For Publication on the ICAO Website

TECHNICAL REPORT

LDS2 - PKI

Version 1.1

March 2019

File: TR LDS2 PKI 1.1 March 2019 V1.1.pdf

Author: ISO/IEC JTC1 SC17 WG3/TF5

TABLE OF CONTENTS

1 SCOPE 2

2 OVERVIEW OF THE PUBLIC KEY INFRASTRUCTURE 3

3 ROLES AND RESPONSIBILITIES 4

3.1 DIGITAL SIGNATURE PKI ROLES AND RESPONSIBILITIES 4
3.2 AUTHORIZATION PKI ROLES AND RESPONSIBILITIES 5

4 KEY MANAGEMENT 8

4.2 CRYPTOGRAPHIC ALGORITHMS FOR DIGITAL SIGNATURE PKI 9
4.3 CRYPTOGRAPHIC ALGORITHMS FOR TERMINAL AUTHENTICATION 9
4.4 CRYPTOGRAPHIC ALGORITHMS FOR SPOC 10

5 DISTRIBUTION MECHANISMS 11

6 PKI TRUST AND VALIDATION 12

6.1 VALIDATION OF X.509 CERTIFICATES 12
6.2 VALIDATION OF CARD-VERIFIABLE CERTIFICATES 13

7 SIGNATURE PKI CERTIFICATE PROFILES 15

7.1 CSCA CERTIFICATE PROFILE 15
7.2 CHANGES TO CSCA CERTIFICATE PROFILE TO CATER FOR LDS 2 15
7.3 LDS2 SIGNER CERTIFICATE PROFILE 15

8 AUTHORIZATION PKI CERTIFICATE PROFILES 16

8.1 SPOC CERTIFICATE PROFILE 16
8.2 CVCA, DV AND TERMINAL CERTIFICATE PROFILES 17
8.3 DATA OBJECTS 19

9 SPOC PROTOCOL 22

9.1 SPOC PROTOCOL MESSAGES 22
9.2 WEB SERVICE 27

REFERENCES 33

APPENDIX A 34

1 SCOPE

This document specifies the PKI to support the ICAO LDS2 project including its three applications:

- Travel records (stamps);
- Visa records; and
- Additional biometrics.

The LDS2 PKI supports two basic functions.

The first function is authenticity and integrity of LDS2 data objects written to the contactless IC of an eMRTD for all three applications, through the use of digital signatures and their verification.

The second function is authorization. The authorization PKI enables an eMRTD Issuing State or organization to authorize the writing of LDS2 data objects to the contactless IC of their eMRTDs after issuance. It also enables the eMRTD Issuing State or organization to authorize read access to LDS2 data objects. These read/write authorizations MAY be granted to foreign States at the discretion of the Issuing State or organization.

This document covers only the PKI aspects of LDS2 and is intended to be integrated into a Technical Report on LDS2 along with material covering other aspects.

2 OVERVIEW OF THE PUBLIC KEY INFRASTRUCTURE

The LDS2 PKI is specific to the three LDS2 applications and has no impact on the use of PKI for Passive Authentication for the LDS1 application.

The LDS2 PKI is comprised of two independent infrastructures. The digital signature PKI is used by the State writing LDS2 data objects and provides integrity and authenticity of those data objects. The authorization PKI enables the eMRTD Issuing State or organization to control and manage the foreign States that are given authorization to write LDS2 data objects to their eMRTDs and to read those data objects. A foreign State intending to read or write LDS2 data must obtain an authorization certificate directly from the eMRTD Issuing State or organization.

The **LDS2 digital signature PKI** is specified as a set of enhancements to the X.509-based PKI used in LDS1 for Passive Authentication.. The same CSCA that is used for LDS1 is also used for LDS2. The LDS1 CSCA issues LDS2 Signer Certificates as defined in this document.

LDS2 Signer certificates MUST comply with the certificate profiles defined in Section 7.

The LDS2 Digital Signature PKI consists of the following entities:

- Country Signing CA (CSCA)
- LDS2 Signers

The **LDS2 authorization PKI** uses a different certificate structure (ISO 7816 card verifiable certificates) and therefore requires additional infrastructure components.

LDS2 requires the terminal to prove to the eMRTD contactless IC that it is entitled to write LDS2 data objects to the contactless IC or that it is entitled to read LDS2 data objects. Such a terminal is equipped with at least one private key and the corresponding Terminal Certificate, encoding the terminal's public key and access rights. After the terminal has proven knowledge of this private key, the MRTD chip grants the terminal access to read/write LDS2 data as indicated in the Terminal Certificate.

The LDS2 authorization PKI consists of the following entities:

- Country Verifying CAs (CVCAs)
- Document Verifiers (DVs)
- Terminals
- Single Point of Contact (SPOC)

Distribution and management of the authorization certificates between CVCAs in one State and DVs in other States is handled through a Single Point of Contact (SPOC) in each State.

3 ROLES AND RESPONSIBILITIES

The LDS2 application is written to the contactless IC of an eMRTD, by the Issuing State or organization at the time of personalization.

Before another State can write LDS2 objects to that contactless IC, it MUST obtain authorization from the Issuing State or organization to do so. Each LDS2 data object is digitally signed by an LDS2 Signer in the writing State and subsequently written to the contactless IC by an authorized terminal in that writing State. The two step process of signing by a signer and writing by an authorized terminal is similar to the LDS1 concept where the Document Signer digitally signs Document Security Objects but they are subsequently written to the contactless IC through the personalization process, as illustrated in Figure 1. Subsequent reading of LDS2 objects from the contactless IC is done through terminals authorized for LDS2 reading of the LDS2 object type in question.

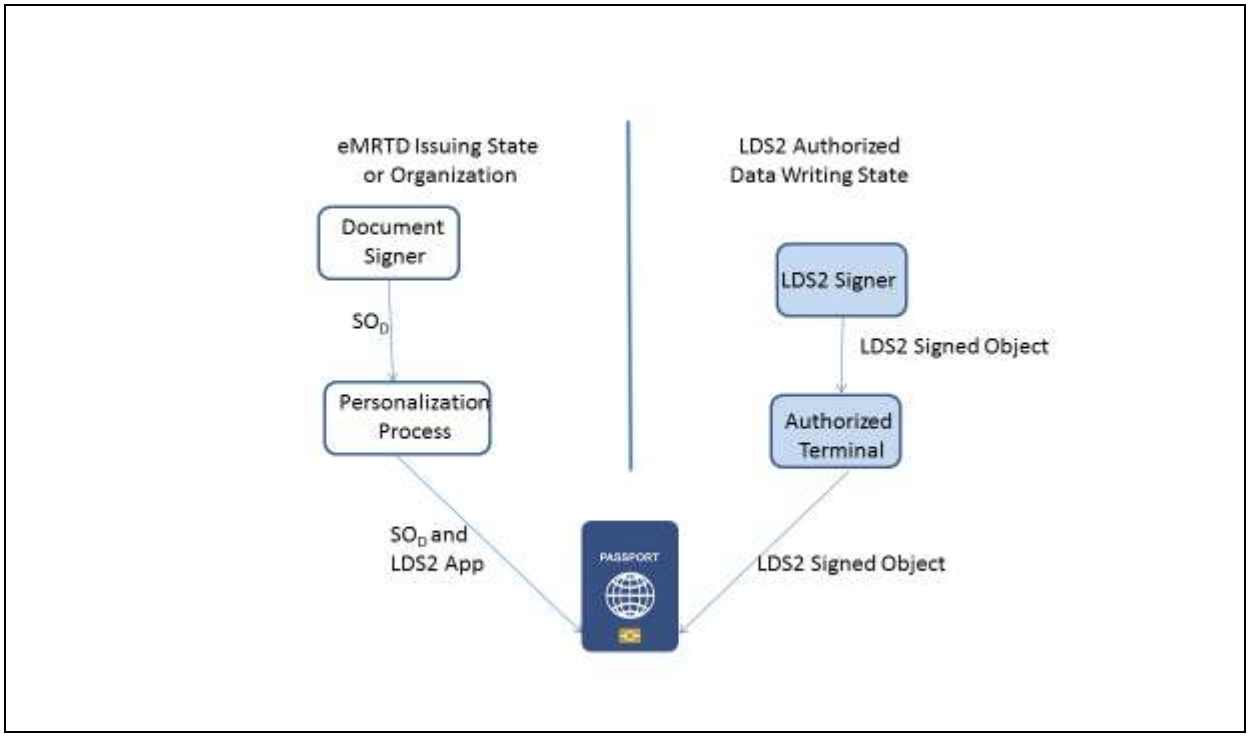

Figure 1: LDS2 Writing Architecture

There are two distinct PKI infrastructures used within LDS2. The Digital Signature PKI, which is the same as the LDS1 PKI is used to ensure and verify the integrity and authenticity of signed LDS2 objects. The Authorization PKI is used to manage authorization of States to read LDS2 objects from, and write LDS2 objects to eMRTDs.

The roles and responsibilities of each entity in these two infrastructures are outlined below.

3.1 Digital Signature PKI Roles and Responsibilities

The LDS2 application adds three new types of data that may be stored on the contactless IC of an eMRTD. The authenticity and integrity of that data is protected by the creation and verification of digital signatures on those data objects.

1. **Country Signing CA (CSCA**): The CSCA issues certificates to LDS2 Signer for one or more of the LDS2 data types. The CSCA issues a single CRL that covers revocation notices for all types of certificates it issues including CSCA Certificates, Document Signers, Master List Signers, Deviation List Signers and LDS2 Signer Certificates.

2. **LDS2 Signer**: An LDS2 Signer digitally signs LDS2 data objects of one or more types.

