibited from processing OMA ROAP Domain Join and Leave protocols.

CMLA DECE Domain Context is shared among RIs (i.e. CMLA-DM, CMLA-DSP). In this case CMLA DECE Domain context is obtained from the CMLA-DM and used by CMLA-DSPs to generate Domain ROs.

To enable identification of CMLA-DM and CMLA-DSP by Devices, additional key purposes are added to RI certificates of CMLA-DM and CMLA-DSP so that a Device can identify the DECE RIs that share CMLA DECE Domain Context. The types of key purposes used are as follows:

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

- A key purpose that identifies the RI is part of a set of RIs that share a RI Common Domain (RICD).
- A key purpose that identifies the RI as a CMLA-DM
- A key purpose that identifies the RI as a CMLA-DSP

17.2 OMA Compliance Requirements

This CMLA OMA DECE implementation is based on OMA DRM Release 2.1 [OMADRM-V2.1] except for the following features which are not required but may be implemented if desired by an adopter:

- Metering is not required to be implemented (includes all associated protocols and commands).
- Rights Object installation confirmation is not required.
- Devices are not required to support WBXML decoding of ROAP Triggers.

The PKI system is unchanged and is only augmented via small changes to RI certificates of the RI entities CMLA Domain Manager (CMLA-DM) and CMLA Download Service Provider (CMLA-DSP) of the DECE system.

This implementation is based on OMA domains except that domain context is shared across multiple RIs where domain control functions are provide though the CMLA-DM and CMLA-DECE domain ROs are issued by CMLA-DSPs.

Devices that support this implementation defined in this section 17 SHALL support domain functionality.

The word delete when used in conjunction with domain operations in the [OMADRM-V2.1] shall mean obliterate or render unusable.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.3 Device

In the DECE ecosystem multiple RIs are allowed to issue Domain ROs for the same domain. The CMLA-DM manages the domain and CMLA-DSPs issue the Domain ROs.

In addition, some elements of the RI context that are required to process a Domain RO may be provided to the device inside the Protection System Specific Header ('pssh') Box of the [DECECFF].

To accommodate these deviations from the OMA DRM v2.1 specifications the following modifications are required for the Domain RO installation process defined in the Section 8.7.2.1 of [OMADRM v2.1]:

1. When the RI context does not exist, then the Device MUST attempt to contact the RI via the riURL as defined in [OMADRMV2.1]. If the Device cannot contact the RI, then the Device is permitted to use the RI context information in the Protection System Specific Header ('pssh') Box of the [DECECFF]. Note, such RI context information MUST NOT be used for other purposes than Domain RO installation and is not considered to be persistently established. The Device MUST continue to attempt to establish RI content using ROAP once a connection exists.
2. If the RI that issued the Domain RO is a CMLA-DSP then comparison of the <domainID> only involves the stored Domain Contexts established by the CMLA-DM.

Device certificates are unchanged for DECE implementations.

Devices must be capable of sending Device-specific information to the CMLA-DM or CMLA-DSP using the Device Details extensions during the 4-pass Registration protocol of [OMADRM-V2.1]. The device details should be structured such that the version field of the Device Details indicates at a minimum the DRM agent version.

For clarification of DECE terms and requirements:

1. The term DRM License used in DECE specifications refers to OMA Domain ROs.
2. The DECE ecosystem also imposes a restriction to the number of CMLA DECE Domains per Device, Specifically, a Device MUST NOT be a member of more than one CMLA DECE Domain at any time.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.3.1 DECE CMLA OMA Domain Manager Interaction

Devices must use URl of the CMLA-DM when sending unsolicited Leave Domain requests. The DM does not generate any ROs and only provides OMA Domain functions Join and Leave.

17.3.1.1 Device Attestation

One of the DECE requirements is Device Attestation, in which the DECE Coordinator verifies that a Device is allowed access to DECE services. The ROAP 2-pass Identification Protocol is used for that purpose. This section describes the Device attestation information that is carried in the ROAP Identification Protocol during a Join procedure.

17.3.1.2 ROAP Identification Protocol

The OMA DRM v2.1 Identification Protocol is designed to be used to identify the Device. To fulfill the Device attestation requirements a new digitally signed extension is introduced to the Identification Request message. The extension will only be sent to specific DECE Rights Issuers, namely the CMLA-DM. The required modification is encapsulated into the creation and validation of the Identification Request message as defined below – all other parts of the ROAP protocol suite, Device and Rights Issuer behavior are left unmodified.

The Extensions element SHALL contain the extension carrying DECE information which consists of the Device ID along with the attestation data. The Extension type SHALL be set to ***roap:ExtensionContainer***.

The following is the XML schema definition of DECE Information attestation data.

```
<complexType name="DeceInformation">
    <sequence>
     <element name="version" type="roap:Version"/>
     <element name="deceInfo">
      <complexType>
       <sequence>
        <element name="deviceID" type="roap:Identifier" minOccurs="0"/>
        <element name="manufacturerId", type="roap:String64" minOccurs="0"/>
       <!-- DECE has yet to define manufacturerdId structure or values
            so it may be 128 chars in length as shown in this xml comment.
       <xs:element name="manufacturerId">
        <xs:simpleType>
         <xs:restriction base="xs:string"> <xs:maxLength value="128"/> </xs:restriction>
        </xs:simpleType>
       </xs:element>
       -->
        <element name="modelId", type="roap:String64" minOccurs="0"/>
        <element name="licensedAppId", type="roap:String64" minOccurs="0"/>
        <any namespace="##any" processContents="lax" minoccurs="0" maxOccurs="unbounded"/>
        <element name="signature" type="base64Binary"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
</complexType>
```

The ***roap:ExtensionContainer*** carrying ***DeceInformation*** extension SHALL be included in the ROAP Identification Request if and only if:

- A valid RI context already exists AND
- The Rights Issuer certificate in the RI context contains the extKeyUsage extension with the key purpose identifier value *cmla-kp-dece-dm*.

The ***critical*** attribute SHALL be set to ”false” for the extension i.e. implementations not understanding the extension SHALL ignore it.

The ***version*** element SHALL define the version of this extension and SHALL be set to “1.0”.

The ***deviceID*** element SHALL contain the Device identifier as defined in [OMADRM-V2.1].

The ***manufacturerId*** element SHALL contain the DECE assigned organization ID for the manufacturer.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

The ***modelId*** element SHALL contain the identifier of model of the DRM Client as defined in [DECESYS].

The ***licencedAppId*** element SHALL contain the identifier of the Licensed Application as defined in the [DECESYS].

The extension contents may be appended in the future with any number of any XML elements. Also, the elements mentioned above are optional which means the extension syntax is defined to support future extensions and modifications.

