ICAO

Doc 9303

Machine Readable Travel Documents

Eighth Edition, 2021

Part 12: Public Key Infrastructure for MRTDs

Approved by and published under the authority of the Secretary General

INTERNATIONAL CIVIL AVIATION ORGANIZATION

ICAO

Doc 9303

Machine Readable Travel Documents

Eighth Edition, 2021

Part 12: Public Key Infrastructure for MRTDs

Approved by and published under the authority of the Secretary General

INTERNATIONAL CIVIL AVIATION ORGANIZATION

Published in separate English, Arabic, Chinese, French, Russian
and Spanish editions by the
INTERNATIONAL CIVIL AVIATION ORGANIZATION
999 Robert-Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7

Downloads and additional information are available at www.icao.int/security/mrtd

Doc 9303, *Machine Readable Travel Documents*
Part 12 — *Public Key Infrastructure for MRTDs*
Order No.: 9303P12
ISBN 978-92-9265-422-1 (print version)
ISBN 978-92-9275-422-8 (electronic version)

AMENDMENTS

Amendments are announced in the supplements to the *Products and Services Catalogue;* the Catalogue and its supplements are available on the ICAO website at www.icao.int. The space below is provided to keep a record of such amendments.

RECORD OF AMENDMENTS AND CORRIGENDA

AMENDMENTS		
No.	Date	Entered by
1	14/6/24	ICAO

CORRIGENDA		
No.	Date	Entered by

TABLE OF CONTENTS

Page

1. SCOPE

Part 12 of Doc 9303 defines the Public Key Infrastructure (PKI) for the eMRTD application. Requirements for issuing States or organizations are specified, including operation of a Certification Authority (CA) that issues certificates and Certificate Revocation Lists (CRLs). Requirements for receiving States and their Inspection Systems validating those certificates and CRLs are also specified.

The Eighth Edition of Doc 9303 incorporates the specifications for Visible Digital Seals (known as VDS) and for the optional Travel Records, Visa Records and Additional Biometric Applications (known as LDS2) as an extension of the mandatory eMRTD application (known as LDS1).

Doc 9303-12 shall be read in conjunction with:

- Doc 9303-10 — *Logical Data Structure (LDS) for Storage of Biometrics and Other Data in the Contactless Integrated Circuit (IC)*;
- Doc 9303-11 — *Security Mechanisms for MRTDs*; and
- Doc 9303-13 — *Visible Digital Seals.*

2. OVERVIEW OF THE PUBLIC KEY INFRASTRUCTURE

The eMRTD Public Key Infrastructure (PKI) enables the creation and subsequent verification of digital signatures on eMRTD objects, including the Document Security Object (SO_D) to ensure the signed data is authentic and has not been modified. Revocation of a certificate, failure of the certification path validation procedure or failure of digital signature verification does not on its own cause an eMRTD to be considered invalid. Such a failure means that the electronic verification of the integrity and authenticity of the LDS data has failed and other non-electronic mechanisms could then be used to make that determination as part of the overall inspection of the eMRTD.

The eMRTD PKI is much simpler than more generic multi-application PKIs such as the Internet PKI defined in [RFC 5280]. In the eMRTD PKI, each issuing State/Authority establishes a single Certification Authority (CA) that issues all certificates directly to end-entities, including Document Signers. These CAs are referred to as Country Signing Certification Authorities (CSCAs). There are no other CAs in the infrastructure. Receiving States establish trust directly in the keys/certificates of each issuing State or organization's CSCA.

The eMRTD PKI is based on generic PKI standards including [X.509] and [RFC 5280]. Those base PKI standards define a large set of optional features and complex trust relationships among CAs that are not relevant to the eMRTD application. A profile of those standards, tailored to the eMRTD application, is specified in this part of Doc 9303. Some of the unique aspects of the eMRTD application include:

- there is precisely one CSCA per issuing State;
- certification paths include precisely one certificate (e.g. Document Signer);
- signature verification must be possible 5-10 years after creation;
- CSCA name change is supported; and
- CSCA Link certificates are not processed as intermediate certificates in a certification path.

For the most part, the eMRTD PKI infrastructure is compliant with [RFC 5280]. However, the fact that CSCAs can undergo a name change imposes unique requirements on the eMRTD PKI that are incompatible with some of the CRL validation procedures defined in [RFC 5280]. These differences have been kept to a minimum and are clearly identified.

For VDS and LDS2, the Digital Signature PKI, which provides integrity and authenticity of the data objects, is an extension of the LDS1 PKI. The Signers for VDS and LDS2 are issued by the same CSCA which issues Signers for LDS1. The changes to the Certificate Profiles for these new applications are specified in this document. Taken together, this infrastructure is referred to as the **eMRTD PKI**.

The Digital Signature PKI consists of the following entities:

- Country Signing CA (CSCA);
- Document Signer Certificates (DSC) which is used to sign the Document Security Objects (SO_D);
- LDS2 Signer Certificates, which consists of the following:
  - LDS2-TS Signer – signs LDS2 Travel Stamps;
  - LDS2-V Signer – signs LDS2 Electronic Visas; and
  - LDS2-B Signer – signs LDS2 Additional Biometrics;
- Bar Code Signer Certificates (BCSC), for which the following two specific types are defined in this document:
  - Visa Signer Certificates (VSC); and
  - Emergency Travel Document Signer Certificates (ESC);
- Master List Signer Certificates (MSC) used to sign Master Lists;
- Deviation List Signer Certificates (DLSC) used to sign Deviation Lists; and
- Certificate Revocation List (CRL).

All the different certificate types are signed by the same CSCA. The CSCA also signs the CRL, which contains any revoked certificate irrespective of the type of certificate. All the certificates issued under the CSCA are collectively referred to as **Signer Certificates**.

For LDS2 applications, a separate **Authorization PKI** is defined. 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 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; and
- 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.

This Part 12 of Doc 9303 specifies the eMRTD PKI profile, the Authorization PKI profile and corresponding objects including:

- roles and responsibilities of entities in the infrastructure;
- cryptographic algorithms and key management;
- certificate and CRL content;
- certificate and CRL distribution mechanisms; and
- certification path validation.

3. ROLES AND RESPONSIBILITIES

This section details the entities and the roles and responsibilities of both the eMRTD PKI and the Authorization PKI.

3.1 eMRTD PKI

The authenticity and integrity of data stored on eMRTDs is protected by Passive Authentication. This security mechanism is based on digital signatures and consists of the following PKI entities for eMRTD PKI:

- **Country Signing CA (CSCA):** Each issuing State/Authority establishes a single CSCA as its national trust point in the context of eMRTDs. The CSCA issues public key certificates for one or more (national) Document Signers and optionally for other end-entities such as Master List Signers and Deviation List Signers. The CSCA also issues periodic Certificate Revocation Lists (CRL) indicating whether any of the issued certificates have been revoked.
- **Document Signers (DS):** A Document Signer digitally signs data to be stored on eMRTDs; this signature is stored on the eMRTD in a Document Security Object.
- **LDS2 Signers**: An LDS2 Signer digitally signs LDS2 data objects of one or more types.

- **Bar Code Signer (BCS):** A Bar Code Signer digitally signs the data (header and message) encoded in the bar code. The signature is also stored in the bar code. This document specifies two use cases for the use of Bar Code Signer, viz. Visa and Emergency Travel Documents.

- **Inspection Systems (IS):** An Inspection System verifies the digital signature, including certification path validation to verify the authenticity and integrity of the electronic data stored on the eMRTD as part of Passive Authentication.

- **Master List Signers:** A Master List Signer is an optional entity that digitally signs a list of CSCA certificates (domestic and foreign) in support of the bilateral distribution mechanism for CSCA certificates.

- **Deviation List Signers:** Deviation List Signers are used to sign Deviation Lists. Deviation lists are defined in Doc 9303-3.

The secure facilities to generate key pairs SHALL be under the control of the issuing State or organization. Each key pair includes a “private” key and a “public” key. The private keys and associated systems or facilities SHALL be well protected from any outside or unauthorized access through inherent design and hardware security facilities.

While the CSCA certificate remains relatively static, a large number of Document Signer certificates will be created over time.

The CSCA of each issuing State or organization acts as the trust point for the receiving State. The issuing State or organization distributes its own CSCA public key to receiving States in the form of a certificate. The receiving State establishes that this certificate (and certified key) are “trusted” through out-of-band means, and stores a “Trust Anchor” for that trusted key/certificate. These CSCA certificates SHALL be self-signed certificates issued directly by the CSCA. CSCA certificates MUST NOT be subordinate or cross certificates in a larger PKI infrastructure. CSCA self-issued link certificates may also be issued to help the receiving State in establishing trust in a new CSCA key/certificate following a key-rollover.

Note.— In some States there is a requirement that a centralized Controller of Certification Authority (CCA) be the supreme authority to publish self-signed certificates for all applications. In these cases, a possible solution is for the CSCA to create a self-signed certificate (satisfying the ICAO Doc 9303 requirements) and have that certificate countersigned by the CCA (satisfying the State’s own CCA requirement). However, these countersigned certificates are not part of the eMRTD PKI and would not be distributed to receiving States.

3.1.1 Country Signing Certification Authority

It is RECOMMENDED that CSCA key pairs (KPu_{CSCA}, KPr_{CSCA}) be generated and stored in a highly protected, off-line CA infrastructure.

The CSCA private key (KPr_{CSCA}) is used to sign Document Signer certificates (C_{DS}), other certificates and CRLs.

Country Signing Certification Authority certificates (C_{CSCA}) are used to validate Document Signer certificates, Master List Signer certificates, Deviation List Signer certificates, CRLs and other certificates issued by the CSCA.

All certificates and CRLs MUST comply with the profiles specified in Section 7 and MUST be distributed using the distribution mechanisms as specified in Section 5.

For PKD participants, each CSCA certificate (C_{CSCA}) MUST also be forwarded by the certificate issuer to the PKD (for the purpose of validation of Document Signer certificates (C_{DS})).
CRLs MUST be issued on a periodic basis as specified in Section 4.

3.1.2 Document Signers

It is RECOMMENDED that Document Signer key pairs (KPu_{DS}, KPr_{DS}) be generated and stored in a highly protected infrastructure.

The Document Signer private key (KPr_{DS}) is used to sign Document Security Objects (SO_D).

Document Signer certificates (C_{DS}) are used to validate Document Security Objects (SO_D).

Each Document Signer certificate (C_{DS}) MUST comply with the certificate profile defined in Section 7 and MUST be stored in the contactless IC of each eMRTD that was signed with the corresponding DS private key (see Doc 9303-10 for details). This ensures that the receiving State has access to the Document Signer certificate relevant to each eMRTD.

Document Signer certificates of PKD participants SHOULD also be forwarded by the certificate issuer to ICAO for publication in the ICAO Public Key Directory (PKD).

3.1.3 LDS2 Signers

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; and
- 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 or all of these roles.

If further differentiation is required, such as the location at which 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, it can be included in a proprietary field within the respective LDS2 data object itself.

3.1.4 Bar Code Signers

It is RECOMMENDED that Bar Code Signer key pairs (KPu_{BCS}, KPr_{BCS}) be generated and stored in a highly protected infrastructure.

The Bar Code Signer private key (KPr_{BCS}) is used to sign the data (header and message) encoded in the bar code. The signature is also stored in the bar code.

Bar Code Signer certificates (C_{BCS}) are used to validate the data (header and message) encoded in the bar code.

Each Bar Code Signer certificate (C_{BCS}) MUST comply with the certificate profile defined in Section 7. The Bar Code Signer

Certificates are not contained in the digital seal itself. Hence, a country that issues documents protected with digital seals MUST publish all its Bar Code Signer Certificates. The primary distribution channel for Bar Code Signer Certificates is PKD/bilateral. Other mechanisms, e.g. publication on a website, are secondary channels.

Bar Code Signer certificates of PKD participants SHOULD also be forwarded by the certificate issuer to ICAO for publication in the ICAO Public Key Directory (PKD).

The Visa Signer (VS) and Emergency Travel Document Signer are special cases of the Bar Code Signer.

3.1.5 Inspection System

Inspection Systems perform Passive Authentication to ensure the integrity and authenticity of the data stored on the eMRTD contactless IC. As part of that process, Inspection Systems MUST perform certification path validation as indicated in Section 6.

3.1.6 Master List Signer

The Master List Signer private key is used to sign CSCA Master Lists.

Master List Signer certificates are used to validate CSCA Master Lists.

3.1.7 Deviation List Signer

The Deviation List Signer private key is used to sign Deviation Lists.

Deviation List Signer certificates are used to validate Deviation Lists.

3.2 Authorization PKI

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.

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.

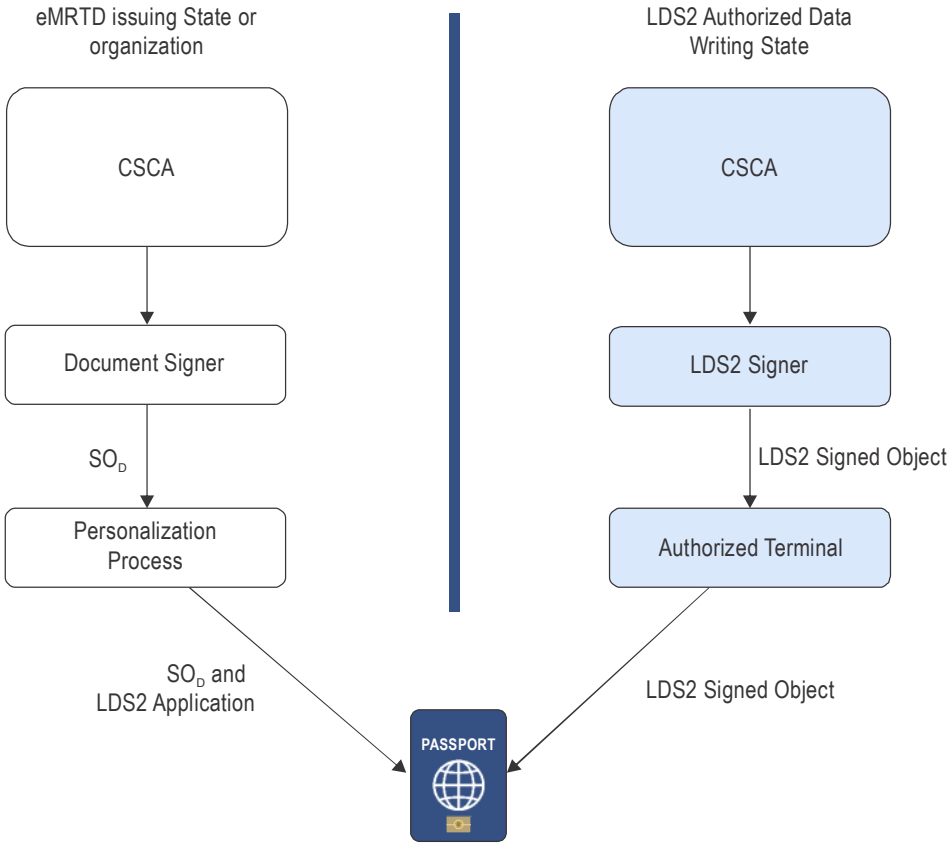

Figure 1. LDS2 Trust Model and Writing Architecture

3.2.1 Country Verifying Certificate Authority

Each issuing State or organization that allows LDS2 data to be added to its eMRTDs MUST set up a single Country Verifying Certificate Authority (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 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.