Where there is a need to refer to an LDS2 Signer as one that signs a particular LDS2 data object type, it is referred to as follows:

- LDS2-TS Signer – signs LDS2 Travel Stamps
- LDS2-V Signer – signs LDS2 Electronic Visas
- LDS2-B Signer – signs LDS2 Additional Biometrics

It is RECOMMENDED that each State have no more than one LDS2-TS Signer, one LDS2-V Signer and one LDS2-B Signer. It is also possible for one LDS2 Signer to combine some/all of these roles.

If further differentiation is required, such as the location a travel stamp was added, the individual officer who cleared a traveler, which officer granted a visa, or the location at which additional biometrics were added, these details can be included in a proprietary field within the respective LDS2 data object itself.

Figure 2 illustrates the trust model for the Digital Signature PKI.

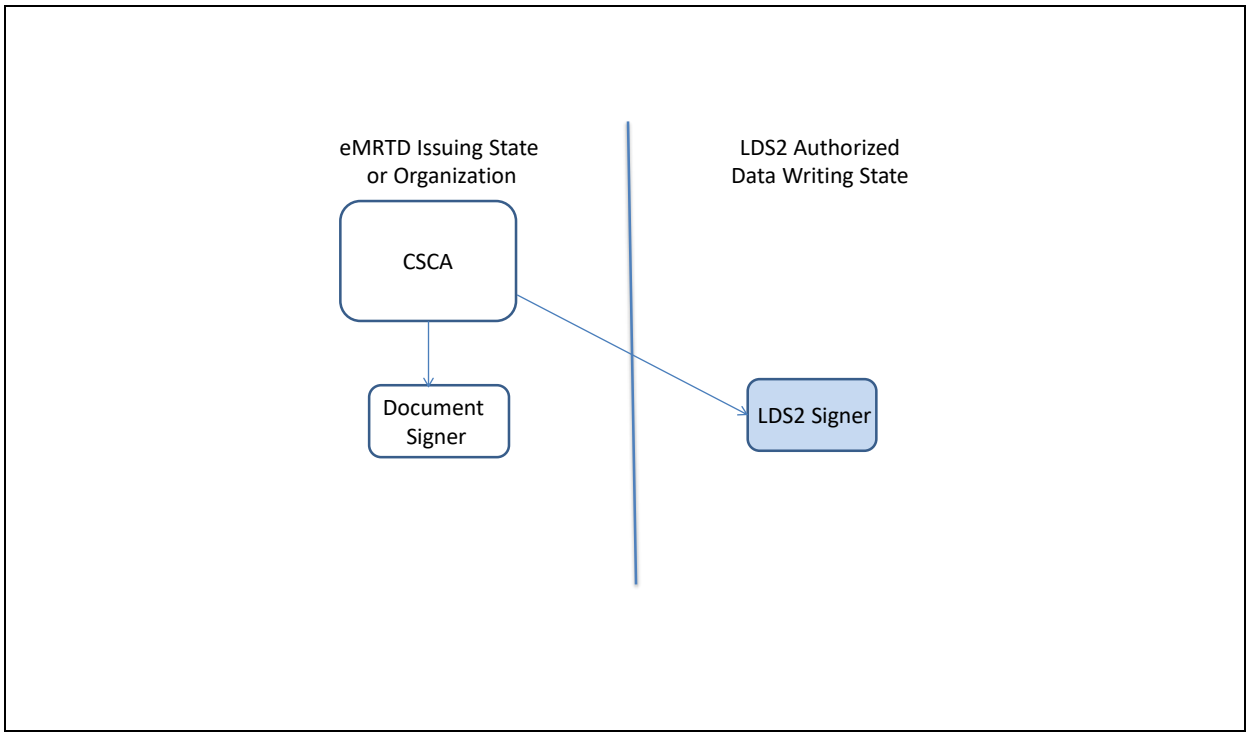

Figure 2: Digital Signature PKI Trust Model

3.2 Authorization PKI Roles and Responsibilities

The authorization PKI enables the eMRTD Issuing State or organization to control access (read and write) to LDS2 data on contactless ICs in eMRTDs it issues.

1. **Country Verifying CA (CVCA):** Each issuing State or organiztion that allows LDS2 data to be added to its eMRTDs MUST set up a single CVCA. This CVCA is a Certification Authority (CA) that is the trust anchor for the authorization PKI of that State or organization and covers all the LDS2 applications. The CVCA may be a stand-alone entity or it may be integrated with the CSCA of that same State or organization. However, even if co-located, the CVCA MUST use a different key pair than that of the CSCA. The CVCA determines the access rights that will be granted to all Document Verifiers (DV), foreign and domestic and issues certificates containing the individual authorizations to each of those DVs.

2. **Document Verifier (DV):** A Document Verifier is a CA that, as part of an organizational unit that manages a group of terminals (e.g. terminals operated by a State's border police) and issues authorization certificates to those terminals. A DV MUST have already received an authorization certificate from the responsible CVCA before it can issue associated certificates to its terminals. Certificates issued by a DV to terminals MAY contain the same authorization, or a subset, that has been granted to the DV. They MUST NOT contain any authorization beyond that granted to the DV.

3. **Terminal/Inspection System (IS)**: Within the context of the authorization PKI, a terminal is the entity that accesses the contactless IC of an eMRTD and writes a digitally signed LDS2 data object, or reads an LDS2 data object. The terminal MUST have an authorization certificate issued to it, from its local DV that grants the required authorization. The terminal is also referred to as an Inspection System.

4. **Single Point of Contact (SPOC):** Each State that participates in the LDS2 authorization PKI MUST set up a single SPOC. This SPOC is the interface that is used for all communication between the CVCA of one State with the DVs in another State. Certificate requests and responses are communicated between the SPOCs of each State using the SPOC protocol defined in Section 9.

Figure 3 illustrates the trust model for the Authorization PKI.

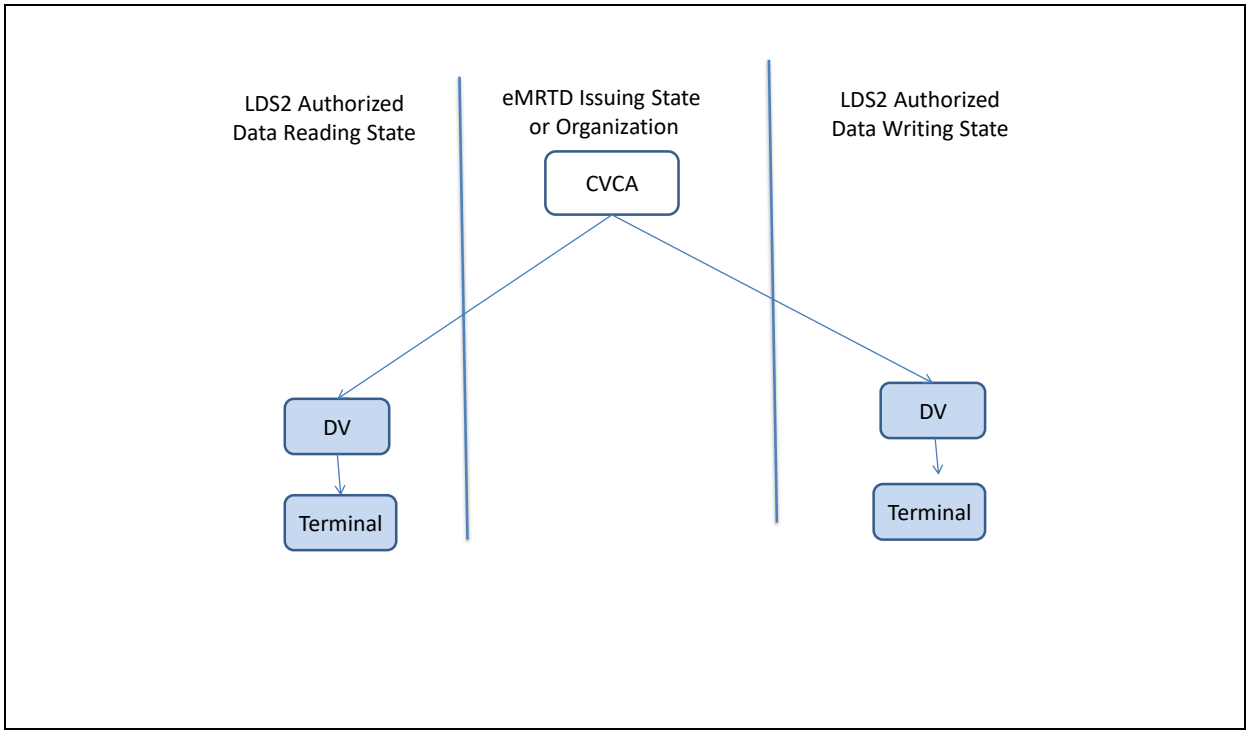

Figure 3: Authorization PKI Trust Model

Figure 4 illustrates the role of SPOC as a communication tool for the Authorization PKI.