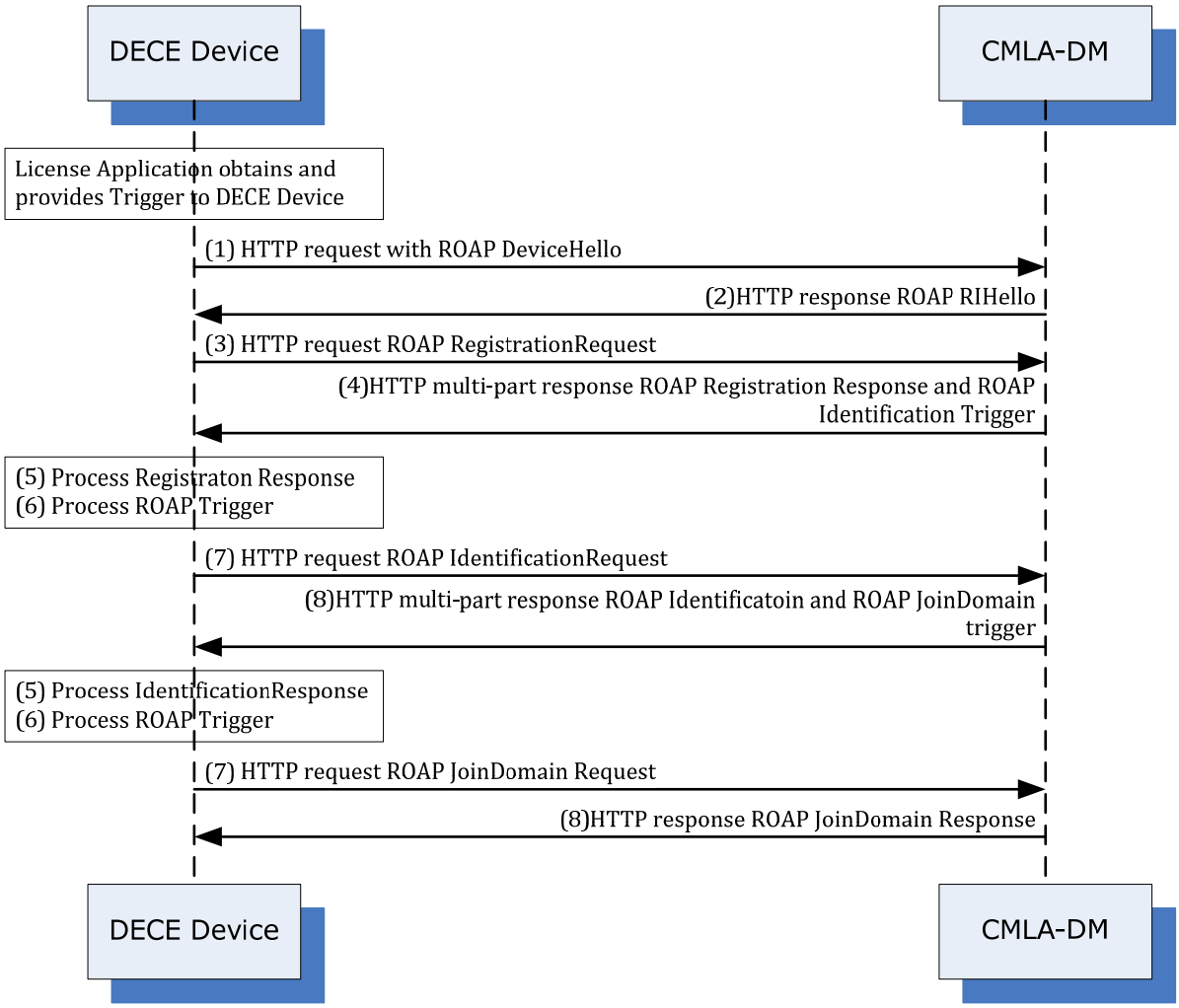
The signature element SHALL contain a digital signature calculated over the extension, excluding the signature element itself.

- The extension except the signature element SHALL be canonicalization according to Section 5.3.3. of [OMADRM-V2.1].
- The result of the canonicalization SHALL be used as the input to the signature operation.
- The signature SHALL be calculated using the previously negotiated signature scheme and private key.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.3.1.3 Device Join Procedure

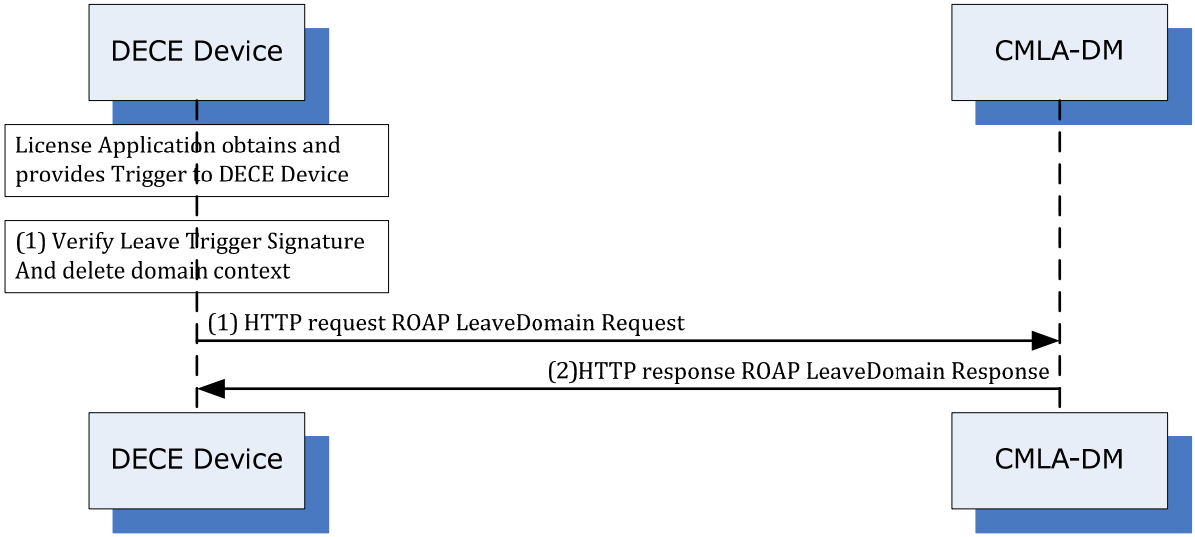
The sequence diagram below describes a successful Device standalone domain join process. Note, that the Device is required to retain the riURL of the RI of the CMLA-DM that is used in the Domain Join. The riURL is used when transmitting attestation information and to support user initiated leave protocol. Note, the Licensed Application provides the trigger to Device.



© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.3.1.4 Domain Leave Procedure

Domain leave process is architecturally and technically almost identical to the domain join process. The sequence diagram below describes the domain leave process.



The domain leave operation may also be initiated by the user. In those unsolicited domain leave scenarios the ROAP domain leave trigger is not used at all. Instead, the DRM Client posts the ROAP leave domain request to the riURL that is part of the RI context.

17.3.2 Device DSP Interaction

The Device uses normal ROAP protocols when communicating with the DSP to obtain DECE Domain ROs. The DSP only provides ROs and will not support any domain management functions (Domain Join and Domain Leave).

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.4 CMLA Download Service Provider

CMLA-DSP must only generate Domain ROs as described in section 17.6 of this specification.

CMLA-DSPs must be capable of requesting the Device-specific information via the Device Details extension of the ROAP-RIHello message. Note, DECE intends to use the Device Details to determine if the associated Device is permitted to receive Domain ROs.

For clarification from DECE specification:

1. CMLA-DSPs may insert a Domain RO into the CMLA-PSSH box of a [DECECFF]. The contents of the CMLA-PSSH box are described in section 17.7.
2. DECE content must be delivered in a [DECECFF].

CMLA-DSPs are not permitted to enable and/or execute domain management ROAP protocols Join and Leave.

To ensure uniqueness of RO identifiers, CMLA shall assign part of <roID> structure that is unique to each CMLA-DSP.

Note, concerning Section 12 use of CMLA Key Transport Algorithms, while CMLA-DSP MUST support CMLA Key Transport Algorithms for the initial implementation, no CMLA Key Transport Algorithms will be used by CMLA-DSPs.

CMLA-DSPs when obtaining CMLA Domain information shall use the Coordinator DomainInformationRequest() function in 9.1.7 of [DECECOR] where for CMLA:

- The request body contains:
  - The certificate of the CMLA-DSP that has requested the domain credentials.
  - The certificate is sent in a <ds.X509Certificate> element within the <ds:X509Data> element as defined in [XMLDSIG]
- The response body contains:
  - The response contains the domain credentials, which includes a randomly generated 128 bit AES Domain Key (K_D) protected by the CMLA-DSP's public key. It also contains the Domain ID.

In section 12.3, License Acquisition, of [DECESYST] there are "DRM specific parameters" that are part of the License Acquisition URL (LAURL), which contain the Domain ID. The "DRM specific parameters" of the LAURL are defined as follows.

?domainId="roap:DomainIdentifier"

Example: cmlauv-license.xyzstore.uvvu.com/urn:dece:apid:org:mycompany:abcdefg:100?domainID= AKJaxVws6uRw000

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.4.1 Rights Issuer Certificate for Download Service Providers

The Rights Issuer certificate for Download Service Providers is identical to a regular Rights Issuer certificate with the exception of two additional key purpose identifiers are included in the ExtKeyUsage extension. This certificate is issued by the Rights Issuer CA and is sent, upon request, from the CMLA-DSP to the Device as a part of the CMLA-DSP's certificate chain.