3.2.2 Document Verifier

A Document Verifier (DV) is a CA that, as part of an organizational unit, 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.

3.2.3 Terminal/Inspection System

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.

3.2.4 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 8.

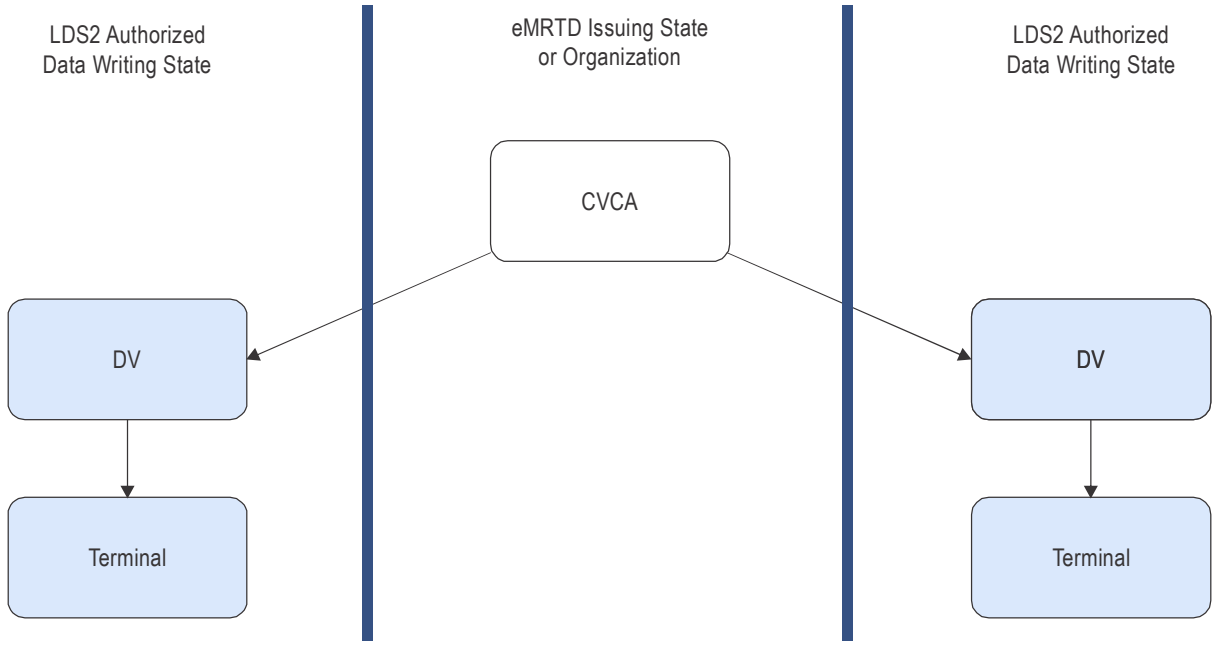

Figure 2. Authorization PKI Trust Model

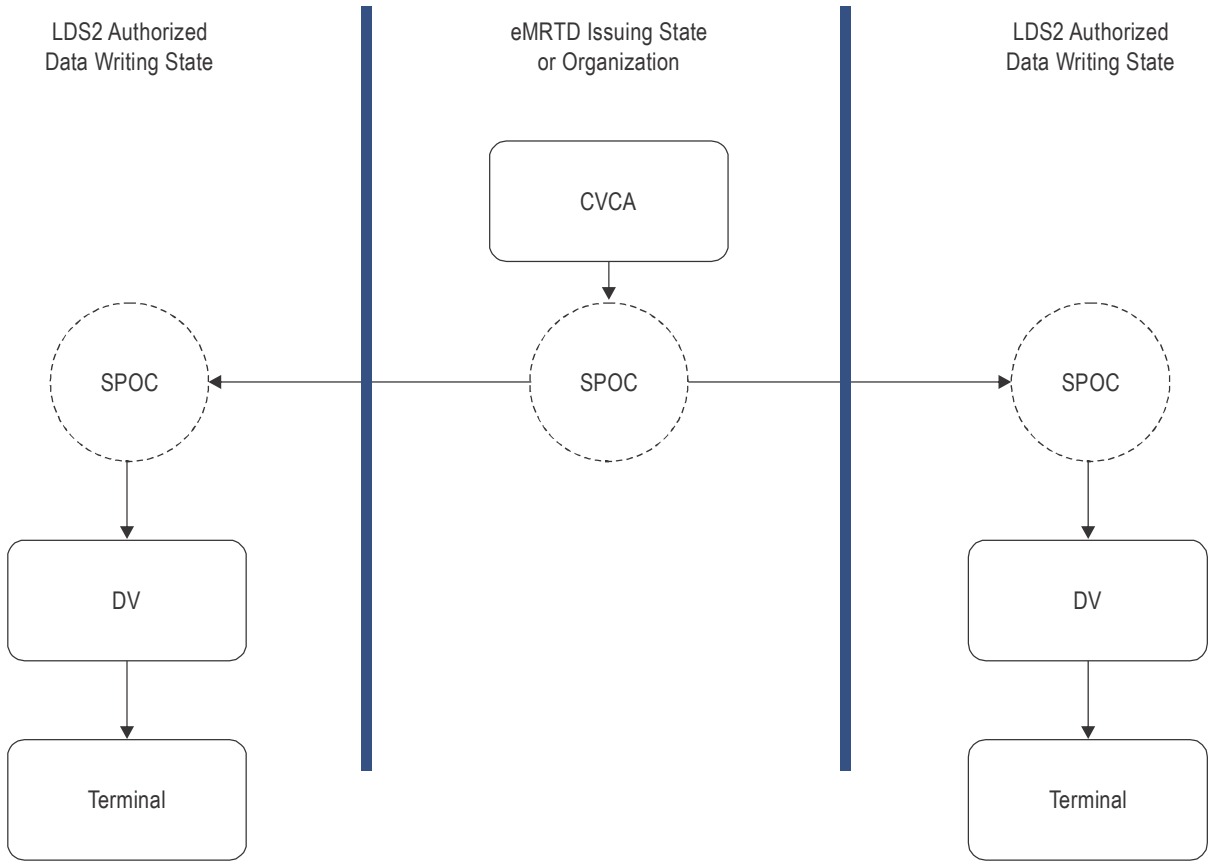

Figure 3. SPOC Role

4. KEY MANAGEMENT

Key Management is defined for the two Public Key Infrastructures separately.

4.1 eMRTD PKI

Issuing States or organizations SHALL have at least two key pair types:

- Country Signing CA key pair; and
- Document Signer key pair.

Issuing States or organizations MAY have additional key pair types:

- Master List Signer key pair;
- Deviation List Signer key pair;

- LDS2 Signer key pair;
- SPOC client key pair;
- SPOC server key pair; and
- Visa Signer key pair/Emergency Travel Document Signer key pair (both are types of Bar Code Signers).

The Country Signing CA, Signer Certificate and SPOC certificate public keys are issued using [X.509] certificates. The public keys contained in CSCA certificates are used to verify the CSCA signature on issued Signer Certificates, SPOC Certificates, CSCA and on issued CRLs.

For Master List Signer, Deviation List Signer and Communications keys and certificates, the private key lifetime and the certificate validity period are left to the discretion of the issuing State or organization.

Both the CSCA certificates and Document Signer certificates are associated with a private key usage and a public key validity period as outlined in Table 1.

Table 1. Key Usage and Validity

	Use of Private Key	***Public Key Validity (assuming 10-year valid passports)***
Country Signing CA	3-5 years	13-15 years
Document Signer	Up to 3 months[1]	approx. 10 years
LDS2-TS Signer	1-2 years	10 years + 3 months
LDS2-V Signer	1-2 years	10 years + 3 months
LDS2-B Signer	1-2 years	10 years + 3 months
SPOC Client	Not Specified	6-18 months
SPOC Server	Not Specified	6-18 months
Visa Bar Code Signer	1-2 years	Private Key Usage Time + Validity of Visa
Emergency Travel Document Bar Code Signer	1 year + 2 months (the 2 months are meant for smooth roll-over)	Private Key Usage Time + ETD validity timeframe
Master List Signer	Discretion of issuing State or organization	Discretion of issuing State or organization
Deviation List Signer	Discretion of issuing State or organization	Discretion of issuing State or organization
Communication	Discretion of issuing State or organization	Discretion of issuing State or organization

1. Note the corresponding `privateKeyUsage` extension in DS certificate might be slightly longer to allow for overlap or production requirements.

4.1.1 Document Signer Keys and Certificates

The usage period of a Document Signer private key is much shorter than the validity period of the DS certificate for the corresponding public key.

4.1.1.1 Document Signer Public Key validity

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

- the length of time the corresponding private key will be used to issue eMRTDs, with;
- the longest validity period of any eMRTD issued under that key[2].

The Document Signer certificate (C_{DS}) SHALL be valid for this total period to enable the authenticity of eMRTDs to be verified. However the corresponding private key SHOULD only be used to issue documents for a limited period; once the last document it was used to issue has expired, the public key is no longer required.

4.1.1.2 Document Signer Private Key issuing period

When deploying their systems, issuing States or organizations may wish to take into account the number of documents that will be signed by any one individual Document Signer private key.

An issuing State or organization may deploy one or more Document Signers, each with its own unique key pair, that are active at any given time.

In order to minimize business continuity costs in the event of a Document Signer certificate being revoked, an issuing State or organization that issues a large number of eMRTDs per day may wish to:

- use a very short private key usage period; and/or
- deploy several concurrent Document Signers that are active at the same time, each with its own unique private key and public key certificate.

An issuing State or organization that issues a small number of eMRTDs per day may choose to deploy a single Document Signer and may also be comfortable with a slightly longer private key usage period.

Regardless of the number of eMRTDs issued per day, or number of Document Signers active at the same time, it is RECOMMENDED that the maximum period any Document Signer private key is used to sign eMRTDs be three months.

Once the last document signed with a given private key has been produced, it is RECOMMENDED that issuing States or organizations erase the private key in an auditable and accountable manner.

2. Some issuing States or organizations may issue eMRTDs before they become valid, for instance on a change of name upon marriage. In these situations, the “longest validity period of any eMRTD” includes the actual validity of the eMRTD (e.g. 10 years) plus the maximum time between when the eMRTD is issued and the time it becomes valid.

4.1.2 LDS2 Signer Keys and Certificates

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).

4.1.2.1 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 to 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.

4.1.3 Bar Code Signer Keys and Certificates

A Bar Code Signer is a specific type of signature server used to sign a unique Document Type Category, e.g. a visa, Emergency Travel Document, etc. To follow the best practices in the field, it is RECOMMENDED that only a limited number of signing keys (a lower one-digit number) is used in parallel to create signatures for digital seals, unless operational requirements make a larger number of keys absolutely necessary. To ensure availability of the Bar Code Signer in case of a security incident related to the signing keys, it is RECOMMENDED to have measures in place to ensure business continuity (e.g. preparation of backup keys, backup site, etc.).

In order to facilitate the handling of the corresponding certificates (see Section 5), the number of published signature validation keys MUST be limited to five signature keys per year.

4.1.3.1 Bar Code Signer Public Key Validity

This section applies to all Bar Code Signers, including Visa Signer and Emergency Travel Document Signer.

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

- the length of time the corresponding private key will be used to issue a Visa or ETD, with;
- the longest validity period of any document issued under that key[3].

3. Some issuing States or organizations may issue eMRTDs before they become valid, for instance on a change of name upon marriage. In these situations, the “longest validity period of any eMRTD” includes the actual validity of the eMRTD (e.g. 10 years) plus the maximum time between when the eMRTD is issued and the time it becomes valid.

The Bar Code Signer certificate SHALL be valid for this total period to enable the authenticity of document to be verified. However, the corresponding private key SHOULD only be used to issue documents for a limited period; once the last document it was used to issue has expired, the public key is no longer required.

Private Key Usage Time:	As per document profile
Certificate Validity:	Private Key Usage Time + document Validity Timeframe

Example

Note.— The actual validity periods used for the calculation in this example do not imply any recommendations.

Suppose that documents with a validity period of 5 years are issued, and the private key usage time of the BarCode Signer Certificate is 1 year. Then, the validity of the Bar Code Signer Certificate is 1 + 5 = 6 years. If the usage time of the private key of the CSCA Certificate is 3 years, then the validity of the CSCA Certificate is 3 + 6 = 9 years.

4.1.4 CSCA Keys and Certificates

The usage period of a CSCA private key is much shorter than the validity period of the CSCA certificate for the corresponding public key.

4.1.4.1 Country Signing CA Public Key validity

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

- the length of time the corresponding CSCA private key will be used to sign any certificate below the CSCA; and,
- the maximum key lifetime of any certificate issued below the CSCA.

4.1.4.2 Country Signing CA Private Key issuing period

The usage period for the CSCA private key to sign certificates and CRLs is a delicate balance among the following factors:

- In the unlikely event of an issuing State or organization Country Signing Private CA Key being compromised, then the validity of all eMRTDs issued using Document Signer Keys whose certificates were signed by the compromised CSCA private key is called into doubt. Consequently issuing States or organizations MAY wish to keep the issuing period quite short;
- Keeping the issuing period very short, however, leads to having a very large number of CSCA public keys valid at any one time. This can lead to more complex certificate management within the border processing systems.