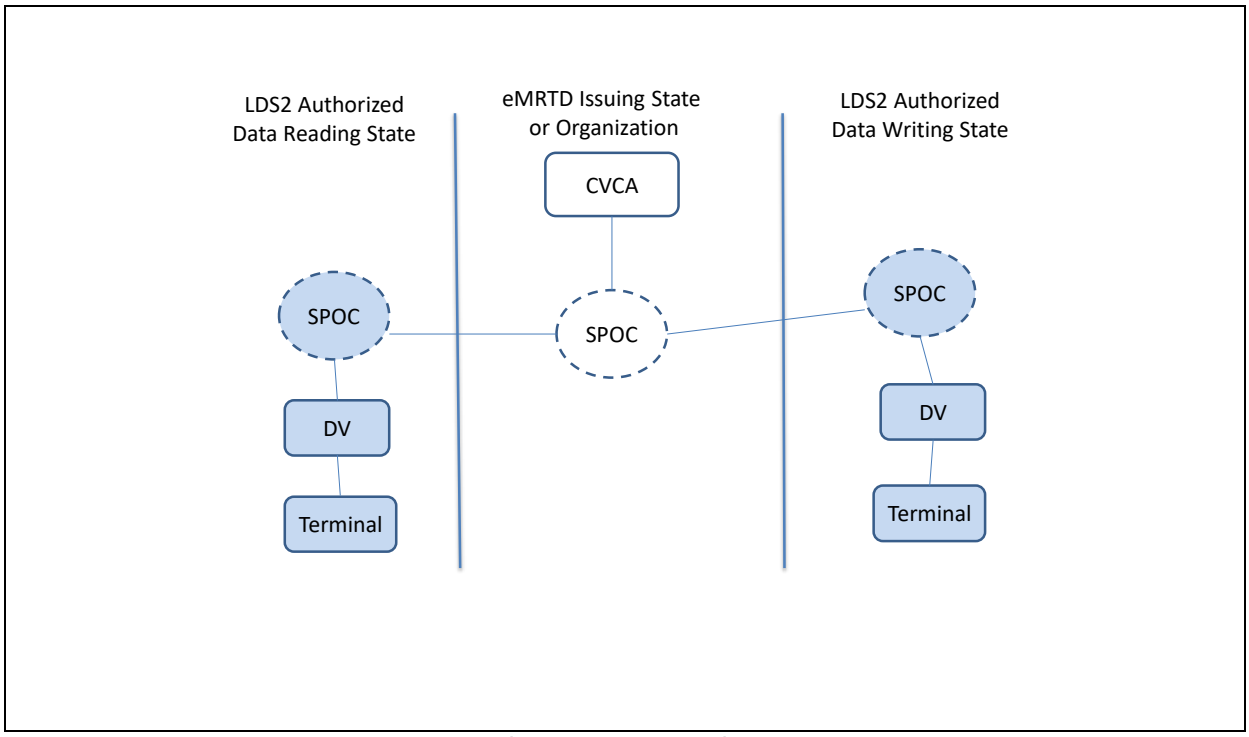

Figure 4: SPOC Role

4 KEY MANAGEMENT

Issuing States or organizations MAY have additional key pair types for the LDS2 application:

- LDS2 Signer Key Pair;
- Country Verifying CA (CVCA) Key Pair;
- Document Verifier (DV) Key Pair;
- Terminal Key Pair;
- SPOC client Key Pair; and
- SPOC server Key Pair.

LDS2 Signer, SPOC client and SPOC server public keys are certified by the CSCA through issuance of X.509 certificates. All X.509 certificates MUST comply with their respective certificate profiles defined in Section 7.

LDS2 Signer key pairs are similar to Document Signer key pairs in that the usage period of the private key is much shorter than the validity period of the corresponding certificate. The certificates MUST remain valid for the lifetime of the eMRTD or the signed LDS2 object (whichever is longest). Because signed data objects will be written to eMRTDs from various States, these certificates MUST be valid for at least the duration of the longest eMRTD lifetime (i.e. 10 years).

4.1.1 LDS2 Signer Public Key Validity

The lifetime, i.e. the certificate validity period, of the LDS2 Signer public key is determined by concatenating the following two periods:

- The length of time the corresponding private key will be used sign LDS2 objects, with;
- The validity period of whichever of the following is longest:
  - Any eMRTD that will store an LDS2 object signed with that key; or
  - Any LDS2 object signed with that key. Note that in the case of LDS2 eVisa, it is possible for the validity period of a signed eVisa to extend beyond the validity period of the eMRTD including that visa.

4.1.2 Country Signing CA Public Key Validity

The lifetime, i.e. the certificate validity, of the CSCA public key, in an LDS2 infrastructure, is determined by concatenating the following periods:

- The length of time the corresponding CSCA private key will be used to sign Document Signer; and,
- The key lifetime of Document Signer certificates.

The private key usage period and public key validity period for each of the X.509 key pair types is outlined in Table 1.

Table 1: Key Usage and X.509 Certificate Validity

	Use of Private Key	**Public Key Validity** (assuming 10 year valid passports)
CSCA	3 – 5 years	13 – 15 years
LDS2-TS Signer	1 – 2 years	10 years + 3 months
LDS2-V Signer	1 – 2 years	10 years + 3 months

	Use of Private Key	Public Key Validity (assuming 10 year valid passports)
LDS2-B Signer	3 months – 1 year	10 years + 3 months
SPOC Client	Not specified	6-18 months
SPOC Server	Not specified	6-18 months

The CVCA and DV public keys are certified by the CVCA. The terminal public keys are certified by the DV. CVCA, DV and terminal public key certificates are card-verifiable certificates that MUST comply with their respective certificate profiles defined in Section 7. There is no revocation mechanism for CVCA, DV or terminal certificates. Therefore their validity periods are much shorter than the X.509 certificate types.

The private key usage period is not specified and is up to the discretion of the State. However, the private key usage period MUST be at most equal to the public key validity period. The public key validity period for CVCA, DV and terminal key pairs is outlined in Table 2.

Table 2: Key Usage Card-Verifiable Certificate Validity

	Public Key Validity
CVCA	6 months – 3 years
DV	2 weeks – 3 months
Terminal	1 day – 1 month

4.2 Cryptographic Algorithms for Digital Signature PKI

Because LDS2 certificates and signed objects are stored on the contactless IC, they need to be as compact as possible. Therefore LDS2 Signers MUST use ECDSA, irrespective of the algorithm used in the CSCA and Document Signing keys.

4.3 Cryptographic Algorithms for Terminal Authentication

The algorithm used for Terminal Authentication in the authorization PKI is determined by the CVCA of the eMRTD Issuing State. The same signature algorithm, domain parameters and key sizes MUST be used within a certificate chain (i.e. the CVCA, DV and terminal certificates for a given authorization).[1] CVCA Link Certificates MAY include a public key that deviates from the current parameters, i.e. the CVCA MAY switch to a new signature algorithm, new domain parameters, or key sizes.

For Terminal Authentication, either RSA or ECDSA MAY be used.

4.3.1 RSA

For Terminal Authentication with RSA the following algorithms MUST be used.

RSA [RFC-RSA], [PKCS#1] as specified in Table 3 SHALL be used. The default parameters to be used with RSA-PSS are defined as follows:

- Hash Algorithm: The hash algorithm is selected according to Table 3.
- Mask Generation Algorithm: MGF1 [RFC-RSA], [PKCS#1] using the selected hash algorithm.

[1] As a consequence Document Verifiers and terminals will have to be provided with several key pairs.

- Salt Length: Octet length of the output of the selected hash algorithm.
- Trailer Field: 0xBC

Table 3: Terminal Authentication with RSA

OID	Signature	Hash	Parameters
id-TA-RSA-PSS-SHA-256	RSASSA-PSS	SHA-256	default
id-TA-RSA-PSS-SHA-512	RSASSA-PSS	SHA-512	default

4.3.2 ECDSA

For Terminal Authentication with ECDSA, the plain signature format [TR-03111] as specified in Table 4 SHALL be used.

Table 4: Terminal Authentication with ECDSA

OID	Signature	Hash
id-TA-ECDSA-SHA-224	ECDSA	SHA-224
id-TA-ECDSA-SHA-256	ECDSA	SHA-256
id-TA-ECDSA-SHA-384	ECDSA	SHA-384
id-TA-ECDSA-SHA-512	ECDSA	SHA-512

4.4 Cryptographic Algorithms for SPOC

The TLS Encryption Suites to be used for the SPOC protocol are listed in Table 5.

Table 5: TLS Encryption Suites

Cipher Suite	Certificate and Key Exchange Algorithm
TLS_RSA_WITH_AES_128_CBC_SHA	RSA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE_RSA
TLS_RSA_WITH_AES_256_CBC_SHA	RSA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE_RSA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE_ECDSA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE_ECDSA

In the scope of the TLS handshake negotiation, the client SHALL support all the TLS cipher suites defined in Table 5. Both the server and the client side SHALL support RSA and ECDSA based authentication. It is permissible for a server to request and also for the client to send a client certificate of a different type than the server certificate.

The use of the ECDHE_ECDSA key agreement in TLS handshake is in accordance with the additions defined in [TLSECC], [TLS1.2] and [TLSEXT]. Both the client and the server SHALL support the appropriate Elliptic curves extensions as specified in [TLSECC] specification in the scope of TLS handshake. The supported Elliptic curves and EC Point formats are defined in Section 5 of [TLSECC]. The use of the supported TLS cipher suites defined in Table 4 which uses Advanced Encryption Standard (AES) for encryption SHALL be in accordance with [TLSAES] specification.

5 DISTRIBUTION MECHANISMS

The LDS2 PKI objects that need to be distributed from issuing States or organizations to receiving States include:

- LDS2 Signer certificates;
- Initial CVCA certificates;
- CVCA Link certificates; and
- DV certificates.

The relevant distribution mechanisms for LDS2 PKI objects include:

- eMRTD contactless IC;
- SPOC;
- Bilateral; and
- PKD.

Table 6 outlines the distribution mechanism for each.

Table 6: Distribution of LDS2 PKI Objects

	Contactless IC	***SPOC***	***Bilateral***	***PKD***	***Notes***
LDS2 Signer certificates	Y				Certificates written at same time signed object is written
CVCA Initial Certificate	Y				Certificate written at eMRTD personalization time
CVCA Link Certificates	Y	Y			Certificates distributed to DVs via SPOC and CVCA Trust Anchor updated on contactless IC at next verification
DV Certificates		Y			Distributed only to subject DV
CRLs (Null and Non-null)			Y	Y	CRLs issued by CSCA include revocation information relevant to LDS2 PKI objects

6 PKI TRUST AND VALIDATION

Validation procedures for X.509 certificates (CSCA, SPOC, LDS2 Signer) and card-verifiable certificates (CVCA, DV, Terminal Certificates) are different.