All Rights Issuer Certificates SHALL be based on OMA DRM specification appendix E.2 "Rights Issuer Certificates" [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the issuing Rights Issuer CA Certificate's *subject* field.
Validity	Expires 5 years from the issuance date but not later than the certificate of the corresponding Rights Issuer CA Certificate.
Subject	Contains always attributes: - *organizationName* (e.g. "CoolService Ltd.") as assigned by CMLA for the specific Service Provider May contain additional attributes as defined in [OMADRM-v2]. The values of those fields are assigned by the Service Provider.
SubjectPublicKeyInfo	Carries a 1024 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Not used
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bit *digitalSignature* is set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding Rights Issuer CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Not used
CertificatePolicies extension	Not used
ExtKeyUsage extension	Mandatory, critical. The extension contains the follow object identifiers: *oma-kp-rightsIssuer, cmla-kp-dece-ricd, cmla-kp-dece-dsp*
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.5 CMLA DRM Domain Manager

The main and only purpose of the CMLA-DM is to manage CMLA DECE Domains. The CMLA-DM is not permitted to generate any type of OMA DRM Rights Objects.

CMLA-DMs must be capable of requesting the Device-specific information via the Device Details extension of the ROAP-RIHello message.

Note, CMLA-DM will use CMLA Key Transport Algorithms as called out in Section 12 of this specification.

17.5.1 Rights Issuer Certificate for Domain Managers

The Rights Issuer certificate for the Domain Manager is identical to a regular Rights Issuer certificate with the exception of two additional key purpose identifiers that are included in the ExtKeyUsage extension. This certificate is issued by the Rights Issuer CA and is sent, upon request, from the CMLA-DM to the Device as a part of the CMLA-DM's certificate chain.

All Rights Issuer Certificates SHALL be based on OMA DRM specification appendix E.2 "Rights Issuer Certificates" " [OMADRM-v2].

Certificate field name	Value
Version	*v3* (integer value 2)
Signature	*sha-1WithRSAEncryption* as defined in [RFC3279]
Issuer	Same as the issuing Rights Issuer CA Certificate's *subject* field.
Validity	Expires 5 years from the issuance date but not later than the certificate of the corresponding Rights Issuer CA Certificate.
Subject	Contains always attributes: - *organizationName* (e.g. "CoolService Ltd.") as assigned by CMLA for the specific Service Provider May contain additional attributes as defined in [OMADRM-v2]. The values of those fields are assigned by the Service Provider.
SubjectPublicKeyInfo	Carries a 1024 bit RSA public key identified with the *pkcs-1* algorithm identifier as defined in [RFC3279].
IssuerUniqueID	Not used
SubjectUniqueID	Not used
BasicConstraints extension	Not used
CRLDistributionPoints extension	Not used
KeyUsage extension	Mandatory, critical. Only key usage bit *digitalSignature* is set.
AuthorityKeyIdentifier extension	Mandatory, non-critical. The extension contains only the *keyIdentifier* field that has the same value as the corresponding Rights Issuer CA Certificate's *subjectKeyIdentifier* extension value.
SubjectKeyIdentifier extension	Not used
CertificatePolicies extension	Not used
ExtKeyUsage extension	Mandatory, critical. The extension contains only the following object identifiers: *oma-kp-rightsIssuer*, cmla-kp-dece-ricd, cmla-kp-dece-dm
Id-pkix-ocsp-nocheck extension	Not used

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.6 DECE Domain ROs

The section details the structure of the two types of Domain ROs that CMLA-DSPs will issue. The Domain RO containing two content keys, one for Audio and one for Video, may be used for HD content as specified in "Constraints on Encryption" of the HD Media Profile Definition of [DECECFF]. Otherwise, the Domain RO containing a single content key is used. The following sections contain examples of each type of Domain RO.

17.6.1 Domain RO with Key for both Audio and Video Tracks

```
<roap:protectedRO
 xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
 xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <!-- ro ID needs to be globally unique within CMLA DECE implementation
  RI assigned unique identifier by CMLA-->
 <roap:ro     id="n8yu98hy0e2109eu09ewf09u"
domainRO="true"
version="1.1"
riURL="http://www.CMLAOMADSP1.com">
   <!—RI ID if from the RI certificate, it's the hash of the complete DER-encoded subjectPublicKeyInfo
component in the RI certificate -->
   <riID>
    <keyIdentifier xsi:type="roap:X509SPKIHash">
     <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
    </keyIdentifier>
   </riID>
   <rights o-ex:id="REL1">
    <o-ex:context>
     <o-dd:version>2.1</o-dd:version>
     <o-dd:uid>n8yu98hy0e2109eu09ewf09u</o-dd:uid><!-- same as roID -->
    </o-ex:context>
    <o-ex:agreement>
     <o-ex:asset o-ex:id ="Audio-Video Key KIDValue"><!-- note KID value is appended -->
      <o-ex:context>
       <o-dd:uid>ContentID</o-dd:uid> <!—contains APID of CFF AssetInformationBox -->
      </o-ex:context>
      <ds:KeyInfo>
       <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
         <ds:KeyInfo>
           <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
         </ds:KeyInfo>
        <xenc:CipherData>
         <xenc:CipherValue>EncryptedCEK</xenc:CipherValue>
        </xenc:CipherData>
       </xenc:EncryptedKey>
      </ds:KeyInfo>
     </o-ex:asset>
     <o-ex:permission>
      <o-dd:play/>
<!-- outputs -->
      <o-dd:export oma-dd:mode="copy" oma-dd:transcribe="false">
        <o-ex:constraint>
         <oma-dd:system>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-analog-outputs</o-dd:uid>
          </o-ex:context>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-dtcp</o-dd:uid>
          </o-ex:context>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-hdcp</o-dd:uid>
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
          </o-ex:context>
         </oma-dd:system>
         </o-ex:constraint>
       </oma-dd:export>
      </o-ex:permission>
     </o-ex:agreement>
    </rights>
    <signature>
     <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod
       Algorithm="http://www. rsasecurity.com/rsalabs/pkcs/schemas/pkcs-1#rsa-pss-default"/>
      <ds:Reference URI="#REL1">
       <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <ds:DigestValue>sIo5hb+id8JtuOMNKs12=drf5+3df=</ds:DigestValue>
      </ds:Reference>
     </ds:SignedInfo>
     <ds:SignatureValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:SignatureValue>
     <ds:KeyInfo>
      <roap:X509SPKIHash>
       <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
      </roap:X509SPKIHash>
     </ds:KeyInfo>
    </signature>
    <encKey Id="K_MAC_and_K_REK">
     <xenc:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
     <ds:KeyInfo>
      <roap:domainID>Domain-XYZ-001</roap:domainID>
     </ds:KeyInfo>
     <xenc:CipherData>
      <xenc:CipherValue>32fdsorew9ufdsoi09ufdskrew9urew0uderty5346wq</xenc:CipherValue>
     </xenc:CipherData>
    </encKey>
   </roap:ro>
   <mac>
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     <ds:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
     <ds:Reference URI="#n8yu98hy0e2109eu09ewf09u">
      <ds:Transforms>
       <ds:Transform Algorithm=http://www.w3.org/2001/10/xml-exc-c14n#/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>sIo5hb+id8JtuOMNKs12=drf5+3df=</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:SignatureValue>
    <ds:KeyInfo>
     <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
    </ds:KeyInfo>
   </mac>
</roap:protectedRO>
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

17.6.2 Domain RO with separate keys, one for Audio and one for Video Track