It is therefore RECOMMENDED that an issuing State or organization’s CSCA key pair be replaced every three to five years.

4.1.4.3 Country Signing CA Re-key

CSCA keys provide the trust points in the whole system and without these the system would collapse. Therefore issuing States or organizations SHOULD plan the replacement of their CSCA key pair carefully. Once the issuance period for the initial CSCA private signing key has elapsed, an issuing State or organization will always have at least two CSCA certificates (C_{CSCA}) valid at any one time.

Issuing States or organizations MUST notify receiving States that a CSCA key rollover is planned. This notification MUST be provided 90 days in advance of the key rollover. Once the key rollover has occurred the new CSCA certificate (certifying the new CSCA public key) is distributed to receiving States.

If the CSCA certificate is a new self-signed certificate, authentication of that certificate should be done using an out-of-band method.

When a CSCA key rollover occurs a certificate MUST be issued that links the new key to the old key to provide a secure transition for relying parties. Generally this is achieved through the issuance of a self-issued-certificate where the issuer and subject fields are identical but the key used to verify the signature represents the old key pair and the certified public key represents the new key pair. These CSCA Link certificates need not be verified using an out-of-band method as the signature on the CSCA Link certificate is verified using an already trusted public key for that CSCA. Master Lists can also be used to distribute CSCA Link and CSCA self-signed root certificates.

Issuing States or organizations should refrain from using their new CSCA private key for the first two days after the CSCA key rollover, to ensure the corresponding new CSCA public key certificate has been distributed successfully.

Issuing States or organizations MUST use the newest CSCA private key for signing all certificates, and for signing CRLs.

4.1.5 Certificate Revocation

Issuing States or organizations may need to revoke certificates in case of an incident (like a key compromise).

All CSCAs MUST produce periodic revocation information in the form of a Certificate Revocation List (CRL).

CSCAs MUST issue at least one CRL every 90 days, even if no certificates have been revoked since the previous CRL was issued. CRLs MAY be issued more frequently than every 90 days but not more frequently than every 48 hours.

If a certificate is revoked, a CRL indicating that revocation MUST be distributed within 48 hours.

Only certificates can be revoked, not Document Security Objects. The use of CRLs is limited to notifications of revoked certificates that had been issued by the CSCA that issued the CRL (including revocation notices for CSCA certificates, DS certificates, Master List Signer certificates, Deviation List Signer certificates and any other certificate types issued by that CA).

Partitioned CRLs are not used in the eMRTD application. All certificates revoked by a CSCA, including DS certificates, CSCA certificates, Master List Signer certificates and Deviation List Signer certificates are listed on the same CRL. Although the CRL is always signed with the newest (current) CSCA private signing key, the CRL includes revocation notices for certificates signed with that same private key as well as certificates signed with earlier CSCA private signing keys.

4.1.5.1 Revocation of CSCA Certificates

Revocation of a CSCA certificate is both extreme and difficult. Upon informing a receiving State that a CSCA certificate has been revoked, all other certificates signed using the corresponding CSCA private key are effectively revoked.

Where a CSCA Link certificate has been signed using an old CSCA private key to certify a new CSCA public key (see “Country Signing CA Re-key” in 4.1.4.3), revoking the old CSCA certificate SHALL also revoke the new CSCA certificate.

If a CSCA certificate needs to be revoked, the CSCA may issue a CRL signed with the private key that corresponds to the public key being revoked, as this is the only key users of the CRL will be able to verify at that time. The CSCA public key should be considered valid only for the purpose of verifying that CRL signature. Once a CRL user has verified the CRL signature, the CSCA private signing key is considered compromised and the certificate revoked for all future verifications.

To issue new documents, the issuing State or organization MUST revert to bootstrapping its authentication process from the beginning, by issuing a new CSCA Root certificate, distributing that certificate to receiving States, and supporting out-of-band confirmation that the certificate received by each receiving State is in fact the current authentic CSCA certificate.

4.1.5.2 Revocation of other Certificates

When an issuing State or organization wishes to revoke a Signer certificate issued under the CSCA, it does not need to wait until the `nextUpdate` period in the current CRL is due to issue a new CRL. It is RECOMMENDED that a new CRL be issued within a 48-hour period of revocation notification.

4.1.6 Cryptographic Algorithms

An issuing State or organization MAY support different algorithm for use in their CSCA and Signing Certificate keys. For example, the CSCA may have been issued using RSA, but Signer Certificates could be Elliptic Curve DSA (ECDSA) and vice versa.

Issuing States or organizations SHALL choose appropriate key lengths offering protection against attacks. Suitable cryptographic catalogues SHOULD be taken into account.

Receiving States MUST support all algorithms at points where they wish to validate the signature on eMRTDs.

For use in their CSCA, Signing keys and, where applicable, Document Security Objects, issuing States or organizations SHALL support one of the algorithms below.

4.1.6.1 RSA

Those issuing States or organizations implementing the RSA algorithm for signature generation and verification of certificates and the Document Security Object (SO_D) SHALL use [RFC 4055]. [RFC 4055] specifies two signature mechanisms, RSASSA-PSS and RSASSA-PKCS1_v15. It is RECOMMENDED that issuing States or organizations generate signatures according to RSASSA-PSS, but receiving States MUST also be prepared to verify signatures according to RSASSA-PKCS1_v15.

4.1.6.2 Digital Signature Algorithm (DSA)

Those issuing States or organizations implementing DSA for signature generation or verification SHALL use [FIPS 186-4].

4.1.6.3 Elliptic Curve DSA (ECDSA)

Those issuing States or organizations implementing ECDSA for signature generation or verification SHALL use [X9.62] or [ISO/IEC 15946]. The elliptic curve domain parameters used to generate the ECDSA key pair MUST be described explicitly in the parameters of the public key, i.e. parameters MUST be of type ECParameters (no named curves, no implicit parameters) and MUST include the optional co-factor. ECPoints MUST be in uncompressed format.

It is RECOMMENDED that the guideline [TR 03111] be followed.

4.1.6.4 Hashing Algorithms

SHA-224, SHA-256, SHA-384 and SHA-512, are the only permitted hashing algorithms. See [FIPS 180-2].

4.1.7 Cryptographic Algorithms for LDS2 Signer Certificates

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.

4.1.8 Cryptographic Algorithms for Visa or ETD Signer Certificates

The visa or ETD Signers MUST use ECDSA, irrespective of the algorithm used in the CSCA and Document Signing keys.

4.2 Authorization PKI

Issuing States or organizations that implement LDS2 SHALL have the following key pair types:

- Country Verifying CA (CVCA) Key Pair;
- Document Verifier (DV) Key Pair; and
- Terminal Key Pair.

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

4.2.1 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). As a consequence, Document Verifiers and terminals will have to be provided with several key pairs. 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. Details are provided in Doc 9303-11.

4.2.2 Cryptographic Algorithms for SPOC

The TLS Encryption Suites to be used for the SPOC protocol are listed in Table 3.

Table 3. 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 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 Table 3 which uses Advanced Encryption Standard (AES) for encryption SHALL be in accordance with [TLSAES] specification.

5. DISTRIBUTION MECHANISMS

For eMRTD PKI, the PKI objects need to be distributed to the receiving States. A number of different distribution mechanisms are used, depending on the type of object and operational requirements. It is important to note that distribution of these objects does NOT establish trust in those objects, or the private/public keys associated with them. Mechanisms for establishing trust are specified in Section 6.1.

The distribution mechanism for the Authorization PKI is covered in Section 8.

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

- CSCA certificates;
- CSCA Link Certificates;
- Document Signer certificates;
- LDS2 Signer certificates;
- Initial CVCA certificates;
- CVCA Link certificates;
- DV certificates;
- Bar Code Signer certificates;
- CRLs (null and non-null);
- Master List Signer certificates; Master Lists; and
- Deviation List Signer certificates; Deviation Lists.

The distribution mechanisms used in the eMRTD and Authorization PKI include:

- PKD;
- Bilateral exchange;
- SPOC;
- Master Lists;

- Deviation Lists; and
- eMRTD contactless IC.

A primary and secondary (where relevant) distribution mechanism is specified for each object as outlined in Table 4.

Table 4. Distribution of PKI Objects

	Contactless IC	SPOC	Bilateral	PKD	Deviation List	Master List	Notes
CSCA Certificates			Y (primary)			Y (secondary)	
Document Signer Certificates	Y (primary)			Y (secondary)			Certificates written at same time SOD is written
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 (secondary)	Y (primary)			CRLs issued by CSCA include revocation information relevant to LDS2 PKI objects
Master List Signer Certificates						Y	
Bar Code Signer Certificates			Y (secondary)	Y (primary)			Bar Code Signers are not encoded in the Bar Code and hence distribution must be ensured for validation of bar code
Master Lists			Y	Y			
Deviation List Signer Certificates					Y		

Operationally, receiving States are not obliged to use both the primary and secondary source. In the daily operation of an Inspection System, it is at the inspecting authority's discretion whether to use the primary or the secondary source. If the authority of the receiving State uses the secondary source for a certificate or CRL in its daily operations, it should be prepared to support the primary source as well.

Issuing States or organizations need to plan their key pair rollover strategies for both CSCA keys and Signer Keys in order to enable propagation of certificates and CRLs into receiving States' border control systems in a timely manner. Ideally propagation will occur within 48 hours, but some receiving States may have remote and poorly connected border outposts to which it may take more time for certificates and CRLs to propagate out. Receiving States SHOULD make every effort to distribute these certificates and CRLs to all border stations within 48 hours.

Issuing States or organizations should expect that CSCA certificates (C_{CSCA}) will be propagated by receiving States within 48 hours.

Issuing States or organizations ensure the timely propagation of Document Signer certificates (C_{DS}) by including the Document Signer certificate (C_{DS}) within the Document Security Object (SO_D). They should expect that Document Signer certificates (C_{DS}) published in the PKD will also be propagated to border stations within 48 hours.

The Bar Code Signer Certificates are not contained in the digital seal itself. Hence, a country that issues documents protected with digital seals MUST publish all its Bar Code Signer Certificates. The primary distribution channel for Bar Code Signer Certificates is PKD/bilateral. Other mechanisms, e.g. publication on a website, are secondary channels.

For Bar Code Signers, Publication MUST adhere to the following principles:

- as soon as a new certificate is created, it MUST be published with a delay of no more than 48 hours; and
- the certificates MUST remain published until their expiration or revocation.

Receiving States SHOULD make every attempt whether electronically or by other means to act upon CRLs, including those CRLs issued under exceptional circumstances.

Timely propagation of Master List Signer certificates is ensured by including them within each Master List.

5.1 PKD Distribution Mechanism

ICAO provides a Public Key Directory (PKD) service. This service SHALL accept PKI objects, including certificates, CRLs and Master Lists, from PKD participants, store them in a directory, and make them accessible to all receiving States.

CSCA certificates (C_{CSCA}) are not stored individually as part of the ICAO PKD service. However, they may be present in the PKD if they are contained on Master Lists.

Every certificate remains in the PKD until its certificate validity period has expired, regardless of whether or not the corresponding private key is still in use.

Certificates, CRLs and Master Lists stored in the PKD by all PKD participants SHALL be made available to all parties (including non-PKD participants) that need this information for validating the authenticity and integrity of digitally stored eMRTD data, LDS2 Objects and VDS objects.

5.1.1 PKD Upload

Only PKD participants MAY upload certificates, CRLs and Master Lists to the PKD. All certificates and CRLs MUST comply with the profiles in Section 7. All Master Lists MUST comply with the specifications in Section 9.

The PKD consists of a “Write Directory” and a “Read Directory”. PKD participants SHALL use the Lightweight Directory Access Protocol (LDAP) to upload their objects to the Write Directory. Once the digital signature has been verified on an object, and other due diligence checks completed, the object is published in the Read Directory.

5.1.2 PKD Download

Read access to all certificates, CRLs and Master Lists published in the PKD SHALL be available to PKD participants and non-participants. Access control SHALL NOT be implemented for PKD read access.

It is the receiving State’s responsibility to distribute objects downloaded from the PKD to its Inspection Systems and to maintain a current CRL cache along with the certificates necessary to verify the signatures on eMRTD data.

5.2 Bilateral Exchange Distribution Mechanism

For CRLs and CSCA certificates (C_{CSCA}), the primary distribution channel is bilateral exchange between issuing States or organizations and receiving States. Bilateral exchange can also be used to distribute Master Lists.

The specific technology used for that bilateral exchange may vary depending on the policies of each issuing State or organization that has a need to distribute its certificates, CRLs and Master Lists, as well as the policies of each receiving State that needs access to those objects. Some examples of technologies that may be used in bilateral exchange include:

- diplomatic courier/pouch;
- email exchange;
- download from a website associated with the issuing CSCA; and
- download from an LDAP server associated with the issuing CSCA.

This is not an exhaustive list and other technologies may also be used.

5.3 Master List Distribution Mechanism

Master Lists are a supporting technology for the bilateral distribution scheme. As such, distribution of CSCA certificates via Master Lists is a subset of the bilateral distribution scheme.

A Master List is a digitally signed list of the CSCA certificates that are “trusted” by the receiving State or organization that issued the Master List. CSCA self-signed Root certificates and CSCA Link certificates may be included in a Master List. The structure and format of a Master List is defined in Section 8. Publication of a Master List enables other receiving States or organizations to obtain a set of CSCA certificates from a single source (the Master List issuer) rather than establish a direct bilateral exchange agreement with each of the issuing authorities or organizations represented on that list.

A Master List Signer is authorized by a CSCA to compile, digitally sign, and issue Master Lists. Master Lists MUST NOT be signed and issued directly by a CSCA itself. Master List Signer certificates MUST comply with the certificate profile defined in Section 7.

Before issuing a Master List the issuing Master List Signer SHOULD extensively validate the CSCA certificates to be countersigned, including ensuring that the certificates indeed belong to the identified CSCAs. The procedures used for this out-of-band validation SHOULD be reflected in the published certificate policies of the CSCA that issued the Master List Signer certificate.

Each Master List MUST include the Master List Signer's certificate that will be used to verify the signature on that Master List as well as the CSCA certificates of the CSCA that issued that Master List Signer certificate.