Validation of LDS2 Signer and SPOC certificates follows the same basic validation procedure for LDS 1 certificates as already specified in Doc 9303-12 (7th edition), with a few exceptions. The Trust Anchor for validation of these certificate types is the same CSCA. The characteristics of LDS2 that impose additional requirements on validation are:

Extended Key Usage extension (EKU) MUST be included in all LDS2 Signer certificates. The validation algorithm MUST ensure that the particular EKU value that is relevant to the terminal and current operation is present in all certificates in the path, excluding the CSCA.

6.1 Validation of X.509 Certificates

This validation algorithm simplifies that specified in [RFC 5280] by excluding the variables and processing that are not part of the eMRTD PKI, such as constraints on certificate policies and permitted/excluded named subtrees. The algorithm also adds eMRTD-specific requirements beyond those specified in the [RFC 5280] algorithm.

Conforming implementations are not required to implement this specific algorithm, but MUST provide functionality equivalent to the external behavior resulting from this procedure. Any algorithm may be used by a particular implementation so long as it derives the correct result. This algorithm is written for the LDS2 X.509 certificate types, but could also be used for the LDS1 certificate types including Document Signer, Master List Signer and Deviation List Signer. Differences between validation of LDS1 and LDS2 certificate types are indicated within the algorithm specification.

6.1.1 Path Validation Inputs

The inputs to path validation are:

- Certification Path
- Trust Anchor information,
- The current date and time; and
- CRLs for each certificate in the path.

In LDS2, each certification path contains exactly one certificate (a LDS2 Signer certificate).

Trust anchors, trust anchor management and trust anchor information are defined in Doc 9303-12.

The CRLs that will be needed in the validation are those issued by the same CAs that issued the certificates in the path. The CRL issued by the Trust Anchor CSCA is required to check revocation status of the LDS2 Signer certificate.

Note: For LDS 1 path validation:

- *Each certification path contains exactly one certificate and the AuthorityKeyIdentifier extension of that certificate is used to identify the correct Trust Anchor.*
- *Only the CRL issued by the Trust Anchor CSCA is required.*
- There are no application OIDs input to path validation.

6.1.2 Initialization of Variables

The variables used in the path validation algorithms are:

- `maximum_path_length;`
- `working_issuer_name;`

- working_public_key;
- working_public_key_algorithm; and
- working_public_key_parameters.

The maximum_path_length variable is initialized to value “0” for LDS2 validation, indicating that no CA Certificate is permitted in the path between the Trust Anchor and the certificate being validated (LDS2 Signer or SPOC server/client).

Note: This is identical to LDS1 path validation where this variable is initialized to value “0” indicating that no CA certificates are permitted in the path between the Trust Anchor and the certificate being validated (Document Signer, Master List Signer or Deviation List Signer).

The remaining variables are initialized to values contained in the Trust Anchor information input to the validation as follows:

- working_issuer_name is initialized to the trusted issuer name;
- working_public_key is initialized to the trusted public key;
- working_public_key_algorithm is initialized to the trusted public key algorithm; and
- working_public_key_parameters is initialized to the trusted public key parameters (if present).

6.1.3 Certificate Processing

Each certificate in the path is processed individually, beginning with the certificate signed by the CSCA represented by the Trust Anchor. For LDS 2 validation that would be the same CSCA. Once that certificate has completed processing, the variables are updated before processing of the LDS Signer certificate is begun.

a) Verify that the value of the Issuer field of the certificate and the working_issuer_name variable are identical;
b) Verify the signature on the certificate using the working_public_key; working_public_key_algorithm and any associated working_public_key_parameters;
c) Verify that the current date/time is within the validity period of the certificate;
d) Verify that the EKU extension contains all LDS2 application OIDs provided at input;
e) Verify that there are no unknown critical extensions; and
f) Verify that the certificate has not been revoked.

If any of these checks fails, path validation failed and an error is returned.

If all checks succeed, and this is the final certificate in the path (i.e.: maximum_path_lenth = “0”), path validation succeeds and a notification of success is returned.

Note:This is identical to LDS1 validation, where there is always a single certificate in the path.

6.2 Validation of Card-Verifiable Certificates

For DV and terminal certificates in the authorization PKI, the Trust Anchor is the recent public key of the CVCA of the State that issued the eMRTD. The initial Trust Anchor SHALL be stored securely in the eMRTD contactless IC in the production or (pre-) personalization phase. As the key pair used by the CVCA changes over time, CVCA Link Certificates are produced. The eMRTD contactless IC MUST internally update its Trust Anchor(s) according to received valid link certificates. Due to the scheduling of CVCA Link Certificates, at most two CVCA Trust Anchors will be stored on the contactless IC at any one time.

To validate a Terminal Certificate, the eMRTD contactless IC MUST be provided with a certificate chain starting at a Trust Anchor stored on the eMRTD contactless IC.

The validation procedure for DV and terminal certificates is specific to the LDS2 terminal authentication protocol and is specified in Doc 9303-11.

7 SIGNATURE PKI CERTIFICATE PROFILES

Issuing States or organizations MUST issue certificates and CRLs that conform to the profiles specified below.

The CSCA that issues LDS2 certificates is the same CSCA that issues Document Signer certificates. Revocation notices for LDS2 signer certificates would appear on the same CRL issued by the CSCA that covers revocation notices for all certificate types issued by the CSCA.

Profiles for the certificate types and CRL are defined below.

The profiles use the following terminology for presence requirements of each of the components/extensions:

- m mandatory – the field MUST be present
- x do not use – the field MUST NOT be populated
- o optional – the field MAY be present

The profiles use the following terminology for criticality requirements of extensions that may/must be included:

- c critical – receiving applications MUST be able to process this extension.
- nc non-critical - receiving applications that do not understand this extension MAY ignore it.

7.1 CSCA Certificate Profile

There is no change to the CSCA certificate profile.

7.2 Changes to CSCA Certificate Profile to cater for LDS 2

The CSCA certificates MUST comply with the CSCA certificate profile defined in Doc 9303-12 (7th edition).There are no changes to CSCA profile for use in LDS2.

7.3 LDS2 Signer Certificate Profile

LDS2 Signer certificates MUST comply with the Document Signer certificate profile defined in Doc 9303-12 (7th edition), with the following exceptions:

Subject Field:

The “subject” field of LDS2 Signer certificates MUST be populated as follows:

- countryName: MUST be present. The value contains a country code that MUST follow the format of two letter country codes, specified in Doc 9303-3 (7th edition).
- commonName: MUST be present. The value in this attribute MUST NOT exceed 9 characters in length.
- Other attributes MUST NOT be included.

Certificate extensions:

LDS2 Signer certificates MUST contain the certificate extensions identified in Table 8 below. All other certificate extensions MUST NOT be included.

Table 8: Mandatory Certificate Extensions

Extension name	LDS2 Signer	Comments

Extension name	LDS2 Signer		Comments
	Presence	Criticality	
AuthorityKeyIdentifier	**m**	**nc**	
keyIdentifier	m		
authorityCertIssuer	o		
authorityCertSerialNumber	o		
ExtKeyUsage	**m**	**c**	See note 1

- *Note 1: The EKU extension for each LDS2 Signer certificate type MUST be populated as indicated below. Note that a single LDS2 Signer could be authorized to sign multiple LDS2 data object types. In that case the EKU extension would contain all relevant OIDs for that signer:*
  id-icao-mrtd-security-lds2 OBJECT IDENTIFIER ::= {id-icao-mrtd-security 9}

  id-icao-lds2Signer ::= {id-icao-mrtd-security-lds2 8}

  - *LDS2 Travel Stamp Signer (LDS2-TS) certificates*

    id-icao-tsSigner OBJECT IDENTIFIER ::= { id-icao-lds2Signer 1}

  - *LDS2 Visa Signer (LDS2-V) certificates:*

    id-icao-vSigner OBJECT IDENTIFIER ::= { id-icao-lds2Signer 2}

  - *LDS2 Biometrics Signer (LDS2-B) certificates:*

    id-icao-bSigner OBJECT IDENTIFIER ::= { id-icao-lds2Signer 3}

- *Note 2: LDS2 Signer Certificates must comply with the size restrictions imposed by EF.Certificates in Part 10.*

Although the CRL Distribution Points extension is not included in these certificates, it is mandatory that the revocation status be checked for each certificate as part of the normal validation process. The CRL issued by the CSCA that issued the certificate in question is the CRL used to verify its revocation status.

8 AUTHORIZATION PKI CERTIFICATE PROFILES

The authorization PKI includes X.509 certificates for SPOC and card-verifiable certificates for CVCA, DV and terminals. The profile for SPOC certificates is specified in Section 8.1. Profiles for CVCA, DV and IS certificates are specified in Section 8.2. An overview of the data objects contained in card-verifiable certificates is provided in Section 8.3, and the encoding of those objects is covered in Section 8.3.1 and 8.3.2.

8.1 SPOC Certificate Profile

A separate CA setup can be used to directly issue SPOC certificates with the following restrictions to the self-signed CA Certificate profile.

- CA certificate MUST conform to [RFC 5280]
- SHA-224, SHA-256, SHA-384 and SHA-512, are the only permitted hashing algorithms as defined in Doc 9303-12 (7th edition)
- `countryName` MUST be present in the Subject field

LDS2 SPOC certificates (client and server) MUST comply with the communication certificate profile defined in Doc 9303-12 (7th edition), with the following restrictions.