```
<roap:protectedRO
 xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
 xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <!-- ro ID needs to be globally unique within CMLA DECE implementation  -->
 <roap:ro      id="n8yu98hy0e2109eu09ewf09u"
domainRO="true"
version="1.1"
 riURL="http://www.DSP1.com">
  <!—RI ID if from the RI certificate, it's the hash of the complete DER-encoded subjectPublicKeyInfo
component in the RI certificate -->
  <riID>
    <keyIdentifier xsi:type="roap:X509SPKIHash">
     <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
    </keyIdentifier>
   </riID>
   <rights o-ex:id="REL1">
    <o-ex:context>
     <o-dd:version>2.1</o-dd:version>
     <o-dd:uid>n8yu98hy0e2109eu09ewf09u</o-dd:uid><!-- same as roID -->
    </o-ex:context>
    <o-ex:agreement>
     <o-ex:asset o-ex:id ="Video Key KIDValue"><!-- note KID value is appended -->
      <o-ex:context>
       <o-dd:uid>ContentID</o-dd:uid><!—contains APID of [DECECFF] AssetInformationBox -->
      </o-ex:context>
      <ds:KeyInfo>
       <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
         <ds:KeyInfo>
           <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
        </ds:KeyInfo>
        <xenc:CipherData>
         <xenc:CipherValue>EncryptedCEK</xenc:CipherValue>
        </xenc:CipherData>
       </xenc:EncryptedKey>
      </ds:KeyInfo>
     </o-ex:asset>
    <o-ex:asset o-ex:id ="Audio Key KIDValue"> <!-- note KID value is appended -->
      <o-ex:context>
       <o-dd:uid>ContentID</o-dd:uid><!—contains APID of [DECECFF] AssetInformationBox -->
      </o-ex:context>
      <o-ex:digest>
       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <ds:DigestValue>bLLLc+Um/5/NvmYKiHDLaErK0fk=</ds:DigestValue>
       <!-- base64 encoded hash value of the [DECECFF] -->
      </o-ex:digest>
      <ds:KeyInfo>
       <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
         <ds:KeyInfo>
           <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
        </ds:KeyInfo>
        <xenc:CipherData>
         <xenc:CipherValue>EncryptedCEK</xenc:CipherValue>
        </xenc:CipherData>
       </xenc:EncryptedKey>
      </ds:KeyInfo>
     </o-ex:asset>
     <o-ex:permission>
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
      <o-dd:play/>
<!-- outputs -->
      <o-dd:export oma-dd:mode="copy" oma-dd:transcribe="false">
        <o-ex:constraint>
         <oma-dd:system>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-analog-outputs</o-dd:uid>
          </o-ex:context>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-dtcp</o-dd:uid>
          </o-ex:context>
          <o-ex:context>
           <o-dd:uid>urn:oma:drms:org-cmla:uv-hdcp</o-dd:uid>
          </o-ex:context>
         </oma-dd:system>
         </o-ex:constraint>
      </oma-dd:export>
     </o-ex:permission>
    </o-ex:agreement>
   </rights>
   <signature>
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
           <ds:SignatureMethod
      Algorithm="http://www. rsasecurity.com/rsalabs/pkcs/schemas/pkcs-1#rsa-pss-default"/>
     <ds:Reference URI="#REL1">
      <ds:Transforms>
       <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>sIo5hb+id8JtuOMNKs12=drf5+3df=</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:SignatureValue>
    <ds:KeyInfo>
     <roap:X509SPKIHash>
      <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
     </roap:X509SPKIHash>
    </ds:KeyInfo>
   </signature>
   <encKey Id="K_MAC_and_K_REK">
    <xenc:EncryptionMethod
     Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
    <ds:KeyInfo>
     <roap:domainID>Domain-XYZ-001</roap:domainID>
    </ds:KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>32fdsorew9ufdsoi09ufdskrew9urew0uderty5346wq</xenc:CipherValue>
    </xenc:CipherData>
   </encKey>
  </roap:ro>

<mac>
   <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod
     Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#n8yu98hy0e2109eu09ewf09u">
     <ds:Transforms>
      <ds:Transform Algorithm=http://www.w3.org/2001/10/xml-exc-c14n#/>
     </ds:Transforms>
     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>sIo5hb+id8JtuOMNKs12=drf5+3df=</ds:DigestValue>
    </ds:Reference>
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
    </ds:SignedInfo>
    <ds:SignatureValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
    </ds:KeyInfo>
  </mac>
</roap:protectedRO>
```

17.7 CMLA-PSSH Elements

17.7.1 CMLA-PSSH Box

RI context information and Domain RO may be stored in the CMLA Protection System Specific Header (CMLA-PSSH) box of the [DECECFF] file. CMLA-DSPs and Devices may insert this box.

17.7.2 CMLA-PSSH UUID

The System ID field in the CMLA-PSSH box is a UUID that uniquely identifies the content protection system. The value of this UUID for CMLA-OMA DECE information is:

A2B55680-6F43-11E0-9A3F-0002A5D5C51B

Devices must verify the SystemID field in the CMLA-PSSH before using any information in this box.

17.7.3 CMLA-PSSH Contents

CMLA-PSSH may contain the “cmlapssBox” element which contains a Domain RO and associated RI context information (RI Certificate Chain) which can be used by a Device to consume the content of the CFF.

The CMLA-PSSH contents are shown below. These contents SHALL be canonicalization according to Section 5.3.3. of [OMADRM-V2.1].:

```
<complexType name="cmlaPsshBox”>
<sequence>
<element name="certificateChain" type="roap:CertificateChain"minOccurs="1" maxOccurs="1"/>
<element name=”protectedRo” type=”roap:ProtectedRO” minOccurs=”1”maxOccurs="1"/>
</sequence>
<attribute name=”version” type=”roap:Version” use=”required”/>
</complexType>

<element name=”cmlapsshBox” type=”cmlauv:cmlaPsshBox”/>
```

Version is a <major.minor> representation of the highest CMLAUV version number supported by the Device. Devices MUST support all version prior to the one they suggest. Note, version must be set to 1.0 for this version of the specification as shown in following example.

```
<cmlapsshBox>
version="1.0"
<!—Certificate Chain -->
<certificateChain>
<certificate>MIIC3DCCAcSgAwIBAgIEDpGzQTANBgkqhkiG9w0BAQUFADBGMQswCQYDVQQGDAJVUz
ENMAsGA1UECgwEQ01MQTEoMCYGA1UEAwwfUmlndGhzIElzc3VlciBDQSBmb3IgdGVzdHMgb25seTAe
Fw0wNjAxMjYwMDU4MzVaFw0xMTAxMjYwMDU4MzVaMFQxCzAJBgNVBAYMAlVTMRMwEQYDVQQIDAp
Tb21lLVN0YXRlMRcwFQYDVQQKDA5DTUxBLVRFU1QtMDEwNjEXMBUGA1UEAwwOQ01MQS1URVNULT
AxMDYwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAPdQzgKmh1v0S4hB4gKwpYg/W7YhUCKDHp
xqeV4wx0OdLU/VOYu1dlaRXhRbM+8vhJTiBlLUZfTRZlcocgccpas0TLf/mDGdzs6yQOIEOXlyqQK8Apciq9
N889ZeqdrZMKAXv+KzMopVzke9uDwVNwy5r+KkV5dwOfR2KjbO6JXFAgMBAAGjSDBGMA4GA1UdDw
EB/wQEAwIHgDAfBgNVHSMEGDAWgBSCGbn+1fVmJNhEVRaad20chmLAvzATBgNVHSUBAf8ECTAHBg
VnKwYBATANBgkqhkiG9w0BAQUFAAOCAQEAGQrscEcakVgs88VVK1TCcCnYl5SS+ij24/LJ6Ebw7hDD1b
DjJPoaNdYjohxNVXiVGXV5BNrnAfGCNtiu8+7S1TFKdKetkIK87oCCvsC74zUfBGLhrQj0g1AFBHOGON3Jp
IcN8baqxcxWExaB1ND1WvFa6qPCK/jz88K3Cfj6t1At27Y+ocJ+Wqfi8dpUzSwDUl4eI2jEYRGOHW6C9um
1PJIXGa8Nb9Pc8ZO8qo7a8eU559FV91CXww2UZ1jXAdCjLg+gCVqZAYcCB/FAb8YP/BdegVbBUigtb2ZV
ZexLVJ2AGf4jN9kWUqUUsS69AlS65BMJ720rs2UuuEjuOyeVbg==
</certificate>
<certificate>
MIIDZzCCAk+gAwIBAgIEVuRoqzANBgkqhkiG9w0BAQUFADA9MQswCQYDVQQGDAJVUzENMAsGA1UEC
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