If new CSCA certificates have been received by a receiving State, and its validation procedures have been completed, it is RECOMMENDED that a new Master List be compiled and issued.

Use of a Master List does enable more efficient distribution of CSCA certificates for some receiving States. However a receiving State making use of Master Lists MUST still determine its own policies for establishing trust in the certificates contained on that list (see Section 6 for details).

6. PKI TRUST AND VALIDATION

PKI Trust and Validation differ between the eMRTD PKI and Authorization PKI.

6.1 eMRTD PKI

In the eMRTD PKI environment, the Inspection Systems in receiving States act in the role of PKI relying parties. Successful verification of the digital signature on the Document Security Object of an eMRTD ensures the authenticity and integrity of the data stored on the contactless IC of that eMRTD. That signature verification process requires that the relying party establish that the Document Signer public key used to verify the signature is itself "trusted".

The various distribution mechanisms defined in Section 5 allow receiving States to gain access to the certificates and CRLs that they need to verify digital signatures in question. However, these distribution schemes do not establish trust in those certificates, CRLs or the public keys that will be used to verify signatures on those certificates and CRLs.

The public keys contained in CSCA certificates (CCSCA) are used to verify the digital signature on certificates and CRLs. Therefore, to accept an eMRTD from another issuing State, the receiving State MUST already have placed into some form of trust store, accessible by their border control system, a trusted copy of the issuing State or organization CSCA certificate (CCSCA), or other form of Trust Anchor information for that CSCA public key as derived from the certificate.

It is a receiving State's responsibility to establish trust in the CSCA certificates (C_{CSCA}) and store the certificates (or information from the certificates) as Trust Anchors, in a secure way for use by their border inspection systems.

6.1.1 Trust Anchor Management

As specified in [RFC 5280] a Trust Anchor must be established that can be used to anchor the validation procedure for a given Document Signer, Master List Signer, Deviation List Signer or other type of certificate.

Each Trust Anchor is comprised of a trusted public key and associated metadata. Trust Anchors MUST include, at a minimum:

- the trusted public key and any associated key parameters;
- the public key algorithm;
- the name of the key owner; and
- the value of the `SubjectAltName` extension of the CSCA certificate containing the ICAO assigned three-letter code of the issuing authority or organization. Although this is not used in the certification path or CRL validation procedures, it is used in Passive Authentication defined in Doc 9303-11.

In the eMRTD application, a separate Trust Anchor is established for each public key of a given CSCA. For the initial public key obtained from a CSCA, trust MUST be established through an out-of-band mechanism. For example, if a CSCA certificate was downloaded from a server associated with the CSCA, out-of-band communication (e.g. phone or email) could be used to verify that the downloaded certificate is in fact the authentic certificate for that CSCA. Also, the relying party might analyse the policies, procedures and practices of the issuing CSCA to determine whether they are secure enough to satisfy the local requirements for use of certificates. Once an initial Trust Anchor is established for a given CSCA, the process could be simplified for subsequent keys for that same CSCA. If the CSCA issues a CSCA Link certificate, then out-of-band communication with the CSCA to verify the authenticity of the new certificate could be skipped because the already trusted public key for that same CSCA is used to verify the signature on that CSCA Link certificate.

Trust Anchor information may be stored as a trusted copy of the CSCA certificate itself, or in some other trusted format.

Because signatures on certificates issued by CSCAs need to be verifiable long after that CSCA has updated its key pair, a receiving State will typically have more than one Trust Anchor for the same CSCA at any one time. If a CSCA has undergone a name change, some of these Trust Anchors will contain the old CSCA name and others will contain the new name.

6.1.2 Certificate/CRL Validation and Revocation Checking

As part of the process of verifying the authenticity and integrity of data objects in the eMRTD application (e.g. Document Security Objects, Master Lists, Deviation Lists, etc.), a Receiving State:

- validates the certificate used to verify the signature on the data object (e.g. Document Signer Certificate, Master List Signer certificate, Deviation List Signer certificate);
- validates the CRL that is used to check the revocation status of the certificate in question; and
- processes the CRL to verify the revocation status of the certificate in question.

Sample algorithms for these processes are available, such as those specified in [RFC 5280]. Receiving States need not implement the specific algorithm defined in RFC 5280, but MUST provide functionality equivalent to the external behaviour resulting from this procedure. Any algorithm may be used by a particular implementation as long as it derives the correct result.

Appendix D provides guidance for receiving States that choose to base their algorithm on that specified in [RFC 5280].

6.1.3 Bar Code Validation Authority

The bar code Validation Authority validates a digital seal by applying a Validation Policy. Doc 9303-13 specifies in detail validation criteria and algorithms to generate a validation status.

Figure 4 illustrates the functional architecture of the bar code Validation Authority. The bar code Validation Authority relies on validation software which can be deployed on any computer used by the border control authorities.

The validation software is connected with a reader that takes an image of the bar code to retrieve the bar code and the MRZ of the document, and also, an image of the document to retrieve its MRZ. To verify the validity of the signature of the digital seal, the validation software SHOULD be synchronized with the PKI publication point at least every 24 hours to retrieve the latest Bar Code Signer Certificates and CRLs.

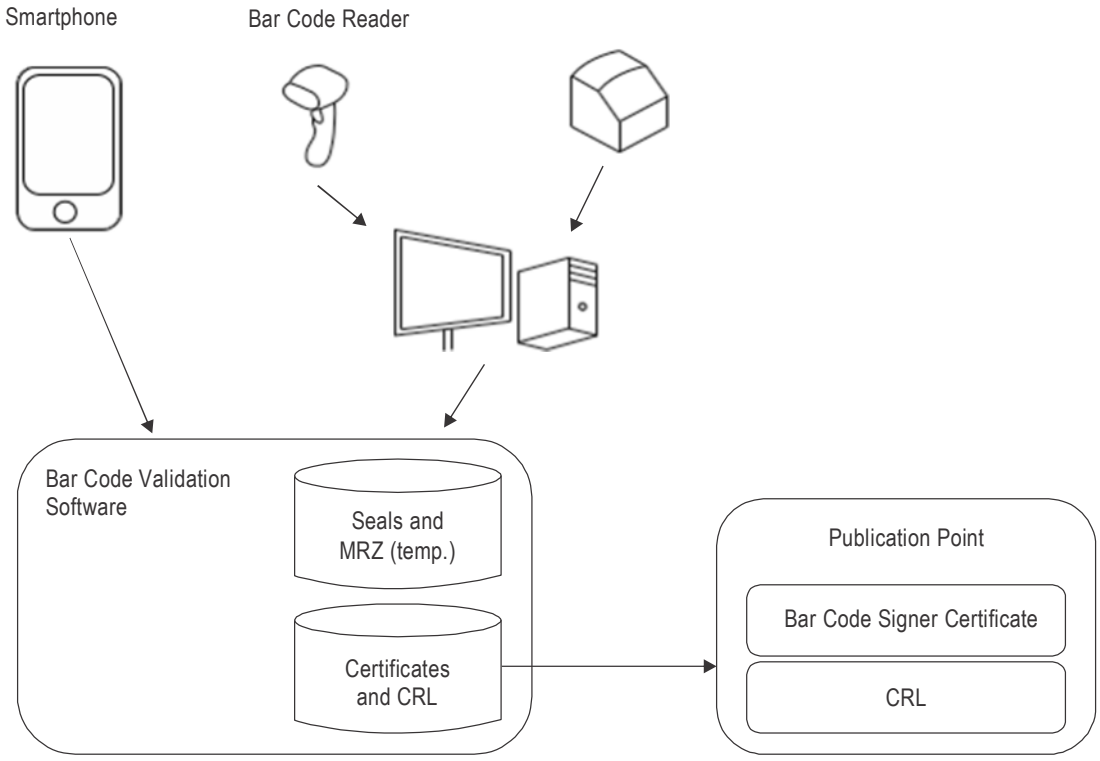

Figure 4. Bar Code Validation

The bar code validation software decodes the digital seal and the MRZs of any associated documents (e.g. visa or passport), validates the signature of the digital seal, and applies a validation policy (cf. 9303-13) to generate a validation status of the document.

In mobile scenarios, the validation software can also be directly run on a smartphone. Whereas the validity of the seal can be verified by the software on the smartphone, the comparison between the (signed) data inside the seal and the printed MRZs (e.g. of the visa or passport) MUST be done either manually, or by OCR of the MRZs out of the captured image, the latter being often a challenging problem in practice.

The following data are processed by the bar code validation software:

- Input data provided by readers, e.g. the images of visas or passports; and
- Certificates and CRLs.

6.2 Authorization PKI

For Authorization PKI, the Trust Anchor and Validation is handled differently.

6.2.1 Validation of Card Verifiable Certificates

For DV and terminal certificates in the authorization PKI, the Trust Anchor is the most 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. CERTIFICATE AND CRL PROFILES

Certificate profiles are defined for both the eMRTD PKI and the Authorization PKI.

7.1 eMRTD PKI

Issuing States or organizations MUST issue certificates and CRLs that conform to the profiles specified below. All certificates and CRLs MUST be produced in Distinguished Encoding Rule (DER) format to preserve the integrity of the signatures within them. The profiles for CSCA and DS certificates that were included in the sixth edition of this specification differ in some areas from the current profiles. Inspection Systems MUST be capable of handling certificates that were issued in accordance with those earlier profiles (see Appendix C) as well as the current profiles.

These profiles are based on the requirement that each issuing State or organization or entity SHALL create a single CSCA for the purpose of signing all Doc 9303 compliant eMRTDs.

Certificate profiles are defined in this section for the following certificate types:

- Country Signing CA;
- Document Signer;

- CSCA Master List Signer;
- Deviation List Signer; and
- Communications – even though it is not strictly needed today. This is a future proofing step. These certificates may be used for access to the PKD or for LDAP/EMAIL/HTTP communications between States. It is recommended that these certificates be issued by the CSCA.

The Country Signing CA, Document Signer, Deviation List Signer and CSCA Master List Signer objects are defined in Section 3.

The CRL profile is defined in Section 7.1.4.

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 present;
o optional — the field MAY be present;
C conditional — the field SHALL be present under certain conditions.

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.

Some of the requirements identified in these profiles are inherited from the referenced base profiles (e.g. RFC 5280). For convenience, the relevant text from the base profile that covers the specific requirement is duplicated in a table in Appendix B.

7.1.1 Certificate Profiles

Table 5 defines the certificate profile requirements common to all certificates for the fields of the certificate body. Table 6 defines the requirements for certificate extensions.

Table 5. Certificate Fields Profile

Certificate Component	*Presence*	*Comments*
`Certificate`	m	
`TBSCertificate`	m	See Table 6
`signatureAlgorithm`	m	Value inserted here dependent on algorithm selected
`signatureValue`	m	Value inserted here dependent on algorithm selected
`TBSCertificate`		
`version`	m	MUST be `v3`
`serialNumber`	m	MUST be positive integer and maximum 20 Octets MUST use 2's complement encoding and be represented in the smallest number of octets

Certificate Component	*Presence*	*Comments*
signature	m	Value inserted here MUST be the same as that in signatureAlgorithm component of Certificate sequence
issuer	m	countryName and serialNumber, if present, MUST be PrintableString Other attributes that have DirectoryString syntax MUST be either PrintableString or UTF8String countryName MUST be Upper Case See 7.1.1.1 for naming conventions
validity	m	MUST terminate with Zulu (Z) Seconds element MUST be present Dates through 2049 MUST be in UTCTime UTCTime MUST be represented as YYMMDDHHMMSSZ Dates in 2050 and beyond MUST be in GeneralizedTime. GeneralizedTime MUST NOT have fractional seconds GeneralizedTime MUST be represented as YYYYMMDDHHMMSSZ
subject	m	countryName and serialNumber, if present, MUST be PrintableString Other attributes that have DirectoryString syntax MUST be either PrintableString or UTF8String countryName MUST be Upper Case countryName in issuer and subject fields MUST match See 7.1.1.1 for naming conventions
subjectPublicKeyInfo	m	
issuerUniqueID	x	
subjectUniqueID	x	
extensions	m	See Table 6 on which extensions should be present Default values for extensions MUST NOT be encoded

Table 6. Certificate Extensions Profile

Extension name	*CSCA Self-Signed Root*		*CSCA Link*		*Document Signer*		*Master List Signer and Deviation List Signer*		*Communication*		*Comments*
	Presence	*Criticality*	*Presence*	*Criticality*	*Presence*	*Criticality*	*Presence*	*Criticality*	*Presence*	*Criticality*	
AuthorityKeyIdentifier	**o**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	
keyIdentifier	m		m		m		m		m		
authorityCertIssuer	o		o		o		o		o		
authorityCertSerialNumber	o		o		o		o		o		
SubjectKeyIdentifier	**m**	**nc**	**m**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	
subjectKeyIdentifier	m		m		m		m		m		
KeyUsage	**m**	**c**	**m**	**c**	**m**	**c**	**m**	**c**	**m**	**c**	
digitalSignature	x		x		m		m		o		Some communication certificates (e.g. TLS certificates) require that the `keyUsage` bits be set in accordance with the particular cipher suite used. Some cipher suites do, and some do not require the `digitalSignature` bit to be set.
nonRepudiation	x		x		x		x		x		
keyEncipherment	x		x		x		x		o		
dataEncipherment	x		x		x		x		x		
keyAgreement	x		x		x		x		o		
keyCertSign	m		m		x		x		x		
cRLSign	m		m		x		x		x		
encipherOnly	x		x		x		x		x		
decipherOnly	x		x		x		x		x		
PrivateKeyUsagePeriod	**m**	**nc**	**m**	**nc**	**m**	**nc**	**o**	**nc**	**o**	**nc**	
notBefore	o		o		o		o		o		At least one of `notBefore` or `notAfter` MUST be present MUST be encoded as `generalizedTime`
notAfter	o		o		o		o		o		

Extension name	***CSCA Self-Signed Root***		***CSCA Link***		***Document Signer***		***Master List Signer and Deviation List Signer***		***Communication***		***Comments***
CertificatePolicies	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	
PolicyInformation	m		m		m		m		m		
policyIdentifier	m		m		m		m		m		
policyQualifiers	o		o		o		o		o		
PolicyMappings	**x**		**x**		**x**		**x**		**x**		See Note 1
SubjectAltName	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	See 7.1.1.2
IssuerAltName	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	See 7.1.1.2
SubjectDirectoryAttributes	**x**		**x**		**x**		**x**		**x**		
Basic Constraints	**m**	**c**	**m**	**c**	**x**		**x**		**x**		
cA	m		m		x		x		x		
PathLenConstraint	m		m		x		x		x		MUST always be ‘0’
NameConstraints	**x**		**x**		**x**		**x**		**x**		See Note 1
PolicyConstraints	**x**		**x**		**x**		**x**		**x**		See Note 1
ExtKeyUsage	**x**		**x**		**x**		**m**	**c**	**m**	**c**	See 7.1.1.3
CRLDistributionPoints	**m**	**nc**	**m**	**nc**	**m**	**nc**	**m**	**nc**	**o**	**nc**	
distributionPoint	m		m		m		m		m		MUST be ldap, http or https See 7.1.1.4
reasons	x		x		x		x		x		
cRLIssuer	x		x		x		x		x		
InhibitAnyPolicy	**x**		**x**		**x**		**x**		**x**		See Note 1
FreshestCRL	**x**		**x**		**x**		**x**		**x**		See Note 2
privateInternetExtensions	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	See Note 3
NameChange	**o**	**nc**	**o**	**nc**	**x**		**x**		**x**		See 7.1.1.5
DocumentType	**x**		**x**		**m**	**nc**	**x**		**x**		See 7.1.1.6
Netscape Certificate Type	**x**		**x**		**x**		**x**		**x**		See Note 4
other private extensions	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	**o**	**nc**	

Note 1.— The extension, by definition, can only appear in intermediate CA certificates (certificates issued by one CA to another CA). Intermediate CA certificates are not used in the eMRTD PKI. Therefore this extension is prohibited from eMRTD certificates.