Issuer Field:

SPOC certificates are issued either by the CSCA or a separate CA setup specifically to issue SPOC certificates.

Subject Field:
For LDS2 SPOC certificates the subject field MUST be populated as follows:

- countryName: MUST be present. The value contains a country code that MUST follow the format of two letter country codes, specified in Doc 9303-3 (7th edition).
- commonName: MUST be present. For SPOC TLS client certificates, the value SHOULD be "SPOC TLS client". For SPOC TLS server certificates, the value SHOULD be "SPOC TLS server".
- Other attributes MAY also be included at the discretion of the issuing State or organization.

Key Usage Extensions
For SPOC certificates, the value(s) are dependent on the cipher suite used.

Subject Alternative Names Extensions
In addition to the values indicated in the communication certificate profile, SPOC TLS server certificates MUST also contain a dNSName value that is the host part of the SPOC URL.

Extended Key Usage Extensions
For SPOC client and server certificates the relevant value listed below MUST be included.

- *SPOC client certificates: OID is 2.23.136.1.1.10.1*
- *SPOC server certificates: OID is 2.23.136.1.1.10.2*

CRL Distribution Point Extensions
This extension is mandatory in SPOC client and server certificates.

8.2 CVCA, DV and Terminal Certificate Profiles

CVCA Link Certificates, DV Certificates, and Terminal Certificates are to be validated by ICs. Due to the computational restrictions of those chips, the certificates MUST be in a card verifiable format, (CV certificates).

The certificate format and profile specified below SHALL be used. Details on encoding values can be found in Doc 9303-11.

Table 9: CV Certificate Profile

Data Object	Certificate Presence
CV Certificate	m
Certificate Body	m
Certificate Profile Identifier	m
Certification Authority Reference	m
Public Key	m
Certificate Holder Reference	m
Certificate Holder Authorization Template	m
Certificate Effective Date	m
Certificate Expiration Date	m
Certificate Extensions	o
Signature	m

8.2.1 Certificate Profile Identifier

The version of the profile is indicated by the Certificate Profile Identifier. Version 1 SHALL be used and is identified by a value of 0.

8.2.2 Certificate Authority Reference & Certificate Holder Reference

Each CV Certificate MUST contain two public key references (a Certificate Holder Reference and a Certification Authority Reference).

The Certificate Authority Reference is a reference to the (external) public key of the Certification Authority (CVCA or DV) that SHALL be used to verify the signature of the certificate.

The Certificate Holder Reference is an identifier for the public key provided in the certificate that SHALL be used to reference this public key.

Note: As a consequence, the Certificate Authority Reference contained in a certificate MUST be equal to the Certificate Holder Reference in the corresponding certificate of the issuing Certification Authority.

The Certificate Holder Reference SHALL consist of the following concatenated elements: Country Code, Holder Mnemonic, and Sequence Number. Those elements MUST be chosen according to Table 10 and the following rules:

a) Country Code:
   - The Country Code SHALL be the Doc 9303-3 (7th edition) 2-letter code of the certificate holder's country.

b) Holder Mnemonic:
   - The Holder Mnemonic SHALL be assigned as unique identifier as follows:
     - The Holder Mnemonic of a CVCA SHALL be assigned by the CVCA itself;
     - The Holder Mnemonic of a DV SHALL be assigned by its domestic CVCA; and
     - The Holder Mnemonic of an IS SHALL be assigned by the supervising DV.

c) Sequence Number:
   - The Sequence Number SHALL be assigned by the certificate holder;
   - The Sequence Number MUST be numeric or alphanumeric;
     i. A numeric Sequence Number SHALL consists of the characters "0"…"9".
     ii. An alphanumeric Sequence Number SHALL consist of the characters "0"…"9" and "A"…"Z".
   - The Sequence Number MUST start with the Doc 9303-3 (7th edition) 2-letter country code of the certifying certification authority, the remaining three characters SHALL be assigned as alphanumeric Sequence Number; and
   - The Sequence Number MAY be reset if all available Sequence Numbers are exhausted.

Table 10: Certificate Holder Reference

	Encoding	**Length**
Country Code	Doc 9303-3 (7th edition) 2-letter code	2F
Holder Mnemonic	ISO/IEC 8859-1	9V
Sequence Number	ISO/IEC 8859-1	5F

F: fixed length (exact number of octets)
V: variable length (up to number of octets)

8.2.3 Public Key

This field contains the public key being certified.

CVCA self-signed certificates MUST contain domain parameters. CVCA Link certificates MAY contain domain parameters, except in the case where domain parameters have changed. In such cases, the Link certificates MUST contain the new domain parameters.

DV and Terminal certificates MUST NOT contain domain parameters. The domain parameters of DV and terminal public keys SHALL be inherited from the respective CVCA public key.

8.2.4 Certificate Holder Authorization Template

The role and authorization of the certificate holder SHALL be encoded in the Certificate Holder Authorization Template. This template is a sequence that consists of the following data objects:

a) An object identifier that specifies the terminal type and the format of the template; and
b) A discretionary data object that encodes the relative authorization, i.e. the role and authorization of the certificate holder relative to the certification authority.

Specific values are defined in Doc 9303-10.

8.2.5 Certificate Effective Date and Certificate Expiration Date

The combination of these two dates indicate the validity period of the certificate. The Certificate Effective Date MUST be the date of the certificate generation. The certificate expiration date is the date after which the certificate expires.

8.2.6 Certificate Extensions

Authorization extensions MAY be included in CVCA, DV and terminal certificates. These extensions convey authorizations additional to those in the Certificate Holder Authorization Template in the certificate.

An authorization extension is a sequence of discretionary data templates, where every discretionary data template SHALL contain a sequence of the following data objects also shown in Table 11:

a) An object identifier that specifies the content and the format of the extension; and
b) A context specific data object that contains the encoded authorization.

Table 11: Certificate Extensions

Data Object
Certificate Extensions
Discretionary Data Template
Object Identifier
Context Specific Data Object
Discretionary Data Template
Object Identifier
Context Specific Data Object
…

Note: The certificate validation procedure described in Doc 9303-11 does not take certificate extensions into account. Thus, extensions are uncritical attributes and the IC MUST NOT reject certificates due to unknown extensions.

8.2.7 Signature

The signature on the certificate SHALL be created over the encoded certificate body (i.e. including tag and length). The Certification Authority Reference SHALL identify the public key to be used to verify the signature.

8.3 Data Objects

An overview of the tags, lengths and values of the data objects used in CVCA, DV and terminal certificates is provided in Table 12.

Table 12: Overview of Data Objects (sorted by tag)

Name	***Tag***	***Len***	***Value***	***Comment***
Object Identifier	0x06	V	Object Identifier	–
Certification Authority Reference	0x42	16V	Character String	Identifies the public key of the issuing certification authority in a certificate.
Discretionary Data	0x53	V	Octet String	Contains arbitrary data.
Certificate Holder Reference	0x5F20	16V	Character String	Associates the public key contained in a certificate with an identifier.

Certificate Expiration Date	0x5F24	6F	Date	The date after which the certificate expires.
Certificate Effective Date	0x5F25	6F	Date	The date of the certificate generation.
Certificate Profile Identifier	0x5F29	1F	Unsigned Integer	Version of the certificate and certificate request format.
Signature	0x5F37	V	Octet String	Digital signature produced by an asymmetric cryptographic algorithm.
Certificate Extensions	0x65	V	Sequence	Nests certificate extensions.
Authentication	0x67	V	Sequence	Contains authentication related data objects.
Discretionary Data Template	0x73	V	Sequence	Nests arbitrary data objects.
CV Certificate	0x7F21	V	Sequence	Nests certificate body and signature.
Public Key	0x7F49	V	Sequence	Nests the public key value and the domain parameters.
Certificate Holder Authorization Template	0x7F4C	V	Sequence	Encodes the role of the certificate holder (i.e. CVCA, DV, Terminal) and assigns read/write access rights.
Certificate Body	0x7F4E	V	Sequence	Nests data objects of the certificate body.

F: fixed length (exact number of octets), V: variable length (up to number of octets)

8.3.1 Encoding of Values

The basic value types used in this specification are the following: (unsigned) integers, elliptic curve points, dates, character strings, octet strings, object identifiers, and sequences.

8.3.1.1 Unsigned Integers

All integers used in this specification are unsigned integers. An unsigned integer SHALL be converted to an octet string using the binary representation of the integer in big-endian format. The minimum number of octets SHALL be used, i.e. leading octets of value 0x00 MUST NOT be used.
Note: In contrast the ASN.1 type `INTEGER` *is always a signed integer.*

8.3.1.2 Elliptic Curve Points

The conversion of Elliptic Curve Points to octet strings is specified in [TR-03111]. The uncompressed format SHALL be used.

8.3.1.3 Dates

A date is encoded in 6 digits $d1 \cdots d6$ in the format YYMMDD using timezone GMT. It is converted to an octet string $o1 \cdots o6$ by encoding each digit dj to an octet oj as unpacked BCDs $(1 \leq j \leq 6)$.
The year YY is encoded in two digits and to be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

8.3.1.4 Character Strings

A character string $c1 \cdots cn$ is a concatenation of n characters cj with $1 \leq j \leq n$. It SHALL be converted to an octet string $o1 \cdots on$ by converting each character cj to an octet oj using the ISO/IEC 8859-1 character set.

The character codes 0x00-0x1F and 0x7F-0x9F are unassigned and MUST NOT be used. The conversion of an octet to an unassigned character SHALL result in an error.

8.3.1.5 Octet Strings

An octet string $o1 \cdots on$ is a concatenation of n octets oj with $1 \leq j \leq n$. Every octet oj consists of 8 bits.