gwEQ01MQTEfMB0GA1UEAwwWUm9vdCBDQSBmb3IgdGVzdHMgb25seTAeFw0wNTAxMTcyMjQwNTBa
Fw0zNDEwMTcyMjQwNTBaMEYxCzAJBgNVBAYMAlVTMQ0wCwYDVQQKDARDTUxBMSgwJgYDVQQDDB
9SaWd0aHMgSXNzdWVyIENBIGZvciB0ZXN0cyBvbmx5MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBC
gKCAQEAlBfmnyf17jTbxU0bYRyNVBFp/Rrc3mquk2s2QgNzIzCtK0PLJDqDa9ZPRgdyxxL/69gPQFl9FlzSh
xpYsV4lDba+28v1zNVBHp32h74eexInII5OH704ZycJkhMJvkDF6cs00ua6DlRuehvYv7i/0bW90zOaSeaD
EbYjmiIHhwhdIsuNMfCMOHnyHr/gnWmULOIwy9ZYns3qud6PMKs+JQ596MnBnJyzpKqaXk1QM4ezuvfM
T2wqp+8vVoIE07Qn4AizU3p3wMvPE2RyLfUZbgZqRhXoM+k+RQanBbWo0uwD38+yQgFsB27gTPcMiojg
3AZsI6J03PnwCbr/wBu4aQIDAQABo2YwZDAdBgNVHQ4EFgQUghm5/tX1ZiTYRFUWmndtHIZiwL8wDgYD
VR0PAQH/BAQDAgEGMBIGA1UdEwEB/wQIMAYBAf8CAQAwHwYDVR0jBBgwFoAUjEgFYZpvrU+5DFn4/T
1RCOEYO30wDQYJKoZIhvcNAQEFBQADggEBAOMAzLKqh6YW60UB6U2Gquydy7EXsRfdbqPbNbsCPkM
EhffPJOmp9TNI0P2eCn7zMcvbfbVc1jdiQESy1N0lUWouYLdZU99TUmRjhYBl34SN5KwVGQcqfDJT2dqfi6M
EvBozRXKgXUdYK37PgUInUKYE5tTaoBTAMwWyo8A+8T1w70Vf17tgbGSAgHEPHFNgCfSnmCWhDJjBnjb
uGB3IzSVL5BLYQod8Jnjso5AJFyMpVGCIJNruvC8U9UcNgjMonqi+7oyd1R9Hs+Ry344KGZuO9ZN8pDpH
IGlWj7tnsFlgM3Og4BDlU3FzefnWeGm/LSKt/o5CCtEgfbOCkgdobn0=
</certificate>
</certificateChain>

<roap:protectedRO
<!-- Will contain either one of the two Domain ROs detailed in section 17.6 of this document.-->
</roap:protectedRO>
</cmlapsshBox>

17.8 CMLA Object Identifiers for DECE

CMLA has defined the following object identifiers (OID)s to be used in certificates.

iso(1)org(3) dod(6) internet(1) private(4) enterprise(1) Content Management License Administrator(21888)

Value of CMLA OID	Name	Description
1.3.6.1.4.1.21888.2	*cmla-kp-dece-ricd*	*DECE Rights Issuer Common Domain*
1.3.6.1.4.1.21888.3	*cmla-kp-dece-dm*	*DECE CMLA DRM Domain Manager*
1.3.6.1.4.1.21888.4	*cmla-kp-dece-dsp*	*DECE CMLA DSP*

17.9 CMLA XML Schema for DECE

The CMLA DECE schema file “cmla-uv-1_0.xsd” is posted at http://www.cm-la.com/tech/cmla-xml.

© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

Appendix A. CMLA IP Source Code

A1. CMLA DDT Exp

CMLA_DDT_EXP.H

```
/*****************************************************************************************************************
*********************
Copyright (c) 2009 CMLA, LLC. All rights reserved.

This material is protected by copyright controlled by CMLA LLC. All rights are reserved.
Copying, reproducing, storing, adapting or translating of any or all of this material requires the prior written consent of
CMLA, LLC.

*****************************************************************************************************************
********************/
#ifndef CMLA_DDT_EXP_H
#define CMLA_DDT_EXP_H

#ifdef __cplusplus
extern "C" {
#endif/**
  * Calculates the Data-Dependent Transformation given by exponentiation.
  *
  * If the %parameter poutput is NULL then only the size of the output
  * is returned (in loutput), otherwise the result of the DDT is written
  * at the location pointed to by poutput.
  *
  * @param pinput pointer to data to transform
  * @param linput length of data
  * @param poutput pointer to hold the transformed data
  * @param loutput pointer to hold the length of the transformed data
  * @note The parameters pinput and poutput must be disjoint
  */

  void CMLA_DDT_exp(const unsigned char *pinput, long linput, unsigned char *poutput, long *loutput);

  /**
  * Calculates the inverse of Data-Dependent Transformation given by exponentiation.
  *
  * If the %parameter poutput is NULL then only the size of the output
  * is returned (in loutput), otherwise the result of the DDT is written
  * at the location pointed to by poutput.
  *
  * @param pinput pointer to data to transform
  * @param linput length of data
  * @param poutput pointer to hold the transformed data
  * @param loutput pointer to hold the length of the transformed data
  * @note The parameters pinput and poutput must be disjoint
  */
  void CMLA_DDT_exp_inv(const unsigned char *pinput, long linput, unsigned char *poutput, long *loutput);
#ifdef __cplusplus
}
#endif
#endif
```

CMLA_DDT_EXP.C

```
/*****************************************************************************************************************
*********************
Copyright (c) 2009 CMLA, LLC. All rights reserved.

This material is protected by copyright controlled by CMLA LLC. All rights are reserved.
Copying, reproducing, storing, adapting or translating of any or all of this material requires the prior written consent of
CMLA, LLC.

*****************************************************************************************************************
********************/

#include "cmla_ddt_exp.h"
// function for reducing an a, 0 < a <= 2^32 modulo 2^16+1
static unsigned int modp(unsigned int a)
{
  unsigned int x, y;

  // since a > 0, this must be a = 2^32
  if ( a == 0 )
    return 1;

  // write a = x + 2^16y = y(2^16+1) + x - y = x - y (modulo 2^16+1)
  x = a & 65535;
  y = a >> 16;
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
  return ( x > y ) ? (x-y) : ( 65537 - (y-x) );
}

// function for computing a^e mod 2^16+1
static unsigned int expmodp(unsigned int a, unsigned int e)
{
  unsigned int count = 1, res = 1;

  while ( count < e ) count <<= 1;

  count >>= 1;
  // do a square and multiply
  while ( count > 1 ) {
    if ( count & e ){
      res *= a;
      res = modp(res);
    }
    res *= res;
    res = modp(res);
    count >>= 1;
  }
  res *= a;
  return modp(res);

}

void CMLA_DDT_exp(const unsigned char *pinput, long linput, unsigned char *poutput, long *loutput)
{
  unsigned int a, e, res;

  *loutput = 3;

  // a = B2||B3 + 1 = A+1
  a = ( pinput[1] << 8 ) + pinput[2]+1;
  // e = 2*B1 + 1
  e = ( pinput[0] << 1 ) + 1;
  // res = (a^e mod 2^16+1) - 1
  res = expmodp( a, e )-1;

  // copy result to output
  poutput[2] = res % (1 << 8);
  poutput[1] = (res >> 8) % (1 << 8);
  poutput[0] = pinput[0];
}

// function to invert an odd integer < 2^32 modulo 2^16, using the Extended Euclidean Algorithm
static unsigned int inv( unsigned int a)
{
  unsigned int w0 = 0, w1 = 1, r0 = ( 1 << 16 ), r1 = a, r2, w2, q;
  int sgn_w2 = 1;

  while ( r1 > 0 ) {
    r2 = r0%r1;
    q = r0/r1;
    w2 = q*w1+w0;
    w2 = w2 & 65535;
    sgn_w2 = -sgn_w2;

    r0 = r1;
    r1 = r2;
    w0 = w1;
    w1 = w2;
  }

  if (sgn_w2 == 1)
    return ( 1 << 16 ) - w0;
  else
    return w0;

}

void CMLA_DDT_exp_inv(const unsigned char *pinput, long linput,
                      unsigned char *poutput, long *loutput){
  unsigned int a, e, res;

  // a = B2||B3 + 1 = A+1
  a = ( pinput[1] << 8 ) + pinput[2]+1;
  // e = 2*B1 + 1
  e = ( pinput[0] << 1 ) + 1;
  // e = (2*B1 + 1)^-1 modulo 2^16+1
  e = inv(e);
  // res = (a^e mod 2^16+1) - 1
  res = expmodp( a, e )-1;

  // copy result to output
  poutput[2] = res % (1 << 8);
  poutput[1] = (res >> 8) % (1 << 8);
  poutput[0] = pinput[0];
}
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

A2. CMLA DDT Perm

CMLA_DDT_PERM.H

```
/*****************************************************************************************************************
*********************
Copyright (c) 2009 CMLA, LLC. All rights reserved.