Note 2.— The freshest CRL extension is used to point to a delta CRL. Delta CRLs are not supported in the eMRTD PKI. Therefore this extension is prohibited.

Note 3.— There are two Private Internet Extensions (Authority Information Access and Subject Information Access) defined in RFC 5280 that are used to point to information about the issuer or subject of a certificate. These extensions are not required in the eMRTD PKI. However as they do not impact interoperability, and are non-critical, they may optionally be included in eMRTD certificates.

Note 4.— The Netscape Certificate Type extension can be used to limit the purposes for which a certificate can be used. The `extKeyUsage` and `basicConstraints` extensions are now the standard extensions for those purposes and are used in the eMRTD application. Because of the potential conflict between values in the standard extensions and in the Netscape proprietary extension, the Netscape extension is prohibited.

7.1.1.1 Issuer and Subject Field requirements

The Issuer and Subject Fields are common to all certificates, but specific restrictions apply for LDS2 Signer Certificates.

7.1.1.1.1 General requirements

The following naming and addressing conventions for `Issuer` and `Subject` fields are REQUIRED.

- `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.
- `commonName`. MUST be present.

Other attributes MAY also be included at the discretion of the issuing State or organization.

7.1.1.1.2 LDS2 Signer Certificate requirements

LDS2 Signer Certificates MUST comply with the Document Signer Certificate profile defined above with the exceptions defined in 7.1.2.

7.1.1.2 Issuer and Subject Alternative Name requirements

Because the functions served by alternative names in the eMRTD application are specific to this application, and different from those defined for the Internet PKI in [RFC 5280], values in the Subject Alternative Name extension of eMRTD certificates do not generally unambiguously identify the certificate subject.

In the eMRTD application, alternative names serve the following two functions.

The first function is to provide contact information for the subject and/or issuer of the certificate. For that purpose it SHOULD include at least one of the following:

- `rfc822Name`;
- `dNSName`; or
- `uniformResourceIdentifier`.

The second function is to provide a directory string made of ICAO assigned country codes. For this purpose certificates issued using this profile MUST additionally include a directory name that is constructed as follows:

- `localityName` that contains the ICAO country code as it appears in the MRZ; and

- if this country code does not uniquely define the issuing State or organization, the attribute `stateOrProvinceName` SHALL be used to indicate the ICAO assigned three-letter code for the issuing State or organization.

- Other attributes are not permitted.

In CSCA self-signed Root certificates, the `IssuerAltName` and `SubjectAltName` extensions MUST be identical. In CSCA Link certificates, the values MAY be different. For example, if a change has occurred with the `rfc822Name` of the CSCA immediately prior to issuance of a CSCA Link certificate, the `IssuerAltName` extension would contain the old `rfc822Name` and the `SubjectAltName` extension would contain the new `rfc822Name`. Any subsequent CSCA Link certificates would contain the new `rfc822Name` in both extensions.

7.1.1.3 Extended key usage extension requirements

The Object Identifier (OID) that must be included in the `extendedKeyUsage` extension for Master List Signer certificates is `2.23.136.1.1.3`.

The Object Identifier (OID) that must be included in the `extendedKeyUsage` extension for Deviation List Signer certificates is `2.23.136.1.1.8`.

For communication certificates the value of this extension depends on the communication protocol used (see RFC 5280, Section 4.2.1.12).

7.1.1.4 CRL distribution points extension requirements

CSCAs may publish their CRL in several places including the PKD, their own website, etc.

For CRLs that are published in locations other than the PKD (e.g. website or local LDAP server), the values that are to be included in this extension are under the control of the CSCA issuing the certificates and the CRL in question.

For CRLs submitted to the PKD, PKD participants MAY include two URL values for their CRL using the following template (replace “CountryCode” with the issuing State or organization ICAO assigned three-letter code). If this country code does not uniquely identify the issuing State or organization, the entry will be created by appending the symbol “_” to the three-letter country code in the MRZ, and then the ICAO assigned three-letter code for the issuing State or organization which uniquely identifies the issuing State or organization:

https://pkddownload1.icao.int/CRLs/CountryCode.crl
https://pkddownload2.icao.int/CRLs/CountryCode.crl

This is a mandatory extension, and revocation status checks are a mandatory part of the validation procedure. Therefore at least one value MUST be populated:

- The PKD values may be the only values in the extension;

- There may be additional values (e.g. a CSCA may also choose to publish its CRL on a website and include a pointer to that source); or

- A CSCA may also choose to include only a single value (e.g. a pointer to its website as a source) even if it also submits its CRL to the PKD.

The following examples illustrate the PKD values that would be populated in certificates issued by the issuing authority for Singapore and for Hong Kong:

Singapore PKD example:
https://pkddownload1.icao.int/CRLs/SGP.crl
https://pkddownload2.icao.int/CRLs/SGP.crl

Hong Kong example:
https://pkddownload1.icao.int/CRLs/CHN_HKG.crl
https://pkddownload2.icao.int/CRLs/CHN_HKG.crl

7.1.1.5 Name change extension

When a CSCA key rollover occurs, a certificate MUST be issued that links the old public key to the new public key to provide a secure transition for relying parties. Generally this is achieved through the issuance of a self-issued certificate where the `issuer` and `subject` fields are identical but the key used to verify the signature represents the old key pair and the certified public key represents the new key pair.

It is RECOMMENDED that CSCAs do not change their Distinguished Name (DN) unnecessarily as there is an adverse impact on relying parties (they must retain both the old and new names as valid CSCAs for the same issuing State or organization until all eMRPs signed under the old name have expired). However, if a name change is necessary, this MUST be conveyed to relying parties through the issuance of a CSCA Link certificate where the `issuer` field contains the old name and the `subject` field contains the new name. This CSCA Link certificate also conveys a key rollover where the key used to verify the signature represents the old key pair and the certified public key represents the new key pair. Certificates that convey both a CSCA name change and a key rollover for that CSCA MUST include the `NameChange` extension to identify the certificate as such. This has no effect on `PathLengthConstraint`; it remains `'0'`.

In addition, the `NameChange` extension MAY also be included in the new CSCA self-signed certificate created upon the change of the CSCA DN. In such a self-signed CSCA Root certificate, both the `issuer` and `subject` fields contain the new DN. Unlike the CSCA self-issued link certificate, containing both the old and new DN for the CSCA, inclusion of the `NameChange` extension in a CSCA self-signed Root certificate simply indicates that a name change has occurred and does not link the old DN to the new one.

A CSCA MUST NOT re-use certificate serial numbers. Each certificate issued by a CSCA, regardless of whether that CSCA has undergone a name change or not, MUST be unique.

ASN.1 for Name Change extension:

```
nameChange EXTENSION ::= {
    SYNTAX NULL
    IDENTIFIED BY id-icao-mrtd-security-extensions-nameChange}

id-icao-mrtd-security-extensions OBJECT IDENTIFIER ::= {id-icao-
mrtd-security 6}
id-icao-mrtd-security-extensions-nameChange OBJECT IDENTIFIER ::=
{id-icao-
mrtd-security-extensions 1}
```

7.1.1.6 Document type extension

The `DocumentType` extension MUST be used to indicate the document types, as they appear in the MRZ, that the corresponding Document Signer is allowed to produce. This extension MUST always be set to non-critical.

ASN.1 for Document Type List extension:

```
documentTypeList EXTENSION ::= {
   SYNTAX DocumentTypeListSyntax
   IDENTIFIED BY id-icao-mrtd-security-extensions-documentTypeList}

DocumentTypeListSyntax ::= SEQUENCE {
   version             DocumentTypeListVersion,
   docTypeList   SET OF DocumentType }

DocumentTypeListVersion ::= INTEGER {v0(0)}

-- Document Type as contained in MRZ, e.g. "P" or "ID" where a
-- single letter denotes all document types starting with that letter
DocumentType ::= PrintableString(SIZE(1..2))

id-icao-mrtd-security-extensions-documentTypeList OBJECT
IDENTIFIER ::= {id-icao-mrtd-security-extensions 2}
```

7.1.2 LDS2 Signer Certificate Profile

LDS2 Signer certificates MUST comply with the Document Signer certificate profile defined in 7.1.1 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, as specified in Doc 9303-3.
- `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 7 below. All other certificate extensions MUST NOT be included.

Table 7. Mandatory Certificate Extensions for LDS2

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 OBJECT IDENTIFIER ::= {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 Doc 9303-10.

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.

7.1.3 Bar Code Signer Certificate Profile

The Bar Code Signer certificates MUST comply with the LDS2 Signer certificate profile. Since Bar Code Signer certificates serve a different role than LDS2 certificates, their profile deviates in some respects. In particular, there are specific requirements for the subjectDN of the Bar Code signer certificate and the serial number (see Doc 9303-13).

Subject Field:
The subject field of Bar Code Signer Certificates must be populated as follows:

- `commonName`: MUST be present. MUST consist of two uppercase characters, printableString format, that uniquely define the Bar Code Signer within one country, and MUST match the letters 3 and 4 of the Signer Identifier in the bar code, as specified in Doc 9303-13.

- `countryName`: MUST consist of the two-letter country code (see Doc 9303-3) of the Bar Code Signer, uppercase characters, printableString format, and MUST match letters 1 and 2 of the Signer Identifier in the bar code, as specified in Doc 9303-13.

- Other attributes MUST NOT be included.

Certificate extensions:
Bar Code Signer certificates MUST contain the certificate extensions identified in Table 8 below. All other certificate extensions MUST NOT be included.

Table 8. Allowed Extensions for Bar Code Signer Certificates

Extension name	LDS2 Signer		Comments
	Presence	**Criticality**	
AuthorityKeyIdentifier	**m**	**nc**	
keyIdentifier	m		
authorityCertIssuer	o		
authorityCertSerialNumber	o		
DocumentType	**o**		This extension indicates the document type, which the Bar Code Signer is allowed to produce
ExtKeyUsage	**m**	**c**	See note below

Note.— The EKU extension for each Bar Code Signer certificate type MUST be populated as indicated below.

```
id-icao-mrtd-security-vds OBJECT IDENTIFIER ::= {id-icao-mrtd-security 11}
id-icao-vdsSigner OBJECT IDENTIFIER ::= {id-icao-mrtd-security-vds 1}
```

7.1.4 CRL Profile

Table 9 defines the CRL profile requirements for the fields of the CRL body. Table 10 defines the CRL profile requirements for CRL and CRL Entry extensions.

Table 9. CRL Fields Profile

Certificate List Component	*CSCA CRL*	*Comments*
CertificateList	m	
tBSCertList	m	See Table 10
signatureAlgorithm	m	Value inserted here dependent on algorithm selected
signatureValue	m	Value inserted here dependent on algorithm selected
tBSCertList		
Version	m	MUST be v2
Signature	m	value inserted here MUST be the same as that in signatureAlgorithm component of CertificateList sequence
Issuer	m	countryName and serialNumber, if present, MUST be PrintableString Other attributes that have DirectoryString syntax MUST be either PrintableString or UTF8String countryName MUST be Upper Case
thisUpdate	m	MUST terminate with Zulu (Z) Seconds element MUST be present Dates through 2049 MUST be in UTCTime UTCTime MUST be represented as YYMMDDHHMMSSZ Dates in 2050 and beyond MUST be in GeneralizedTime. GeneralizedTime MUST NOT have fractional seconds GeneralizedTime MUST be represented as YYYYMMDDHHMMSSZ
nextUpdate	m	MUST terminate with Zulu (Z) Seconds element MUST be present Dates through 2049 MUST be in UTCTime UTCTime MUST be represented as YYMMDDHHMMSSZ Dates in 2050 and beyond MUST be in GeneralizedTime. GeneralizedTime MUST NOT have fractional seconds GeneralizedTime MUST be represented as YYYYMMDDHHMMSSZ
revokedCertificates	c	SHALL be present if there are revoked certificates. If there are no revoked certificates it SHALL NOT be present. If present, MUST NOT be empty

Certificate List Component	*CSCA CRL*	*Comments*
`crlExtensions`	m	See Table 10 on which extensions should be present Default values for extensions MUST NOT be encoded

Table 10. CRL and CRL Entry Extensions Profile

Extension Name	*CSCA CRL*	*Criticality*	*Comments*
CRL Extensions			
`authorityKeyIdentifier`	m	**nc**	This MUST be the same value as the `subjectKeyIdentifier` field in the CRL issuer's certificate.
`keyIdentifier`	m		
`authorityCertIssuer`	o		
`authorityCertSerialNumber`	o		
`issuerAlternativeName`	o	**nc**	See Note 1
`cRLNumber`	m	**nc**	MUST be non-negative integer and maximum 20 Octets MUST use 2's complement encoding and be represented in the smallest number of octets
`deltaCRLIndicator`	x		
`issuingDistributionPoint`	x		
`freshestCRL`	x		
`CRL Entry Extensions`			
`reasonCode`	x		
`holdInstructionCode`	x		
`invalidityDate`	x		
`certificateIssuer`	x		
`other private extensions`	o	**nc**	

Note 1.— If a CSCA has undergone a name change, this extension MAY be included in CRLs issued following the CSCA name change. If present, the value(s) in this extension MUST be identical to the `issuer` *field of certificates issued by the CSCA under that previous name. Once all certificates issued under a previous CSCA name have expired, that CSCA name can be excluded from subsequent CRLs. Inspection Systems are not required to process this extension. Given that ICAO Doc 9303 dictates a single CSCA per country, the* `countryName` *component of the issuer field is sufficient to uniquely identify the CSCA. The latest public key of that CSCA is used to verify the signature of the CRL. Since a CSCA issues a single CRL, this CRL covers all certificates issued with that* `countryName`*. In addition to that mandatory check, an optional check that the* `issuer` *field of the certificate is equal to the* `issuer` *field of the CRL or one of the values of the* `issuerAltName` *extension in the CRL MAY also be done*.