8.3.1.6 Object Identifiers

An object identifier $i1.i2.\cdots.in$ is encoded as an ordered list of n unsigned integers ij with $1 \leq j \leq n$. It SHALL be converted to an octet string $o1 \cdots on-1$ using the following procedure:

1) The first two integers $i1$ and $i2$ are packed into a single integer i that is then converted to the octet string $o1$. The value i is calculated as follows:

i=i1·40+i2

2) The remaining integers ij are directly converted to octet strings $oj-1$ with $3 \leq j \leq n$.

More details on the encoding can be found in [DER].

Note: The unsigned integers are encoded as octet strings using the big-endian format as described in Doc 9303-11, however only bits 1-7 of each octet are used. Bit 8 (the leftmost bit) set to one is used to indicate that this octet is not the last octet in the string.

8.3.1.7 Sequences

A sequence $D1 \cdots Dn$ is an ordered list of n data objects Dj with $1 \leq j \leq n$. The sequence SHALL be converted to a concatenated list of octet strings $O1 \cdots On$ by DER encoding each data object Dj to an octet string Oj.

8.3.2 Encoding of Public Key Data Objects

A public key data object contains a sequence of an object identifier and several context specific data objects:

- The object identifier is application specific and refers not only to the public key format (i.e. the context specific data objects) but also to its usage.
- The context specific data objects are defined by the object identifier and contain the public key value and the domain parameters.

The format of public keys data objects used in this specification is described below.

8.3.2.1 RSA Public Keys

The data objects contained in an RSA public key are shown in Table 13. The order of the data objects is fixed.

Table 13: RSA Public Key

Data Object	***Abbrev***	***Tag***	***Type***	***CV Certificate***
Object Identifier		0x06	Object Identifier	m
Composite Modulus	n	0x81	Unsigned Integer	m
Public Exponent	e	0x82	Unsigned Integer	m

8.3.2.2 Elliptic Curve Public Keys

The data objects contained in an EC public key are shown in Table 14. The order of the data objects is fixed, CONDITIONAL domain parameters MUST be either all present, except the cofactor, or all absent as follows:

- Self-signed CVCA Certificates SHALL contain domain parameters.
- CVCA Link Certificates MAY contain domain parameters.
- DV and Terminal Certificates MUST NOT contain domain parameters. The domain parameters of DV and terminal public keys SHALL be inherited from the respective CVCA public key.
- Certificate Requests MUST always contain domain parameters

Table 14: EC Public Key

Data Object	***Abbrev***	***Tag***	***Type***	***CV Certificate***
Object Identifier		0x06	Object Identifier	m
Prime Modulus	p	0x81	Unsigned Integer	c

First coefficient	a	0x82	Unsigned Integer	c
Second coefficient	b	0x83	Unsigned Integer	c
Base point	G	0x84	Elliptic Curve Point	c
Order of the point	r	0x85	Unsigned Integer	c
Public point	Y	0x86	Elliptic Curve Point	m
Cofactor	f	0x87	Unsigned Integer	c

9 SPOC PROTOCOL

Single Point of Contact (SPOC) is the only interface exposed by a State for key management operations with foreign States for the LDS2 authorization PKI. The SPOC protocol is the key management protocol for operations between CVCAs and DVs in different States. Although the SPOC protocol MAY also be used for domestic communications between a CVCA and its domestic DVs and between a DV and the set of domestic terminals it manages, this is not required. Other key management protocols can be used for domestic key management.

The SPOC protocol is used to exchange keys and certificates, in order that:

- A DV can send a certification request to the foreign CVCA;
- A CVCA can send the issued certificate to the requesting DV;
- CVCAs and DVs can request the set of valid certificates from a foreign CVCA; and
- General messages can be exchanged between DVs and CVCAs.

Within a State:

- The CVCA SHALL utilize its domestic SPOC to accept incoming foreign certification requests and to send the resulting certificates or failure notifications to the requestor;
- DVs SHALL utilize their domestic SPOC to send certification requests to foreign CVCAs and to receive the resulting certificates or failure notifications;
- The SPOC MUST collect requests and responses from the domestic CVCA and DVs and forward them to the SPOC of the recipient State; and
- The SPOC MUST collect requests and responses from the SPOCs of other States and deliver them to the relevant domestic CVCA/DV.

The SPOC web-service communication SHALL use HTTPS with TLS authentication of both client and server.

Note: The SPOCs are communication hubs between the entities of the Authorization PKI which therefore should be available 24/7 and should be accessible by foreign SPOCs.

Each SPOC registers separately with all other SPOCs of interest, providing at least the following information:

- SPOC State – the State for which the SPOC provides the communication interface;
- SPOC URL – URL of WSDL describing SPOC interface and service location; and
- SPOC CA certificate – certificate(s) used to verify SPOC communication certificates.

9.1 SPOC Protocol Messages

9.1.1 Request Certificate Message

Intended Use:
The RequestCertificate message is used by a SPOC for requesting the generation of a new certificate for one of its DVs from a foreign CVCA.

Input Parameters:
callerID: (Mandatory)
This parameter contains the identifier of the request originating State. The value SHALL be the 2 letter country code according to Doc 9303-3 (7th edition) 2-letter code. The value of callerID SHALL be

verified by the recipient SPOC with the value recorded from the originating SPOC during its registration.

messageID: (Mandatory)
This parameter contains the identification of the message. It MUST identify the message uniquely within all messages from that originator. If a response message will be sent to the originator as a result of this message, the response message will contain the same messageID. Hence an incoming response message can be assigned to the correct original message. Construction and allocation of the messageID can be decided by the originator and is not verified by the receiving party.

certReq: (Mandatory)
This parameter contains the actual certificate request. It MUST be constructed according to Section 9.1.1.1. The coding MUST follow the specifications in Section 8.4.

Output Parameters:
certificateSeq: (Conditional)
This parameter will contain the result (one or more certificates) after processing this message, if the message has been processed successfully and synchronously by the receiver. It is REQUIRED if certificates have to be sent with the response. It MUST be absent if no certificates will be sent with the message.

Return Codes:
ok_cert_available: The message has been processed successfully and synchronously. The output parameter certificateSeq contains one or more certificates.

ok_reception_ack: The reception of the message is acknowledged. No further verification of the message has been done yet. The processing of the message will be done asynchronously. The result of the processing will be sent to the registered URL using the message SendCertificates.

failure_inner_signature: The verification of the inner signature of the actual certificate request failed.

failure_outer_signature: The verification of the outer signature of the actual certificate request failed.

failure_syntax: The message is syntactically not correct.

failure_request_not_accepted: The message has been processed correctly but the request has not been accepted.

failure_request_syntax: The certificate request is not correct (e.g. syntax or file format)

failure_expired: The certificate to be used to verify the outer signature of the request is expired.

failure_domain_parameters: The domain parameters contained in the request do not match the domain parameters of the CVCA certificate intended to sign the requested DV certificate.

failure_internal_error: Error other than above.

Remarks:
The body of the certificate request SHOULD contain a Certification Authority Reference (CAR) to inform the CVCA which private key the requestor expects will be used to sign the certificate. If the CAR in the request differs from the CAR in the issued certificate, the corresponding certificate of the CVCA SHALL also be provided in the response. In such a case, and if the message is processed synchronously, the CVCA certificate SHALL be part of the certificateSeq output parameter. The DV certificate SHALL be the first certificate in the sequence. CVCA certificates (root and/or link) SHALL be ordered by effective date (ascending) in the sequence.

9.1.1.1 Certificate Request Structure

Certificate requests are reduced card-verifiable certificates that may carry an additional signature. The certificate request profile specified in Table 15 SHALL be used.

Table 15: CV Certificate Request Profile

Data Object	Certificate Presence
Authentication	c
CV Certificate	m
Certificate Body	m
Certificate Profile Identifier	m
Certification Authority Reference	r
Public Key	m
Certificate Holder Reference	m
Signature	m
Certification Authority Reference	c
Signature	c

Certificate Profile Identifier
The version is version 1, identified by a value of 0.

Certification Authority Reference
The Certification Authority Reference SHOULD be used to inform the certification authority about the private key that is expected by the applicant to be used to sign the certificate. If the Certification Authority Reference contained in the request deviates from the Certification Authority Reference contained in the issued certificate (i.e. the issued certificate is signed by a private key that is not expected by the applicant), the corresponding certificate of the certification authority SHOULD also be provided to the applicant in response.

Public Key
Certificate Requests MUST always contain domain parameters.

Certificate Holder Reference
The Certificate Holder Reference is used to identify the public key contained in the request and the resulting certificate.

Signature(s)
A certificate request may have up to two signatures, an *inner signature* and an *outer signature*:

Inner Signature *(REQUIRED)*
The certificate body is self-signed, i.e. the inner signature SHALL be verifiable with the public key contained in the certificate request. The signature SHALL be created over the encoded certificate body (i.e. including tag and length).

Outer Signature *(CONDITIONAL)*

- The signature is OPTIONAL if an entity applies for the initial certificate. In this case the request MAY be additionally signed by another entity trusted by the receiving certification authority (e.g. the national CVCA may authenticate the request of a DV sent to a foreign CVCA).

- The signature is REQUIRED if an entity applies for a successive certificate. In this case the request MUST be additionally signed by the applicant using a recent key pair previously registered with the receiving certification authority.

If the outer signature is used, an authentication data object SHALL be used to nest the CV Certificate (Request), the Certification Authority Reference and the additional signature. The Certification Authority Reference SHALL identify the public key to be used to verify the additional signature. The signature SHALL be created over the concatenation of the encoded CV Certificate *and* the encoded Certification Authority Reference (i.e. both including tag and length).