This material is protected by copyright controlled by CMLA LLC. All rights are reserved.
Copying, reproducing, storing, adapting or translating of any or all of this material requires the prior written consent of
CMLA, LLC.

*****************************************************************************************************************
********************/
#ifndef CMLA_DDT_PERM_H
#define CMLA_DDT_PERM_H

#ifdef __cplusplus
extern "C" {
#endif /**
  * Calculates the Data-Dependent Transformation given by permutation
  *
  * If the %parameter poutput is NULL then only the size of the output
  * is returned (in loutput), otherwise the result of the DDT is written
  * at the location pointed to by poutput.
  *
  * @param pinput pointer to data to transform
  * @param linput length of data
  * @param poutput pointer to hold the transformed data
  * @param loutput pointer to hold the length of the transformed data
  * @note The parameters pinput and poutput must be disjoint
  */
  void CMLA_DDT_perm(const unsigned char *pinput, long linput, unsigned char *poutput, long *loutput);

  /**
  * Calculates the inverse of the Data-Dependent Transformation given by permutation
  *
  * If the %parameter poutput is NULL then only the size of the output
  * is returned (in loutput), otherwise the result of the DDT is written
  * at the location pointed to by poutput.
  *
  * @param pinput pointer to data to transform
  * @param linput length of data
  * @param poutput pointer to hold the transformed data
  * @param loutput pointer to hold the length of the transformed data
  * @note The parameters pinput and poutput must be disjoint
  */
  void CMLA_DDT_perm_inv(const unsigned char *pinput, long linput, unsigned char *poutput, long *loutput);

#ifdef __cplusplus
}
#endif
#endif
```

CMLA_DDT_PERM.C

```
/*****************************************************************************************************************
*********************
Copyright (c) 2009 CMLA, LLC. All rights reserved.

This material is protected by copyright controlled by CMLA LLC. All rights are reserved.
Copying, reproducing, storing, adapting or translating of any or all of this material requires the prior written consent of
CMLA, LLC.

*****************************************************************************************************************
********************/

#include <string.h>
#include "cmla_ddt_perm.h"
/* function for doing the actual permutation */
static void dopermute( const unsigned char in, unsigned char *out, int i )
{
  *out = 0x00;
  switch(i) {
  case 0:
    *out |= ( in & 0x10 ) ?  0x01 : 0x00;
    *out |= ( in & 0x01 ) ?  0x02 : 0x00;
    *out |= ( in & 0x08 ) ?  0x04 : 0x00;
    *out |= ( in & 0x20 ) ?  0x08 : 0x00;
    *out |= ( in & 0x04 ) ?  0x10 : 0x00;
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
    *out |= ( in & 0x40 ) ?  0x20 : 0x00;
    *out |= ( in & 0x02 ) ?  0x40 : 0x00;
    break;
  case 1:
    *out |= ( in & 0x04 ) ?  0x01 : 0x00;
    *out |= ( in & 0x10 ) ?  0x02 : 0x00;
    *out |= ( in & 0x20 ) ?  0x04 : 0x00;
    *out |= ( in & 0x40 ) ?  0x08 : 0x00;
    *out |= ( in & 0x08 ) ?  0x10 : 0x00;
    *out |= ( in & 0x02 ) ?  0x20 : 0x00;
    *out |= ( in & 0x01 ) ?  0x40 : 0x00;
    break;
  case 2:
  case 7:
    *out |= ( in & 0x08 ) ?  0x01 : 0x00;
    *out |= ( in & 0x04 ) ?  0x02 : 0x00;
    *out |= ( in & 0x40 ) ?  0x04 : 0x00;
    *out |= ( in & 0x02 ) ?  0x08 : 0x00;
    *out |= ( in & 0x20 ) ?  0x10 : 0x00;
    *out |= ( in & 0x01 ) ?  0x20 : 0x00;
    *out |= ( in & 0x10 ) ?  0x40 : 0x00;
    break;
  case 3:
  case 6:
    *out |= ( in & 0x20 ) ?  0x01 : 0x00;
    *out |= ( in & 0x08 ) ?  0x02 : 0x00;
    *out |= ( in & 0x02 ) ?  0x04 : 0x00;
    *out |= ( in & 0x01 ) ?  0x08 : 0x00;
    *out |= ( in & 0x40 ) ?  0x10 : 0x00;
    *out |= ( in & 0x10 ) ?  0x20 : 0x00;
    *out |= ( in & 0x04 ) ?  0x40 : 0x00;
    break;
  case 4:
    *out |= ( in & 0x40 ) ?  0x01 : 0x00;
    *out |= ( in & 0x20 ) ?  0x02 : 0x00;
    *out |= ( in & 0x01 ) ?  0x04 : 0x00;
    *out |= ( in & 0x10 ) ?  0x08 : 0x00;
    *out |= ( in & 0x02 ) ?  0x10 : 0x00;
    *out |= ( in & 0x04 ) ?  0x20 : 0x00;
    *out |= ( in & 0x08 ) ?  0x40 : 0x00;
    break;
  case 5:
    *out |= ( in & 0x02 ) ?  0x01 : 0x00;
    *out |= ( in & 0x40 ) ?  0x02 : 0x00;
    *out |= ( in & 0x10 ) ?  0x04 : 0x00;
    *out |= ( in & 0x04 ) ?  0x08 : 0x00;
    *out |= ( in & 0x01 ) ?  0x10 : 0x00;
    *out |= ( in & 0x08 ) ?  0x20 : 0x00;
    *out |= ( in & 0x20 ) ?  0x40 : 0x00;
    break;
  }
}

/* function for permuting the i'th block */
static void permute( const unsigned char *pinput, unsigned char *poutput, int i ) {
  unsigned char temp, res;

  // temp = i'th group of 7 bits with 0 zero in front
  temp = ( (( unsigned char ) ( pinput[i] << (8-i) )) >> 1 ) | ( pinput[i + 1] >> ( i + 1 ) );

  // if b_i = 1 do the permutation, otherwise just copy
  if ( pinput[0] & ( 1 << (7-i) ))
    dopermute( temp, &res, i );
  else
    res = temp;

  // copy the result to the right place in the output
  poutput[i] |= res >> (7-i);
  if ( i < 7 )
    poutput[i+1] = (unsigned char) ( res << ( i + 1 ) );
}

/* function for doing the inverse permutation of the i'th block */
static void permute_inv( const unsigned char *pinput, unsigned char *poutput, int i ) {
  unsigned char temp, res;

  // temp = i'th group of 7 bits with 0 zero in front
  temp = ( (( unsigned char ) ( pinput[i] << (8-i) )) >> 1 ) | ( pinput[i + 1] >> ( i + 1 ) );

  // if b_i = 1 do the inverse permutation, otherwise just copy
  if ( pinput[0] & ( 1 << (7-i) ))
    dopermute( temp, &res, ( i < 6 ) ? ( 5 - i ) : ( i - 4 ) );
  else
    res = temp;

  // copy the result to the right place in the output
  poutput[i] |= res >> (7-i);
  if ( i < 7 )
    poutput[i+1] = (unsigned char) ( res << ( i + 1 ) );
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

```
}

void CMLA_DDT_perm(const unsigned char *pinput, long linput,
                    unsigned char *poutput, long *loutput) {

  int i;

  *loutput = 8;
  memset(poutput, 0, 8);

  // run through the 8 blocks, and do the translation
  for ( i = 0; i < 8; i++ ) {
    permute( pinput, poutput, i );
  }

  // copy the b_i's
  poutput[0] = pinput[0];

}

void CMLA_DDT_perm_inv(const unsigned char *pinput, long linput,
                        unsigned char *poutput, long *loutput) {

  int i;

  *loutput = 8;
  memset(poutput, 0, 8);

  // run through the 8 blocks, and do the inverse translation
  for ( i = 0; i < 8; i++ ) {
    permute_inv( pinput, poutput, i );
  }

  // copy the b_i's
  poutput[0] = pinput[0];
 }
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

Appendix B. CMLA IP Test Vectors

B1. CMLA IP Test Vector