Note 2.— It is possible that the CRL contains other revocation information, for example, concerning system operator or registration authority certificates.

7.2 Authorization PKI

The authorization PKI includes X.509 certificates for SPOC and card-verifiable certificates for CVCA, DV and terminals. This section specifies the profiles for SPOC certificates, CVCA, DV and IS certificates. An overview of the data objects contained in card-verifiable certificates is provided, and the encoding of those objects is also covered.

7.2.1 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; and
- `countryName` MUST be present in the Subject field.

LDS2 SPOC certificates (client and server) MUST comply with the communication certificate profile defined in Section 7.1, 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, as specified in Doc 9303-3.
- `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.

7.2.2 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 in Table 11 SHALL be used. Details on encoding values can be found in Doc 9303-11.

Table 11. 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

7.2.2.1 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.

7.2.2.2 Certificate Authority Reference and 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 12 and the following rules:

a) Country Code:

- The Country Code SHALL be the Doc 9303-3 two-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;
  - A numeric Sequence Number SHALL consists of the characters "0…9".
  - An alphanumeric Sequence Number SHALL consist of the characters "0…9" and "A…Z".
- The Sequence Number MUST start with the Doc 9303-3 two-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 12. Certificate Holder Reference

	Encoding	Length
Country Code	Doc 9303-3	2F
Holder Mnemonic	ISO/IEC 8859-1	9V
Sequence Number	ISO/IEC 8859-1	5F

7.2.2.3 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.

7.2.2.4 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.

7.2.2.5 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.

7.2.2.6 Certificate Extensions (Authorization 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 13:

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 13. 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.

7.2.2.7 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.

7.2.3 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 14.

Table 14. 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.

Name	Tag	Len	Value	Comment
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).

7.2.3.1 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.

7.2.3.1.1 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.

7.2.3.1.2 Elliptic Curve Points

The conversion of elliptic curve points to octet strings is specified in [TR-03111]. The uncompressed format SHALL be used.

7.2.3.1.3 Dates

A date is encoded in 6 digits “d1…d6” in the format YYMMDD using timezone GMT. It is converted to an octet string “o1…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 is to be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

7.2.3.1.4 Character Strings

A character string “c1…cn” is a concatenation of n characters cj with $1 \leq j \leq n$. It SHALL be converted to an octet string “o1…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.

7.2.3.1.5 Octet Strings

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

7.2.3.1.6 Object Identifiers

An object identifier “i1.i2…in” is encoded as an ordered list of n unsigned integers ij with 1 ≤ j ≤ n. It SHALL be converted to an octet string “o1…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:

$$i=i1\cdot 40+i2$$

2) The remaining integers ij are directly converted to octet strings oj−1 with 3 ≤ j ≤ n.
More details on the encoding can be found in [X.690].

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.

7.2.3.1.7 Sequences

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

7.2.3.2 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.

7.2.3.2.1 RSA Public Keys

The data objects contained in an RSA public key are shown in Table 15. The order of the data objects is fixed.

Table 15. 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

7.2.3.2.2 Elliptic Curve Public Keys

The data objects contained in an EC public key are shown in Table 16. 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; and
- Certificate Requests MUST always contain domain parameters.

Table 16. 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

8. SPOC PROTOCOL

The 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.

8.1 SPOC Related Structures

The following structures are defined for use in SPOC messages.

8.1.1 Certificate Request Structure

Certificate requests are reduced card-verifiable certificates that may carry an additional signature. The certificate request profile specified in Table 17 SHALL be used.

Table 17. 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

8.1.1.1 Certificate Profile Identifier

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

8.1.1.2 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.

8.1.1.3 Public Key

Certificate Requests MUST always contain domain parameters.

8.1.1.4 Certificate Holder Reference

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

8.1.1.5 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).

8.2 SPOC Protocol Messages

This section details the messages used in the SPOC protocol.

8.2.1 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 two-letter country code according to Doc 9303-3. 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 8.1.1. The coding MUST follow the specifications in Section 7.2.3.1.

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.

8.2.2 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 two-letter country code according to Doc 9303-3. 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 7.2.3.

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 parameters 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

8.2.3 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 two-letter country code according to Doc 9303-3. 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 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.

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).

8.2.4 General Messages

Intended Use:

This message is sent by a SPOC to a foreign SPOC in order to send a 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 two-letter country code according to Doc 9303-3. 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 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.

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 the 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.

8.3 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 8.3.3.

8.3.1 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 be implemented in accordance to [WS-IBP] and [WSI-SSBP].

The SPOC SOAP interface MUST conform to WSDL definitions as described in Section 8.3.3.

8.3.2 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 7.2.1 SPOC TLS server certificate; and
- 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 7.2.1 SPOC TLS client certificate; and
- 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 4.2.2. Both the server and the client sides 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 4.2.2 which uses Advanced Encryption Standard (AES) for encryption SHALL be in accordance with the [TLSAES] specification.

8.3.3 WSDL for SPOC Web Service Interface

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

```
<?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>
```

9. CSCA MASTER LIST STRUCTURE

Master Lists are implemented as instances of the `ContentInfo` Type, as specified in [RFC 5652]. The `ContentInfo` MUST contain a single instance of the `SignedData` Type as profiled below. No other data types are included in the `ContentInfo`. All Master Lists MUST be produced in DER format to preserve the integrity of the signatures within them.

9.1 SignedData Type

The processing rules in [RFC 5652] apply.

The specification of Master List structure uses the following terminology for presence requirements of each field:

- m mandatory — the field MUST be present;
- r recommended — the field SHOULD be present;
- x do not use — the field MUST NOT be present;
- o optional — the field MAY be present.

Table 18. Master List

Value		Comments
`SignedData`		
`Version`	m	Value = `v3`
`digestAlgorithms`	m	
`encapContentInfo`	m	
`eContentType`	m	`id-icao-cscaMasterList`
`eContent`	m	The encoded contents of an `cscaMasterList`
`Certificates`	m	The Master List Signer certificate MUST be included and the CSCA certificate, which can be used to verify the signature in the `signerInfos` field SHOULD be included.
`Crls`	x	
`signerInfos`	m	It is RECOMMENDED that States only provide 1 `signerinfo` within this field.

Value		Comments
SignerInfo	m	
Version	m	The value of this field is dictated by the `sid` field. See [RFC 5652] for rules regarding this field.
Sid	m	
subjectKeyIdentifier	r	It is RECOMMENDED that this field be supported rather than `issuerandSerialNumber`.
digestAlgorithm	m	The algorithm identifier of the algorithm used to produce the hash value over `encapsulatedContent` and `SignedAttrs`. See note below.
signedAttrs	m	Additional attributes may be included. However these do not have to be processed by Receiving States except to verify the signature value. `signedAttrs` MUST include signing time (see [PKCS #9]).
signatureAlgorithm	m	The algorithm identifier of the algorithm used to produce the signature value, and any associated parameters. See note below.
signature	m	The result of the signature generation process.
unsignedAttrs	o	Although this field MAY be included, Receiving States may choose to ignore it.

Note.— `DigestAlgorithmIdentifiers` MUST omit "NULL" parameters, while the `SignatureAlgorithmIdentifier` (as defined in RFC 3447) MUST include `NULL` as the parameter if no parameters are present, even when using SHA2 Algorithms in accordance with RFC 5754. Implementations MUST accept `DigestAlgorithmIdentifiers` with both conditions, absent parameters or with `NULL` parameters.

9.2 ASN.1 Master List Specification

```
CscaMasterList
{ joint-iso-itu-t(2) international-organization(23) icao(136) mrtd(1)
security(1) masterlist(2)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS

 -- Imports from RFC 5280 [PROFILE], Appendix A.1
     Certificate
       FROM PKIX1Explicit88
```

```
          { iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5) pkix(7)
              mod(0) pkix1-explicit(18) };
-- CSCA Master List

CscaMasterListVersion ::= INTEGER {v0(0)}

CscaMasterList ::= SEQUENCE {
  version            CscaMasterListVersion,
  certList           SET OF Certificate }

-- Object Identifiers

id-icao-cscaMasterList OBJECT IDENTIFIER ::=
                                     {id-icao-mrtd-security 2}
id-icao-cscaMasterListSigningKey OBJECT IDENTIFIER ::=
                                     {id-icao-mrtd-security 3}
END
```

10. DEVIATION LIST STRUCTURE

The Deviation List is implemented as a SignedData type, as specified in [RFC 3852]. All Deviation Lists MUST be produced in DER format to preserve the integrity of the signatures within them.

The range of deviations will be bounded by:

- date range (including both the issue and expiry date);
- issuer name and serial number;
- Subject Key Identifier of DSC;
- list of eMRTD numbers.

Appropriate combinations of these values will be used to accurately bind the range of MRTDs affected. When combining values, they are to be processed as joined by “AND”. There is no option to process values as joined using “OR”.

10.1 SignedData Type

The processing rules in [RFC 3852] apply:

m mandatory – the field MUST be present;
r recommended – the field SHOULD be present;
x do not use – the field MUST NOT be populated;
o optional – the field MAY be present.

Table 19. Deviation List

Value		Comments
SignedData		
version	m	Value = v3
digestAlgorithms	m	
encapContentInfo	m	
eContentType	m	id-icao-DeviationList
eContent	m	The encoded contents DeviationList
certificates	m	States MUST include the Deviation List Signer certificate and SHOULD include the CSCA certificate, which can be used to verify the signature in the signerInfos field.
crls	x	
signerInfos	m	It is RECOMMENDED that States provide only 1 signerInfo within this field.
SignerInfo	m	
version	m	The value of this field is dictated by the sid field. See [RFC 3852] Section 5.3 for rules regarding this field.
sid	m	
subjectKeyIdentifier	r	It is RECOMMENDED that States support this field over issuerandSerialNumber.
digestAlgorithm	m	The algorithm identifier of the algorithm used to produce the hash value over encapsulatedContent and SignedAttrs.
signedAttrs	m	Producing States may wish to include additional attributes for inclusion in the signature, however these do not have to be processed by receiving States except to verify the signature value. signedAttrs MUST include signing time (ref. PKCS#9).
signatureAlgorithm	m	The algorithm identifier of the algorithm used to produce the signature value, and any associated parameters.
signature	m	The result of the signature generation process.
unsignedAttrs	x	