9.1.2 Send Certificates Message

Intended Use:

The SendCertificates message is used by a SPOC to send the new certificate or certificate chain to the requesting SPOC. This message SHALL be generated in response to:

- RequestCertificate: upon successful asynchronous request processing after the certificate is issued
- GetCACertificates

In addition the message MUST be used when a new certificate is created (CVCA root and link) to push the certificates to registered foreign SPOC.

Input Parameters:
callerID: (Mandatory)
This parameter contains the identifier of the originating State. The value SHALL be the 2 letter country code according to Doc 9303-3 (7th edition) 2-letter code. The value of callerID SHALL be verified by the recipient SPOC with the value recorded from the originating SPOC during its registration.

messageID: (Conditional)
When the message is generated in response to a request message the parameter MUST contain the same value as the messageID parameter of the request message. When the message generation was triggered without external intervention (CVCA certificate rekey) The statusInfo value SHALL be new_cert_available_notification and the messageID parameter MAY be omitted and SHALL be ignored when present.

statusInfo: (Mandatory)
This parameter contains a status code about the result of processing the corresponding message. The following statuses are possible:

- new_cert_available_notification: The originating SPOC wants to notify that new CVCA certificate(s) are available without being requested.
- ok_cert_available: The request has been processed successfully. The input parameter certificateSeq contains one or more certificates.
- failure_inner_signature: The verification of the inner signature of the actual certificate request failed.
- failure_outer_signature: The verification of the outer signature of the actual certificate request failed.
- failure_syntax: The corresponding message is syntactically not correct.
- failure_request_not_accepted: The corresponding message has been processed correctly but the request has not been accepted.
- failure_certificate: One or more of the certificates sent is not correct (syntax or signature)
- failure_internal_error: error other than above

certificateSeq: (Conditional)
This parameter is REQUIRED if certificates have to be sent with the message. It MUST be absent if no certificates will be sent with the message. The certificates SHALL be binary TLV DER encoded as defined in Section 8.3

When the message is generated in response to a GetCACertificates message, or because there is a new certificate, the sequence SHALL contain a list of CA certificates. The list SHALL be ordered. CVCA certificates (link and/or root) SHALL be ordered by effective date in the sequence. When the sequence contains certificates with different domain parameters at least one certificate with domain parameters included for each domain paramaters variant SHALL be present. All current CA certificates SHALL be included.

When the message is generated in response to RequestCertificate message the content of the sequence is the same as described for synchronous response of RequestCertificate.

Output Parameters:
none

Return Codes:

- ok_received_correctly: The message has been received correctly.
- failure_syntax: The message is syntactically not correct.
- failure_messageID_unknown: The contained messageID cannot be matched with a message formerly sent.
- failure_internal_error: Error other than above

9.1.3 Get CA Certificates Message

Intended Use:
This message is sent by a SPOC to a foreign SPOC in order to get all valid CVCA certificates (link certificates and self-signed certificates) of that State.

Input Parameters:
callerID: (Mandatory)
This parameter contains the identifier of the originating State. The value SHALL be the 2 letter country code according to Doc 9303-3 (7th edition) 2-letter code. The value of callerID SHALL be verified by the recipient SPOC with the value recorded from the originating SPOC during its registration.

messageID: (Mandatory)
This parameter contains the identification of the message. It MUST identify the message uniquely within all messages of the originator. If a response message will be send to the originator as a result of this message, the response message will contain the same messageID. Hence an incoming response message can be assigned to the correct original message. Construction and allocation of the messageID can be decided by the originator.

Output Parameters:
certificateSeq: (Conditional)
This parameter will contain the result (one or more certificates) after processing this message, if the message has been processed successfully and synchronously by the receiver. It is REQUIRED if certificates have to be sent with the response. It MUST be absent if no certificates will be sent with the message.

Return Codes:

- ok_cert_available: The message has been processed successfully and synchronously. The output parameter certificateSeq contains one or more CA certificates.
- ok_reception_ack: The reception of the message is acknowledged. No further verification of the message has been done yet. The processing of the message will be done asynchronously. The result of the processing will be sent to the registered URL using the message SendCertificates.
- failure_syntax: The message is syntactically not correct.
- failure_internal_error: Error other than above.

Remarks:
If the message is processed successfully and accepted the CVCA MUST send all valid CVCA certificates within the response, either in the output parameter certificateSeq (synchronous processing) or in the corresponding response message SendCertificates (asynchronous processing).

9.1.4 General Message

Intended Use:
This message is sent by a SPOC to a foreign SPOC in order to send notification or other general text human readable message.

Input Parameters:
callerID: (Mandatory)
This parameter contains the identifier of the originating State. The value SHALL be the 2 letter country code according to Doc 9303-3 (7th edition) 2-letter code. The value of callerID SHALL be verified by the recipient SPOC with the value recorded from the originating SPOC during its registration, including message security features (digital signature certificate/TLS client certificate is registered for respective State).

messageID: (Mandatory)
This parameter contains the identification of the message. It MUST identify the message uniquely within all messages of the originator. If a response message will be send to the originator as a result of this message, the response message will contain the same messageID. Hence an incoming response message can be assigned to the correct original message. Construction and allocation of the messageID can be decided by the originator.

subject: (Mandatory)
This parameter contains the subject of the message. The subject SHOULD briefly describe the content of the message body. English MUST be used for subject.

body: (Mandatory)
This parameter contains the body of the message. The body SHALL be human readable plain text which is not intended for direct automated processing. English MUST be used for the body.

Return Codes:

- ok: The message has been accepted for delivery.
- failure_syntax: The message is syntactically not correct.
- failure_internal_error: Error other than above.

9.2 Web Service

The web service interface is the interface for the routine inter-SPOC wire data exchange. The interface SHALL use [SOAP] over [HTTPS] protocol. The SPOC web service interface SHALL conform to the WSDL specified in Section 9.2.3.

9.2.1 SOAP Usage

Pure [SOAP] over [HTTPS] SHALL be used to implement the Web-service interfaces. Any other SOAP extensions (e.g. WS-Addressing, WS-Security, WS-Secure Conversation, WS-Authorization, WS-Federation, WSAuthorization, WS-Policy, WS-Trust, WS-Privacy, WS-Test and other extensions of WS) SHALL NOT be used.

The intermediary SOAP node type SHALL NOT be used. Only a direct client SPOC to server SPOC configuration SHALL be used.

The SOAP fault element SHALL be used only when a transport layer processing error that is not covered by this specification occurs. Application level errors SHALL be communicated as normal SOAP responses using the error mechanism as described for each message.

It is RECOMMENDED that the web service interface is implemented in accordance to [WS-IBP] and [WSI-SSBP].

The SPOC SOAP interface MUST conform to WSDL definition as described in Section 9.2.3.

9.2.2 Security Considerations

The SPOC web service communication SHALL use a secure and authenticated channel. SOAP over HTTPS SHALL be used. TLS v1.2 SHALL be used.

The TLS client SHALL perform following verifications:

- The server certificate SHALL be fully validated according to [RFC5280] including revocation status.
- The server certificate ExtKeyUsage extension MUST be present and SHALL contain the OIDs according to Section 8.1 SPOC TLS server certificate.
- The server certificate subject country SHALL be equal to the value of callerID parameter

In case of any failure the TLS client MUST close the connection.

The TLS server SHALL perform following verifications:

- The client SHALL be fully authenticated using a certificate.
- The client certificate SHALL be fully validated according to [RFC5280] including revocation status.
- The client certificate ExtKeyUsage extension MUST be present and SHALL contain the OIDs according to Section 8.1 SPOC TLS client certificate.
- The client certificate subject country SHALL correspond to the intended one.

In case some of the verifications fail the request SHALL be rejected using HTTP 401 Unauthorized response code.

In the scope of the TLS handshake negotiation the client SHALL support all the TLS cipher suites defined in Section 4.2.3. Both the server and the client side SHALL support RSA and ECDSA based authentication. It is permissible for a server to request and also for the client to send a client certificate of a different type than the server certificate.

The use of the ECDHE_ECDSA key agreement in TLS handshake is in accordance with the additions defined in [TLSECC], [TLS1.2] and [TLSEXT]. Both the client and the server SHALL support the appropriate Elliptic curves extensions as specified in [TLSECC] specification in the scope of TLS handshake. The supported Elliptic curves and EC Point formats are defined in Section 5 of [TLSECC]. The use of the supported TLS cipher suites defined in Section 4.4 which uses Advanced Encryption Standard (AES) for encryption SHALL be in accordance with the [TLSAES] specification.