```
# ------------------------------
# Components of the RSA Key Pair
# ------------------------------

# RSA modulus n:
a2 ba 40 ee 07 e3 b2 bd 2f 02 ce 22 7f 36 a1 95
02 44 86 e4 9c 19 cb 41 bb bd fb ba 98 b2 2b 0e
57 7c 2e ea ff a2 0d 88 3a 76 e6 5e 39 4c 69 d4
b3 c0 5a 1e 8f ad da 27 ed b2 a4 2b c0 00 fe 88
8b 9b 32 c2 2d 15 ad d0 cd 76 b3 e7 93 6e 19 95
5b 22 0d d1 7d 4e a9 04 b1 ec 10 2b 2e 4d e7 75
12 22 aa 99 15 10 24 c7 cb 41 cc 5e a2 1d 00 ee
b4 1f 7c 80 08 34 d2 c6 e0 6b ce 3b ce 7e a9 a5

# RSA public exponent e:
01 00 01

# Prime p:
d1 7f 65 5b f2 7c 8b 16 d3 54 62 c9 05 cc 04 a2
6f 37 e2 a6 7f a9 c0 ce 0d ce d4 72 39 4a 0d f7
43 fe 7f 92 9e 37 8e fd b3 68 ed df f4 53 cf 00
7a f6 d9 48 e0 ad e7 57 37 1f 8a 71 1e 27 8f 6b

# Prime q:
c6 d9 2b 6f ee 74 14 d1 35 8c e1 54 6f b6 29 87
53 0b 90 bd 15 e0 f1 49 63 a5 e2 63 5a db 69 34
7e c0 c0 1b 2a b1 76 3f d8 ac 1a 59 2f b2 27 57
46 3a 98 24 25 bb 97 a3 a4 37 c5 bf 86 d0 3f 2f

# p's CRT exponent dP:
9d 0d bf 83 e5 ce 9e 4b 17 54 dc d5 cd 05 bc b7
b5 5f 15 08 33 0e a4 9f 14 d4 e8 89 55 0f 82 56
cb 5f 80 6d ff 34 b1 7a da 44 20 88 53 57 7d 08
e4 26 28 90 ac f7 52 46 1c ea 05 54 76 01 bc 4f

# q's CRT exponent dQ:
12 91 a5 24 c6 b7 c0 59 e9 0e 46 dc 83 b2 17 1e
b3 fa 98 81 8f d1 79 b6 c8 bf 6c ec aa 47 63 03
ab f2 83 fe 05 76 9c fc 49 57 88 fe 5b 1d df de
9e 88 4a 3c d5 e9 36 b7 e9 55 eb f9 7e b5 63 b1

# CRT coefficient qInv:
a6 3f 1d a3 8b 95 0c 9a d1 c6 7c e0 d6 77 ec 29
14 cd 7d 40 06 2d f4 2a 67 eb 19 8a 17 6f 97 42
aa c7 c5 fe a1 4f 22 97 66 2b 84 81 2c 4d ef c4
9a 80 25 ab 43 82 28 6b e4 c0 37 88 dd 01 d6 9f

# ------------------------------
# Setting
# ------------------------------

# seed Z:
7c 09 a0 0a c8 05 88 6e 80 42 7a f1 0f e9 ef 21
d9 74 4c c4 85 51 90 3c ec a8 d9 24 45 ff 7d 6c
31 04 67 dc 31 aa a5 68 fe 45 e9 cb 47 e7 36 fb
53 5f ef 87 81 e2 d2 fd 09 13 d8 e1 03 d5 ac 82
18 6b 16 2b 9c 25 ff 38 50 d6 90 8a 9c 78 c9 88
d3 59 ad 5d 7b f7 66 8f de 64 20 a8 35 5f 45 6d
b1 c6 b2 9c 1a 18 94 d3 83 2f 21 86 0a 92 eb 9d
3e 69 6c 2e 59 24 f9 23 d1 32 d2 f1 73 3b ff b4


# MacKey :
1d 55 bd 0c 59 d7 90 0c 9c 16 3a 15 0d d3 09 f7


# REK :
95 27 57 09 93 c7 d9 77 02 8d ae b2 2c 34 5c 66
```

```
# ------------------------------
# CMLA_RSA_ENCRYPT
# ------------------------------

# CMLA_DDT_Exp (Z) = m:
7c 69 78 0a c8 05 88 6e 80 42 7a f1 0f e9 ef 21
d9 74 4c c4 85 51 90 3c ec a8 d9 24 45 ff 7d 6c
31 04 67 dc 31 aa a5 68 fe 45 e9 cb 47 e7 36 fb
53 5f ef 87 81 e2 d2 fd 09 13 d8 e1 03 d5 ac 82
18 ee 42 2b 9c 25 ff 38 50 d6 90 8a 9c 78 c9 88
d3 59 ad 5d 7b f7 66 8f de 64 20 a8 35 5f 45 6d
b1 c6 b2 9c 1a 18 94 d3 83 2f 21 86 0a 92 eb 9d
3e 69 6c 2e 59 24 f9 23 d1 32 d2 f1 73 3b ff b4



# RSA_Encrypt (m) = C1:
83 d7 e0 e0 fd 69 c7 d9 df c7 83 84 88 ce f1 ab
b7 96 ca 84 65 14 01 72 16 37 41 90 c5 2d 88 17
dd 85 39 38 ef 15 28 5f 7a 0d dc e5 ed c7 4d 10
66 61 e7 e9 f1 d7 c4 7c 21 02 c4 29 45 bb 8f 07
da 57 65 21 da 83 a3 e6 a3 f3 92 ca 9d eb e3 e1
cd 1d f3 e1 05 4c 64 ff 4c 93 18 dc 56 1c b5 4e
e8 d2 5a 5f 41 33 d8 07 81 0a 82 4c ca 5f 7d aa
ca 9b cb 9e 83 3f aa 79 c8 ee c6 9d b1 19 22 c1

# ------------------------------
# CMLA_KDF
# ------------------------------


# CMLA_KDF (Z) = KEK :
74 98 52 e9 dd 93 9a 34 5c 80 57 81 f7 2c 7e de

# ------------------------------
# CMLA_WRAP
# ------------------------------

#CMLA_DDT_Perm (MacKey) = k0 :
1d 55 bd 0d 8e 2b d0 03 9c 16 3a 15 0d d3 09 f7

#CMLA_DDT_Perm (REK) = k1 :
95 87 57 0a a3 cf 59 7e 02 8d ae b2 2c 34 5c 66

# AES_WRAP (KEK, k0|k1) = C2 :
96 5d e9 95 93 61 c2 2f d9 93 d8 48 22 f4 27 1a
41 3c 87 99 84 7f 75 20 49 e2 8d f4 a0 71 ce da
a8 64 0d 9e 12 3f 7f a4



# ------------------------------
# CMLAIP-1
# ------------------------------

# C1|C2 = cipherValue :
83 d7 e0 e0 fd 69 c7 d9 df c7 83 84 88 ce f1 ab
b7 96 ca 84 65 14 01 72 16 37 41 90 c5 2d 88 17
dd 85 39 38 ef 15 28 5f 7a 0d dc e5 ed c7 4d 10
66 61 e7 e9 f1 d7 c4 7c 21 02 c4 29 45 bb 8f 07
da 57 65 21 da 83 a3 e6 a3 f3 92 ca 9d eb e3 e1
cd 1d f3 e1 05 4c 64 ff 4c 93 18 dc 56 1c b5 4e
e8 d2 5a 5f 41 33 d8 07 81 0a 82 4c ca 5f 7d aa
ca 9b cb 9e 83 3f aa 79 c8 ee c6 9d b1 19 22 c1
96 5d e9 95 93 61 c2 2f d9 93 d8 48 22 f4 27 1a
41 3c 87 99 84 7f 75 20 49 e2 8d f4 a0 71 ce da
a8 64 0d 9e 12 3f 7f a4
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending

B2. CMLA IP Test Vector

```
# ------------------------------
# Components of the RSA Key Pair
# ------------------------------

# RSA modulus n:
bb 6f 15 cb dc 0f 48 88 86 d6 3d 5b d0 35 81 ca
ca 66 e0 86 07 a1 cd ef ae d1 6a 3e 67 e9 7f 12
28 86 43 40 e8 96 d1 d3 f1 01 e7 4e f2 33 e6 60
26 13 d6 cc 22 d2 f0 70 3d c2 6e 34 24 36 85 5b
af b0 94 c1 07 d7 2d 18 5d 77 e1 33 2c 85 84 81
e6 64 0c 6d 9f 39 9b 44 67 d7 ac 77 b1 71 51 aa
2f a4 7d a0 72 7c cb 9a 35 08 70 98 6e 4a f6 c4
f8 61 7f 15 25 ce 4e da 04 2a 44 1e 54 73 72 33

# RSA public exponent e:
01 00 01

# Prime p:
dc 60 76 8c fa c1 d1 55 96 76 7f a0 e4 13 ac fe
db cb 25 9b f8 75 ec 27 44 ea 9f 50 b1 8d 0c e7
19 9d 03 0d e2 ba 3c 76 8e ec f2 49 c0 4e 11 2f
d2 aa 9f 9d ef d4 7c c9 2b 78 05 28 5c 11 0d bb

# Prime q:
d9 bb 64 e1 73 81 34 07 51 9c 68 5a 00 90 00 57
6a 19 46 9f 09 b6 c3 aa 60 54 5e 55 fe 44 0e 7c
a4 ef 64 12 c5 e5 06 92 15 ec e7 81 70 22 e4 f7
2f 39 f3 9a 75 01 a7 08 0a 7b 60 2c 40 01 29 e9

# p's CRT exponent dP:
b6 d2 ab e6 ff 2c 85 9e ac 69 78 2e 20 a8 96 0f
04 86 7e 97 eb 42 e0 fc 1e f7 49 dd e6 be 2e 63
16 63 a4 a2 03 63 1f 3f f8 08 68 64 78 ce 0b 02
12 92 0c 43 39 30 9d a9 42 8f 9a 2f c6 59 3b 7b

# q's CRT exponent dQ:
83 de 26 ff 14 81 90 0d 4e b4 37 cf 2a c0 0c 34
e5 21 61 d0 38 85 e4 83 6f cc 29 46 53 b4 cc 41
59 73 53 5c bb 56 36 60 8b be eb 87 4d 6f 14 d5
50 58 fc dc f3 38 88 fd 29 bc 07 47 45 8c 6e e1

# CRT coefficient qInv:
32 81 1c be 78 ff 22 ff 25 b7 0d 53 bc 4c 36 e3
4b 99 5d dc fd ba 1a b9 6a ae 90 cc a0 8a ca b0
31 07 6f ca 45 95 82 3c a2 5d ba 86 7c 00 28 98
54 d8 ab 2d f5 d5 ac a9 b7 a6 a7 72 d1 ed 3b 47

# ------------------------------
# Setting
# ------------------------------

# seed Z:
18 5e 03 e2 ee 6b f4 07 9f cb 1e 62 a5 f3 b2 f6
51 15 eb c1 1f 93 81 4d f3 eb e9 34 bd 2e 09 d0
51 5b 6c ca 4c 6a da 45 bc c3 f3 b9 58 05 70 a1
46 f0 e2 7b 66 e2 83 43 8f c2 7b 30 09 66 48 08
1c b7 26 0d 0c 28 1e be 81 69 b8 11 ca d3 90 a3
81 2e 24 15 12 63 6a be f3 03 67 0b 2f 05 d6 df
ce f0 c6 90 46 dd b7 b6 a2 b5 5b 1b eb 25 94 9e
7f ee 87 15 58 f0 b4 31 fd ac 6c 6c a6 01 dc 10

# MacKey :
4c 52 93 00 58 37 21 a5 ba 72 35 41 5f 9b a9 8d

# REK :
83 81 10 bd 01 63 2f b7 2e 5a a0 43 55 43 2d c9
```

```
# ------------------------------
# CMLA_RSA_ENCRYPT
# ------------------------------

# CMLA_DDT_Exp (Z) = m:
18 9c 2b e2 ee 6b f4 07 9f cb 1e 62 a5 f3 b2 f6
51 15 eb c1 1f 93 81 4d f3 eb e9 34 bd 2e 09 d0
51 5b 6c ca 4c 6a da 45 bc c3 f3 b9 58 05 70 a1
46 f0 e2 7b 66 e2 83 43 8f c2 7b 30 09 66 48 08
1c 7a 0a 0d 0c 28 1e be 81 69 b8 11 ca d3 90 a3
81 2e 24 15 12 63 6a be f3 03 67 0b 2f 05 d6 df
ce f0 c6 90 46 dd b7 b6 a2 b5 5b 1b eb 25 94 9e
7f ee 87 15 58 f0 b4 31 fd ac 6c 6c a6 01 dc 10

# RSA_Encrypt (m) = C1:
5d 98 4f e5 29 15 cd e5 39 e5 9f 61 96 da 04 e4
07 08 dd ab 3d b8 3c c3 a5 98 73 dd da 4a 92 08
5e a8 9f b3 ad 35 0c e9 3f 25 3d 77 fb 52 d7 b0
2c d0 70 c5 4f c3 89 80 3c 4f 8f 7e 74 df f6 d9
4f 1c b6 69 5f fb ad da d0 30 42 4e 3f 12 d3 42
43 3f 19 7b aa 57 03 2b 20 e0 dc 00 7e 89 43 ba
e3 a0 01 99 12 e1 ce 90 22 00 c6 19 b0 2f a9 79
86 2d fa e3 55 ed d5 4a 13 62 ee 5a 30 8b 48 1c

# ------------------------------
# CMLA_KDF
# ------------------------------

# CMLA_KDF (Z) = KEK :
23 f2 70 fd 8a 25 80 5a a4 cc 15 77 1e 29 f8 fc

# ------------------------------
# CMLA_WRAP
# ------------------------------

#CMLA_DDT_Perm (MacKey) = k0 :
4c 52 17 00 50 ab a1 a5 ba 72 35 41 5f 9b a9 8d

#CMLA_DDT_Perm (REK) = k1 :
83 41 10 bd 01 63 3f 7a 2e 5a a0 43 55 43 2d c9

# AES_WRAP (KEK, k0|k1) = C2 :
f0 a3 f8 df 0c bd c9 ce 58 8b 21 05 a6 2b 86 88
bc 7c 6d c0 3b 7c 88 d7 c9 1a d9 65 f6 5a 3a 4c
75 2f ca 37 8a b8 82 7d

# ------------------------------
# CMLAIP-1
# ------------------------------

# C1|C2 = cipherValue :
5d 98 4f e5 29 15 cd e5 39 e5 9f 61 96 da 04 e4
07 08 dd ab 3d b8 3c c3 a5 98 73 dd da 4a 92 08
5e a8 9f b3 ad 35 0c e9 3f 25 3d 77 fb 52 d7 b0
2c d0 70 c5 4f c3 89 80 3c 4f 8f 7e 74 df f6 d9
4f 1c b6 69 5f fb ad da d0 30 42 4e 3f 12 d3 42
43 3f 19 7b aa 57 03 2b 20 e0 dc 00 7e 89 43 ba
e3 a0 01 99 12 e1 ce 90 22 00 c6 19 b0 2f a9 79
86 2d fa e3 55 ed d5 4a 13 62 ee 5a 30 8b 48 1c
f0 a3 f8 df 0c bd c9 ce 58 8b 21 05 a6 2b 86 88
bc 7c 6d c0 3b 7c 88 d7 c9 1a d9 65 f6 5a 3a 4c
75 2f ca 37 8a b8 82 7d
```


© CMLA, LLC. US Patent Nos. 7,564,970 and 7577250; US and foreign patents pending