10.2 ASN.1 specification

```
DeviationList
{ joint-iso-itu-t (2) international-organization(23) icao(136) mrtd(1) security(1)
deviationlist(7)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS

  -- Imports from RFC 3280 [PROFILE], Appendix A.1
       AlgorithmIdentifier
           FROM PKIX1Explicit88
                  { iso(1) identified-organization(3) dod(6)
                    internet(1) security(5) mechanisms(5) pkix(7)
                    mod(0) pkix1-explicit(18) }

  -- Imports from RFC 3852
       SubjectKeyIdentifier, Digest, IssuerAndSerialNumber
           FROM CryptographicMessageSyntax2004
                  { iso(1) member-body(2) us(840) rsadsi(113549)
                    pkcs(1) pkcs-9(9) smime(16) modules(0)
                    cms-2004(24) };

DeviationListVersion ::= INTEGER {v0(0)}

DeviationList ::= SEQUENCE {
 version      DeviationListVersion,
 digestAlgorithm AlgorithmIdentifier OPTIONAL,
 deviations   SET OF Deviation
}

Deviation ::= SEQUENCE{
 documents       DeviationDocuments,
 descriptions SET OF DeviationDescription
}

DeviationDescription ::= SEQUENCE{
 description     PrintableString OPTIONAL,
 deviationType   OBJECT IDENTIFIER,
 parameters   [0] ANY DEFINED BY deviationType OPTIONAL,
 nationalUse     [1] ANY OPTIONAL

 -- The nationalUse field is for internal State use, and is not governed
 -- by an ICAO specification.
}

DeviationDocuments ::= SEQUENCE {
 documentType    [0] PrintableString (SIZE(2)) OPTIONAL,
          -- per MRZ, e.g. 'P'
 dscIdentifier   DocumentSignerIdentifier OPTIONAL,
```

```
 issuingDate     [4] IssuancePeriod OPTIONAL,
 documentNumbers    [5] SET OF PrintableString OPTIONAL
}

DocumentSignerIdentifier ::= CHOICE{
 issuerAndSerialNumber [1] IssuerAndSerialNumber,
 subjectKeyIdentifier [2] SubjectKeyIdentifier,
 certificateDigest  [3] Digest   --  if  used,  digestAlgorithm  must  be  present  in
DeviationList
}

IssuancePeriod ::= SEQUENCE {
 firstIssued   GeneralizedTime,
 lastIssued   GeneralizedTime
}

-- CertField is used to define which part of a certificate is
  affected by a coding error. Parts of the Body are identified by
  the corresponding value of CertificateBodyField, extensions
  by the corresponding OID identifying the extension.

CertField ::= CHOICE {
 body  CertificateBodyField,
 extension OBJECT IDENTIFIER
}
CertificateBodyField ::= INTEGER {
 generic(0), version(1), serialNumber(2), signature(3), issuer(4),
 validity(5), subject(6), subjectPublicKeyInfo(7),
 issuerUniqueID(8), subjectUniqueID(9)
}

Datagroup ::= INTEGER
              {dg1(1), dg2(2), dg3(3), dg4(4), dg5(5), dg6(6),
               dg7(7), dg8(8), dg9(9), dg10(10), dg11(11),
               dg12(12), dg13(13), dg14(14), dg15(15), dg16(16),
               sod(20), com(21)}

MRZField ::= INTEGER
              {generic(0), documentCode(1), issuingState(2), personName(3),
               documentNumber(4), nationality(5), dateOfBirth(6),
               sex(7), dateOfExpiry(8), optionalData(9)}

-- Base Object Identifiers

id-icao OBJECT IDENTIFIER ::= {2 23 136 }
id-icao-mrtd OBJECT IDENTIFIER ::= {id-icao 1}
id-icao-mrtd-security OBJECT IDENTIFIER ::= {id-icao-mrtd 1}
id-icao-DeviationList OBJECT IDENTIFIER ::= {id-icao-mrtd-security 7}
id-icao-DeviationListSigningKey OBJECT IDENTIFIER ::= {id-icao-mrtd-security 8}

-- Deviation Object Identifiers and Parameter Definitions
```

```
id-Deviation-CertOrKey OBJECT IDENTIFIER ::= {id-icao-DeviationList 1}
id-Deviation-CertOrKey-DSSignature OBJECT IDENTIFIER ::= {id-Deviation-CertOrKey 1}
id-Deviation-CertOrKey-DSEncoding OBJECT IDENTIFIER ::= {id-Deviation-CertOrKey 2}
id-Deviation-CertOrKey-CSCAEncoding OBJECT IDENTIFIER ::= {id-Deviation-CertOrKey 3}
id-Deviation-CertOrKey-AAKeyCompromised OBJECT IDENTIFIER ::= {id-Deviation-CertOrKey 4}
id-Deviation-LDS OBJECT IDENTIFIER ::= {id-icao-DeviationList 2}
id-Deviation-LDS-DGMalformed OBJECT IDENTIFIER ::= {id-Deviation-LDS 1}
id-Deviation-LDS-DGHashWrong OBJECT IDENTIFIER ::= {id-Deviation-LDS 2}
id-Deviation-LDS-SODSignatureWrong OBJECT IDENTIFIER ::= {id-Deviation-LDS 3}
id-Deviation-LDS-COMInconsistent OBJECT IDENTIFIER ::= {id-Deviation-LDS 4}

id-Deviation-MRZ OBJECT IDENTIFIER ::= {id-icao-DeviationList 3}
id-Deviation-MRZ-WrongData OBJECT IDENTIFIER ::= {id-Deviation-MRZ 1}
id-Deviation-MRZ-WrongCheckDigit OBJECT IDENTIFIER ::= {id-Deviation-MRZ 2}

id-Deviation-Chip OBJECT IDENTIFIER ::= {id-icao-DeviationList 4}

id-Deviation-NationalUse OBJECT IDENTIFIER ::= {id-icao-DeviationList 5}

END
```

11. REFERENCES (NORMATIVE)

FIPS 180-2	FIPS 180-2, Federal Information Processing Standards Publication (FIPS PUB) 180-2, *Secure Hash Standard*, August 2002.
FIPS 186-4	FIPS 186-4, Federal Information Processing Standards Publication (FIPS PUB) 186-4, *Digital Signature Standard (DSS)*, July 2013 (Supersedes FIPS PUB 186-3 dated June 2009).
ISO 3166-1	ISO/IEC 3166-1: 2006, Codes for the representation of names of countries and their subdivisions — Part 1: Country Codes.
ISO/IEC 15946	ISO/IEC 15946: 2002, Information technology — Security techniques — Cryptographic techniques based on elliptic curves.
RFC 3280	RFC 3280, R. Housley, W. Polk, W. Ford, D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, April 2002.
RFC 4055	RFC 4055, J. Schaad, B. Kaliski, R. Housley, Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, June 2005.
RFC 5652	RFC 5652, R. Housley, Cryptographic Message Syntax, September 2009.
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 v 2.0, 2012.

X9.62	X9.62, Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 7 January 1999.
X.509	ITU-T X.509 | ISO/IEC 9594-8, 2008: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.
X.690	ITU-T X.690 2008: Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).
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 v2.2: RSA cryptography standard, 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 to Part 12

LIFETIMES (INFORMATIVE)

The following examples illustrate calculation of private key usage periods and public key certificate validity for various scenarios as described in Section 4.

A.1 EXAMPLE 1

The first example illustrates a scenario where eMRTDs are valid for five years. Because a relatively large number of eMRTDs are issued per day, the policy is to keep private key usage periods and public key certificate validity to a minimum. For this example, the minimum private key usage period for Document Signer certificates is one month.

Item	***Usage/Validity Period***
eMRTD validity	5 years
Document Signer private key usage period	1 month
Document Signer certificate validity	5 years + 1 month
CSCA private key usage period	3 years
CSCA certificate validity	8 years + 1 month

The consequences of this example are that by the time the first CSCA certificate becomes invalid at least 36 Document Signer certificates will have been issued (one corresponding to each private key that has a one-month usage period). In the last few months before the first CSCA certificate becomes invalid, there will be at least two additional CSCA certificates issued (one corresponding to each private key that has a three-year usage period).

A.2 EXAMPLE 2

The second example illustrates a scenario where eMRTDs are valid for ten years. The policy is to keep private key usage periods and public key certificate validity to an average length.

Item	***Usage/Validity Period***
eMRTD validity	10 years
Document Signer private key usage period	2 months
Document Signer certificate validity	10 years + 2 months

Item	*Usage/Validity Period*
CSCA private key usage period	4 years
CSCA certificate validity	14 years + 2 months

The consequences of this example are by the time the first CSCA certificate becomes invalid at least 24 Document Signer certificates will have been issued (one corresponding to each private key that has a two-month usage period). In the last few months before the first CSCA certificate becomes invalid, there will be at least three additional CSCA certificates issued (one corresponding to each private key that has a four-year usage period).

A.3 EXAMPLE 3

The final example illustrates a scenario where eMRTDs are valid for ten years, and the policy is to use the maximum private key usage periods and public key certificate validity.

Item	*Usage/Validity Period*
eMRTD validity	10 years
Document Signer private key usage period	3 months
Document Signer certificate validity	10 years + 3 months
CSCA private key usage period	5 years
CSCA certificate validity	15 years + 3 months

The consequences of this example are by the time the first CSCA certificate becomes invalid at least 20 Document Signer certificates will have been issued (one corresponding to each private key that has a three-month usage period). In the last few months before the first CSCA certificate becomes invalid, there will be at least three additional CSCA certificates issued (one corresponding to each private key that has a five-year usage period).

— — — — — — — —

Appendix B to Part 12

CERTIFICATE AND CRL PROFILE REFERENCE TEXT (INFORMATIVE)

The certificate and CRL profiles defined in Section 7 are based on definitions and base profile requirements specified in referenced documents. Brief excerpts of some relevant sections from these source documents (as of the time of writing) are replicated in the tables below. These excerpts are provided to assist the reader in understanding the background for some of the requirements specified in the eMRTD certificate and CRL profiles. They are not intended to be relied on instead of the referenced documents. In all cases, to obtain the full specification of the referenced component/extension and to obtain the most current specification, the actual referenced documents MUST be used.

Table B-1. Certificate Fields and Extensions

Component / Extension	*Reference*	*Relevant Excerpts*
`Certificate`	RFC 5280 – 4.1.1	
`TBSCertificate`	RFC 5280 – 4.1.1.1	
`signatureAlgorithm`	RFC 5280 – 4.1.1.2	
`signatureValue`	RFC 5280 – 4.1.1.3	
`TBSCertificate`	RFC 5280 – 4.1.2	
`version`	RFC 5280 – 4.1.2.1	When extensions are used, as expected in this profile, version MUST be 3 (value is `2`).
`serialNumber`	RFC 5280 – 4.1.2.2	The serial number MUST be a positive integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). CAs MUST force the `serialNumber` to be a non-negative integer. Given the uniqueness requirements above, serial numbers can be expected to contain long integers. Certificate users MUST be able to handle `serialNumber` values up to 20 octets. Conformant CAs MUST NOT use `serialNumber` values longer than 20 octets.

Component / Extension	*Reference*	*Relevant Excerpts*
	X.690 – 8.3.2	If the contents octets of an integer value encoding consist of more than one octet, then the bits of the first octet and bit 8 of the second octet: a) shall not all be ones; and b) shall not all be zero. *Note*.— These rules ensure that an integer value is always encoded in the smallest possible number of octets.
	X.690 – 8.3.3	The contents octets shall be a two's complement binary number equal to the integer value, and consisting of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the contents octets.
`signature`	RFC 5280 – 4.1.1.2	This field MUST contain the same algorithm identifier as the `signatureAlgorithm` field in the sequence `Certificate`.
`issuer`	RFC 5280 – Appendix A.1	`X520countryName ::= PrintableString (SIZE (2))` `X520SerialNumber ::= PrintableString (SIZE (1..ub-serial-number))`
	RFC 5280 – 4.1.2.4	CAs conforming to this profile MUST use either the `PrintableString` or `UTF8String` encoding of `DirectoryString`.
	ISO 3166-1	
`validity`	RFC 5280 – 4.1.2.5	Both `notBefore` and `notAfter` may be encoded as `UTCTime` or `GeneralizedTime`. CAs conforming to this profile MUST always encode certificate validity dates through the year 2049 as `UTCTime`. Certificate validity dates in 2050 or later MUST be encoded as `GeneralizedTime`.
`(if encoded as UTCTime)`	X.690 – 11.8.1	The encoding shall terminate with "Z", as described in the ITU-T X.680 | ISO/IEC 8824-1 clause on `UTCTime`.
	X.690 – 11.8.2	The seconds element shall always be present.
`(if encoded as GeneralizedTime)`	X.690 – 11.7.1	The encoding shall terminate with a "Z", as described in the ITU-T Rec. X.680 | ISO/IEC 8824-1 clause on `GeneralizedTime`.
	X.690 – 11.7.2	The seconds element shall always be present.

Component / Extension	***Reference***	***Relevant Excerpts***
	RFC 5280 – 4.1.2.5.2	`GeneralizedTime` values MUST NOT include fractional seconds. For the purposes of this profile, `GeneralizedTime` values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero.
`subject`	RFC 5280 – Appendix A.1	`X520countryName ::= PrintableString (SIZE (2))` `X520SerialNumber ::= PrintableString (SIZE (1..ub-serial-number))`
	RFC 5280 – 4.1.2.6	CAs conforming to this profile MUST use either the `PrintableString` or `UTF8String` encoding of `DirectoryString`.
`subjectPublicKeyInfo`	RFC 5280 – 4.1.2.7	
`issuerUniqueID`	RFC 5280 – 4.1.2.8	CAs conforming to this profile MUST NOT generate certificates with unique identifiers.
`subjectUniqueID`	RFC 5280 – 4.1.2.8	CAs conforming to this profile MUST NOT generate certificates with unique identifiers.
`extensions`	X.690 – 11.5	The encoding of a `set` value or `sequence` value shall not include an encoding for any component value which is equal to its default value.
`AuthorityKeyIdentifier`	RFC 5280 – 4.2.1.1	The `keyIdentifier` field of the `authorityKeyIdentifier` extension MUST be included in all certificates generated by conforming CAs to facilitate certification path construction. There is one exception. Where a CA distributes its public key in the form of a “self-signed” certificate, the authority key identifier MAY be omitted.
`keyIdentifier`		
`authorityCertIssuer`		
`authorityCertSerialNumber`		

Component / Extension	*Reference*	*Relevant Excerpts*
SubjectKeyIdentifier	RFC 5280 – 4.2.1.2	To facilitate certification path construction, this extension MUST appear in all conforming CA certificates, that is, all certificates including the basic constraints extension (section 4.2.1.9) where the value of cA is TRUE.
subjectKeyIdentifier		
KeyUsage	RFC 5280 – 4.2.1.3	The usage restriction might be employed when a key that could be used for more than one operation is to be restricted.
digitalSignature		The digitalSignature bit is asserted when the subject public key is used with a digital signature mechanism to support security services other than certificate signing (bit 5), or CRL signing (bit 6).
nonRepudiation		
keyEncipherment		
dataEncipherment		
keyAgreement		
keyCertSign		The keyCertSign bit is asserted when the subject public key is used for verifying a signature on public key certificates.
cRLSign		The cRLSign bit is asserted when the subject public key is used for verifying a signature on certificate revocation list (e.g., a CRL, delta CRL, or an ARL). This bit MUST be asserted in certificates that are used to verify signatures on CRLs.
encipherOnly		
decipherOnly		
PrivateKeyUsagePeriod	RFC 3280 – 4.2.1.4	CAs conforming to this profile MUST NOT generate certificates with private key usage period extensions unless at least one of the two components is present and the extension is non-critical.
notBefore		Where used, notBefore and notAfter are represented as GeneralizedTime and MUST be specified and interpreted as defined in section 4.1.2.5.2.
notAfter		

Component / Extension	***Reference***	***Relevant Excerpts***
`CertificatePolicies`	RFC 5280 – 4.2.1.4	If this extension is critical, the path validation software MUST be able to interpret this extension (including the optional qualifier), or MUST reject the certificate.
`PolicyInformation`		
`policyIdentifier`		
`policyQualifiers`		
`PolicyMappings`	RFC 5280 – 4.2.1.5	
`SubjectAltName`	RFC 5280 – 4.2.1.6	
`IssuerAltName`	RFC 5280 – 4.2.1.7	
`SubjectDirectoryAttributes`	RFC 5280 – 4.2.1.8	
`Basic Constraints`	RFC 5280 – 4.2.1.9	The basic constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate. Conforming CAs MUST include this extension in all CA certificates that contain public keys used to validate digital signatures on certificates and MUST mark the extension as critical in such certificates.
`cA`		The `cA` boolean indicates whether the certified public key belongs to a CA. If the `cA` boolean is not asserted, then the `keyCertSign` bit in the key usage extension MUST NOT be asserted.
`PathLenConstraint`		
`NameConstraints`	RFC 5280 – 4.2.1.10	
`PolicyConstraints`	RFC 5280 – 4.2.1.11	
`ExtKeyUsage`	RFC 5280 – 4.2.1.12	This extension indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension.