9.2.3 WSDL for SPOC Web Service Interface

The SPOC SOAP interface MUST conform to the following WSDL definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:SPOC="http://namespaces.icao.int/lds2"
  targetNamespace="http://namespaces. icao.int/lds2">

  <wsdl:types>
    <xs:schema xmlns="http://namespaces.icao.int/lds2"
      targetNamespace="http://namespaces."
      elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="certificateSequence">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="certificate" type="xs:base64Binary" minOccurs="1"
maxOccurs="unbounded"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="RequestCertificateRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="callerID" type="xs:string"/>
      <xs:element name="messageID" type="xs:string"/>
      <xs:element name="certificateRequest" type="xs:base64Binary"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="RequestCertificateResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="certificateSequence" minOccurs="0" maxOccurs="1"/>
      <xs:element name="result">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="ok_cert_available"/>
            <xs:enumeration value="ok_reception_ack"/>
            <xs:enumeration value="failure_inner_signature"/>
            <xs:enumeration value="failure_outer_signature"/>
            <xs:enumeration value="failure_syntax"/>
            <xs:enumeration value="failure_request_not_accepted"/>
            <xs:enumeration value="failure_request_syntax"/>
            <xs:enumeration value="failure_expired"/>
            <xs:enumeration value="failure_domain_parameters"/>
            <xs:enumeration value="failure_internal_error"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SendCertificatesRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="callerID" type="xs:string"/>
      <xs:element name="messageID" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="certificateSequence" minOccurs="0" maxOccurs="1"/>
      <xs:element name="statusInfo">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="new_cert_available_notification"/>
            <xs:enumeration value="ok_cert_available"/>
            <xs:enumeration value="failure_inner_signature"/>
            <xs:enumeration value="failure_outer_signature"/>
            <xs:enumeration value="failure_syntax"/>
            <xs:enumeration value="failure_request_not_accepted"/>
            <xs:enumeration value="failure_certificate"/>
            <xs:enumeration value="failure_internal_error"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SendCertificatesResponse">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="result">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="ok_received_correctly"/>
              <xs:enumeration value="failure_syntax"/>
              <xs:enumeration value="failure_messageID_unknown"/>
              <xs:enumeration value="failure_internal_error"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetCACertificatesRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="callerID" type="xs:string"/>
        <xs:element name="messageID" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetCACertificatesResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="certificateSequence" minOccurs="0" maxOccurs="1"/>
        <xs:element name="result">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="ok_cert_available"/>
              <xs:enumeration value="ok_reception_ack"/>
              <xs:enumeration value="failure_syntax"/>
              <xs:enumeration value="failure_internal_error"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GeneralMessageRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="callerID" type="xs:string"/>
        <xs:element name="messageID" type="xs:string"/>
        <xs:element name="subject" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GeneralMessageResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="result">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="ok"/>
              <xs:enumeration value="failure_syntax"/>
              <xs:enumeration value="failure_internal_error"/>
            </xs:restriction>
          </xs:simpleType>
```

```
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
 </xs:schema>
</wsdl:types>

<wsdl:message name="RequestCertificateRequest">
 <wsdl:part name="RequestCertificateRequest" element="SPOC:RequestCertificateRequest"/>
</wsdl:message>
<wsdl:message name="RequestCertificateResponse">
 <wsdl:part name="RequestCertificateResponse" element="SPOC:RequestCertificateResponse"/>
</wsdl:message>

<wsdl:message name="SendCertificatesRequest">
 <wsdl:part name="SendCertificatesRequest" element="SPOC:SendCertificatesRequest"/>
</wsdl:message>
<wsdl:message name="SendCertificatesResponse">
 <wsdl:part name="SendCertificatesResponse" element="SPOC:SendCertificatesResponse"/>
</wsdl:message>

<wsdl:message name="GetCACertificatesRequest">
 <wsdl:part name="GetCACertificatesRequest" element="SPOC:GetCACertificatesRequest"/>
</wsdl:message>
<wsdl:message name="GetCACertificatesResponse">
 <wsdl:part name="GetCACertificatesResponse" element="SPOC:GetCACertificatesResponse"/>
</wsdl:message>

<wsdl:message name="GeneralMessageRequest">
 <wsdl:part name="GeneralMessageRequest" element="SPOC:GeneralMessageRequest"/>
</wsdl:message>
<wsdl:message name="GeneralMessageResponse">
 <wsdl:part name="GeneralMessageResponse" element="SPOC:GeneralMessageResponse"/>
</wsdl:message>

<wsdl:portType name="SPOCPortType">
 <wsdl:operation name="RequestCertificate">
  <wsdl:input message="SPOC:RequestCertificateRequest"/>
  <wsdl:output message="SPOC:RequestCertificateResponse"/>
 </wsdl:operation>
 <wsdl:operation name="SendCertificates">
  <wsdl:input message="SPOC:SendCertificatesRequest"/>
  <wsdl:output message="SPOC:SendCertificatesResponse"/>
 </wsdl:operation>
 <wsdl:operation name="GetCACertificates">
  <wsdl:input message="SPOC:GetCACertificatesRequest"/>
  <wsdl:output message="SPOC:GetCACertificatesResponse"/>
 </wsdl:operation>
 <wsdl:operation name="GeneralMessage">
  <wsdl:input message="SPOC:GeneralMessageRequest"/>
  <wsdl:output message="SPOC:GeneralMessageResponse"/>
 </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="SPOCSOAPBinding" type="SPOC:SPOCPortType">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="RequestCertificate">
  <soap:operation soapAction="RequestCertificate"/>
  <wsdl:input>
   <soap:body parts="RequestCertificateRequest" use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output>
        <soap:body parts="RequestCertificateResponse" use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SendCertificates">
      <soap:operation soapAction="SendCertificates"/>
      <wsdl:input>
        <soap:body parts="SendCertificatesRequest" use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body parts="SendCertificatesResponse" use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetCACertificates">
      <soap:operation soapAction="GetCACertificates"/>
      <wsdl:input>
        <soap:body parts="GetCACertificatesRequest" use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body parts="GetCACertificatesResponse" use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GeneralMessage">
      <soap:operation soapAction="GeneralMessage"/>
      <wsdl:input>
        <soap:body parts="GeneralMessageRequest" use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body parts="GeneralMessageResponse" use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="SPOC">
    <wsdl:port name="SPOCPort" binding="SPOC:SPOCSOAPBinding">
      <soap:address location="http://spoc-server/SPOC"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

REFERENCES

[RFC-RSA] Jonsson, Jakob and Kaliski, Burt RFC 3447, Public-key cryptography standards (PKCS)#1: RSA cryptography specifications version 2.1, 2003

[PKCS#1] RSA Laboratories RSA Laboratories Technical Note, PKCS#1 v2.1: RSA cryptography standard, 2002

[RFC 5280] RFC 5280, D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May, 2008

[TR-03111] BSI TR-03111, Elliptic Curve Cryptography (ECC) Version 2.0, 2012

[TLSAES] Chown, P., „Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)“, RFC 3268, June 2002

[TLSECC] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, „Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)“, RFC 4492, May 2006

[TLS1.2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008

[TLSEXT] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, „Transport Layer Security (TLS) Extensions“, RFC 4366, April 2006

[SOAP] SOAP Version 1.2 Part 1: Messaging framework (Second Edition), W3C Recommendation 27 April 2007

[HTTPS] E. Rescorla., „HTTP Over TLS.“, RFC 2818, May 2000

[WSI-BP] WS-I Basic Profile available at http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[WSI-SSBP] WS-I Basic Binding available at http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

APPENDIX A

The following example illustrates the interactions between the different components of the LDS2 Signature PKI and the LDS2 Authorization PKI.

To illustrate the interactions and preliminaries required for a typical business scenario, consider the scenario where the country of Dystopia wants to write travel stamps to passports of citizens of the country of Utopia. Later, the country of Atlantis wants to read travels stamps written by Dystopia on Utopia's passports.

Preliminaries:

- Utopia has installed an LDS 2 Travel Stamp application on their passports.
- Both Dystopia and Utopia have set up their LDS2 Authorization PKI.
- Dystopia has set up their LDS1 Signing PKI to issue LDS2 Signer Certificates.
- CVCA certificates and SPOC client and server certificates were exchanged in a trusted manner between Dystopia and Utopia at some point in time (subsequently, new CVCA and SPOC certificates can be exchanged directly via the SPOC).
- CVCA certificates and SPOC client and server certificates were exchanged in a trusted manner between Dystopia and Atlantis at some point in time (subsequently, new CVCA and SPOC certificates can be exchanged directly via the SPOC). If the LDS2 travel stamp application is open for reading, i.e. any country can read LDS2 travel stamps (permission is only needed for writing), this step can be omitted.
- CSCA certificates have been exchanged in a trusted manner between Dystopia and Atlantis at some point in time.

Recurring process in order to enable Dystopia to electronically stamp Utopia's eMRTDs:

- Dystopia requests a DV certificate from Utopia.
- Dystopia's SPOC uses its SPOC client certificate and Utopia's SPOC server certificate to initiate a SPOC connection. Then a request is generated by a dystopian DV, and sent from SPOC to SPOC. Upon request, Utopia generates a foreign DV certificate with read/write access for Dystopia, and the certificate is delivered back via SPOC to SPOC.

- Upon receiving the DV certificate from its SPOC, the DV of Dystopia generates Terminal Certificates for the terminals of its borders. Connecting to the passport, the IC on the utopian passports verifies the terminal certificate of Dystopia with the DV certificate of Dystopia, and the DV certificate of Dystopia with the CVCA certificate of Utopia. The IC then grants read/write access for the dystopian terminal to the LDS2 Travel Stamp application.

Process to electronically stamp an eMRTD:

- Dystopia creates an electronic travel stamp, and signs it with the private key corresponding to the public key stored in an LDS2 (Travel Stamp) Signer certificate of the LDS2 Signing PKI of Dystopia. The LDS2 Signer certificate is stored on the contactless IC of the utopian passport.

Upon encountering the utopian passport at the border of Atlantis:

- If reading travel stamps from utopian passports requires a terminal certificate with read-access, a certificate request from Atlantis is sent via SPOC-to-SPOC to Utopia. Upon request, Utopia generates a foreign DV certificate with read-access for Atlantis and sends this certificate to Atlantis via SPOC-to-SPOC. Using that DV certificate, Atlantis generates terminal certificates with read-access for utopian passports for Atlantis' terminals. I travel stamps in utopian passports can be read by any terminal, this step can be omitted.

- To verify a travel stamp of the passport written by Dystopia, Atlantis uses the LDS1 signing PKI of Dystopia: The dystopian LDS2 Signer certificate stored in the passport is used to verify the travel stamp. Then the chain is build up, i.e. the Dystopia LDS2 Signer certificate is verified with the Dystopia CSCA certificate received preliminarily.