Component / Extension	*Reference*	*Relevant Excerpts*
CRLDistributionPoints	RFC 5280 – 4.2.1.13	
distributionPoint		
reasons		
cRLIssuer		
InhibitAnyPolicy	RFC 5280 – 4.2.1.14	
FreshestCRL	RFC 5280 – 4.2.1.15	
privateInternetExtensions	RFC 5280 – 4.2.2	
NameChange		
DocumentType		
Netscape Certificate Type		
other private extensions		

Table B-2. CRL Fields and Extensions

Component / Extension	*Reference*	*Relevant Excerpts*
CertificateList	RFC 5280 – 5.1.1	
tBSCertList	RFC 5280 – 5.1.1.1	
signatureAlgorithm	RFC 5280 – 5.1.1.2	
signatureValue	RFC 5280 – 5.1.1.3	
	RFC 5280 – 5.1.2	
version	RFC 5280 – 5.1.2.1	This optional field describes the version of the encoded CRL. When extensions are used, as required by this profile, this field MUST be present and MUST specify version 2 (the integer value is 1).
signature	RFC 5280 – 5.1.2.2	This field MUST contain the same algorithm identifier as the signature field in the sequence CertificateList.

Component / Extension	***Reference***	***Relevant Excerpts***
issuer	RFC 5280 – Appendix A.1	X520countryName ::= PrintableString (SIZE (2)) X520SerialNumber ::= PrintableString (SIZE 1..ub-serial-number))
	RFC 5280 – 5.1.2.3 and 4.1.2.4	CAs conforming to this profile MUST use either the PrintableString or UTF8String encoding of DirectoryString.
thisUpdate	RFC 5280 – 5.1.2.4	CRL issuers conforming to this profile MUST encode thisUpdate as UTCTime for dates through the year 2049. CRL issuers conforming to this profile MUST encode thisUpdate as GeneralizedTime for dates in the year 2050 or later.
(if encoded as UTCTime)	X.690 – 11.8.1	The encoding shall terminate with "Z", as described in the ITU-T X.680 | ISO/IEC 8824-1 clause on UTCTime.
	X.690 – 11.8.2	The seconds element shall always be present.
(if encoded as GeneralizedTime)	X.690 – 11.7.1	The encoding shall terminate with a "Z", as described in the ITU-T Rec. X.680 | ISO/IEC 8824-1 clause on GeneralizedTime.
	X.690 – 11.7.2	The seconds element shall always be present.
	RFC 5280 – 4.1.2.5.2	GeneralizedTime values MUST NOT include fractional seconds. For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero.
nextUpdate	5.1.2.5	CRL issuers conforming to this profile MUST encode nextUpdate as UTCTime for dates through the year 2049. CRL issuers conforming to this profile MUST encode nextUpdate as GeneralizedTime for dates in the year 2050 or later.
(if encoded at UTCTime)	X.690 – 11.8.1	The encoding shall terminate with "Z", as described in the ITU-T X.680 | ISO/IEC 8824-1 clause on UTCTime.
	X.690 – 11.8.2	The seconds element shall always be present.
(if encoded at GeneralizedTime)	X.690 – 11.7.1	The encoding shall terminate with a "Z", as described in the ITU-T Rec. X.680 | ISO/IEC 8824-1 clause on GeneralizedTime.
	X.690 – 11.7.2	The seconds element shall always be present.

Component / Extension	*Reference*	*Relevant Excerpts*
	RFC 5280 – 4.1.2.5.2	`GeneralizedTime` values MUST NOT include fractional seconds. For the purposes of this profile, `GeneralizedTime` values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero.
`revokedCertificates`	RFC 5280 – 5.1.2.6	When there are no revoked certificates, the revoked certificates list MUST be absent. Otherwise, revoked certificates are listed by their serial numbers.
`crlExtensions`	RFC 5280 – 5.2	Conforming CRL issuers are REQUIRED to include the authority key identifier (Section 5.2.1) and the CRL number (Section 5.2.3) extensions in all CRLs issued.
	X.690 – 11.5	The encoding of a `set` value or `sequence` value shall not include an encoding for any component value which is equal to its default value.
`authorityKeyIdentifier`	RFC 5280 – 5.2.1	Conforming CRL issuers MUST use the key identifier method, and MUST include this extension in all CRLs issued.
`issuerAlternativeName`	RFC 5280 – 5.2.2	
`cRLNumber`	RFC 5280 – 5.2.3	CRL issuers conforming to this profile MUST include this extension in all CRLs and MUST mark this extension as non-critical. `CRLNumber ::= INTEGER (0..MAX)` Given the requirements above, CRL numbers can be expected to contain long integers. CRL verifiers MUST be able to handle `CRLNumber` values up to 20 octets. Conforming CRL issuers MUST NOT use `CRLNumber` values longer than 20 octets.
	X.690 – 8.3.2	If the contents octets of an integer value encoding consist of more than one octet, then the bits of the first octet and bit 8 of the second octet: a) shall not all be ones; and b) shall not all be zero. *Note*.— These rules ensure that an integer value is always encoded in the smallest possible number of octets.

Component / Extension	***Reference***	***Relevant Excerpts***
	X.690 – 8.3.3	The contents octets shall be a two's complement binary number equal to the integer value, and consisting of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the contents octets.
`deltaCRLIndicator`	RFC 5280 – 5.2.4	
`issuingDistributionPoint`	RFC 5280 – 5.2.5	
`freshestCRL`	RFC 5280 – 5.2.6	
`reasonCode`	RFC 5280 – 5.3.1	
`holdInstructionCode`	RFC 5280 – 5.3.2	
`invalidityDate`	RFC 5280 – 5.3.3	
`certificateIssuer`	RFC 5280 – 5.3.4	

— — — — — — — —

Appendix C to Part 12

EARLIER CERTIFICATE PROFILES (INFORMATIVE)

The certificate profiles in this appendix were specified in the Sixth Edition of ICAO Doc 9303. Although CSCAs MUST issue certificates that comply with the current profiles as specified in Section 7, the earlier profiles are included here for information only as certificates that were issued in compliance with the earlier profiles will be in circulation, and processed by Inspection Systems for several years.

Table C-1. Certificate Body

Certificate Component	*Section in RFC 3280*	*Country Signing CA Certificate*	*Document Signer Certificate*	*Comments*
`Certificate`	4.1.1	m	m	
`TBSCertificate`	4.1.1.1	m	m	See Table C-2
`SignatureAlgorithm`	4.1.1.2	m	m	Value inserted here dependent on algorithm selected
`SignatureValue`	4.1.1.3	m	m	Value inserted here dependent on algorithm selected
`TBSCertificate`	4.1.2			
`version`	4.1.2.1	m	m	SHALL be `v3`
`serialNumber`	4.1.2.2	m	m	
`signature`	4.1.2.3	m	m	Value inserted here SHALL match the OID in `signatureAlgorithm`
`issuer`	4.1.2.4	m	m	
`validity`	4.1.2.5	m	m	Implementations SHALL specify using UTC time until 2049 from then on using `GeneralizedTime`
`subject`	4.1.2.6	m	m	
`subjectPublicKeyInfo`	4.1.2.7	m	m	
`issuerUniqueID`	4.1.2.8	x	x	

Certificate Component	*Section in RFC 3280*	*Country Signing CA Certificate*	*Document Signer Certificate*	*Comments*
subjectUniqueID	4.1.2.8	x	x	
extensions	4.1.2.9	m	m	See Table C-2 on which extensions SHOULD be present

Table C-2. Extensions

Extension name	*Paragraph in RFC 3280*	*Country Signing CA Certificate*	*Document Signer Certificate*	*Comments*
AuthorityKeyIdentifier	4.2.1.1	o	m	Mandatory in all certificates except for self-signed CSCA certificates
SubjectKeyIdentifier	4.2.1.2	m	o	
KeyUsage	4.2.1.3	mc	mc	This extension SHALL be marked CRITICAL
PrivateKeyUsagePeriod	4.2.1.4	o	o	This would be the issuing period of the private key
CertificatePolicies	4.2.1.5	o	o	
PolicyMappings	4.2.1.6	x	x	
SubjectAltName	4.2.1.7	x	x	
IssuerAltName	4.2.1.8	x	x	
SubjectDirectoryAttributes	4.2.1.9	x	x	
BasicConstraints	4.2.1.10	mc	x	This extension SHALL be marked CRITICAL
NameConstraints	4.2.1.11	x	x	
PolicyConstraints	4.2.1.12	x	x	
ExtKeyUsage	4.2.1.13	x	x	

Extension name	*Paragraph in RFC 3280*	*Country Signing CA Certificate*	*Document Signer Certificate*	*Comments*
CRLDistributionPoints	4.2.1.14	o	o	If issuing States or organizations choose to use this extension they SHALL include the ICAO PKD as a distribution point. Implementations may also include relative CRL DPs for local purposes; these may be ignored by other receiving States.
InhibitAnyPolicy	4.2.1.15	x	x	
FreshestCRL	4.2.1.16	x	x	
privateInternetExtensions	4.2.2	x	x	
other private extensions	N/A	o	o	If any private extension is included for national purposes then it SHALL NOT be marked. Issuing States or organizations are discouraged from including any private extensions.
AuthorityKeyIdentifier	4.2.1.1			
keyIdentifier		m	m	If this extension is used this field SHALL be supported as a minimum
authorityCertIssuer		o	o	
authorityCertSerialNumber		o	o	
SubjectKeyIdentifier	4.2.1.2			
subjectKeyIdentifier		m	m	
KeyUsage	4.2.1.3			
digitalSignature		x	m	
nonRepudiation		x	x	
keyEncipherment		x	x	
dataEncipherment		x	x	
keyAgreement		x	x	
keyCertSign		m	x	

Extension name	*Paragraph in RFC 3280*	*Country Signing CA Certificate*	*Document Signer Certificate*	*Comments*
cRLSign		m	x	
encipherOnly		x	x	
decipherOnly		x	x	
BasicConstraints	4.2.1.10			
cA		m	x	TRUE for CA certificates
PathLenConstraint		m	x	0 for New CSCA certificate, 1 for Linked CSCA certificate
CRLDistributionPoints	4.2.1.14			
distributionPoint		m	x	
reasons		m	x	
cRLIssuer		m	x	
CertificatePolicies	4.2.1.5			
PolicyInformation				
policyIdentifier		m	m	
policyQualifiers		o	o	

— — — — — — — —

Appendix D to Part 12

RFC 5280 VALIDATION COMPATIBILITY (INFORMATIVE)

This appendix provides guidance to receiving States wishing to use systems that implement the [RFC 5280] certification path and CRL validation algorithms.

The eMRTD PKI trust model is a subset of that covered by the validation procedures defined in [RFC 5280]. Section D.1 identifies the subset of steps from the [RFC 5280] definition that are required for the eMRTD application and provides the necessary inputs and initialization values and processes for certification path validation, CRL validation and revocation checking.

Section D.2 covers the remaining steps from the [RFC 5280] definition that are not relevant to the eMRTD application. The inputs and initialization values for certification path validation and CRL validation are provided. The guidance in this section is for use in situations where the tools implement the full [RFC 5280] algorithms, rather than just the subset described in D.1.

Section D.3 provides guidance to support the extension of [RFC 5280] based CRL processing to cover revocation checking after a CSCA has undergone a name change.

D.1 STEPS RELEVANT TO eMRTD

The eMRTD certification path validation procedure defined here is based on the procedure described in [RFC 5280].The same terminology and process descriptions are used. The eMRTD certificate profiles restrict certification paths to a single certificate and prohibit use of many optional features that are used in other applications, such as the Internet PKI defined in [RFC 5280]. Path validation steps associated with these features are omitted from the eMRTD certification path validation procedure.

D.1.1 Certification Path Validation Procedure

D.1.1.1 Inputs

[RFC 5280] defines a set of nine inputs to the path validation algorithm. Only the following three are relevant to the eMRTD application:

- certification path: A single certificate (e.g. the Document Signer certificate);
- current date/time; and
- Trust Anchor information, including:
    - trusted issuer name: If the Trust Anchor is in the form of a CSCA certificate, the trusted issuer name is the value of the `subject` field of that certificate;

- trusted public key algorithm: If the Trust Anchor is in the form of a CSCA certificate, the trusted public key algorithm is taken from the `SubjectPublicKeyInfo` field of that certificate;
- trusted public key: If the Trust Anchor is in the form of a CSCA certificate, the trusted public key is taken from the `SubjectPublicKeyInfo` field of that certificate; and
- trusted public key parameters: This is an optional input that is included only if the trusted public key algorithm requires parameters. If the Trust Anchor is in the form of a CSCA certificate, these parameters are taken from the `SubjectPublicKeyInfo` field of that certificate.

If an implementation requires that the additional six inputs be supplied, recommendations for these are provided in D.2.

There could be several Trust Anchors for the CSCA that issued the certificate being validated. Of these Trust Anchors, the one that MUST be used is the one that contains the public key that matches the value of the Authority Key Identifier extension in the certificate being validated.

D.1.1.2 Initialization

There are eleven State variables defined in [RFC 5280]. Only the following five are relevant to the eMRTD application:

- application: max_path_length: Initialize to “0”;
- working_issuer_name: Initialize to the value of the trusted issuer name;
- working_public_key_algorithm: Initialize to the value of the trusted public key algorithm;
- working_public_key: Initialize to the value of the trusted public key; and
- working_public_key_parameters: Initialize to the value of the trusted public key parameters.

If an implementation requires that the additional six variables be initialized, recommendations for these are provided in D.2.

D.1.1.3 Certificate processing

eMRTD certificate processing steps are a subset of those defined in [RFC 5280]. The result of processing an eMRTD certificate using this simplified process will be consistent with the result using the full RFC 5280 algorithm. If the additional inputs and State variables are configured as described in D.2:

a) Verify the basic certificate information. The certificate MUST satisfy each of the following:

- the signature on the certificate can be verified using working_public_key_algorithm, the working_public_key, and the working_public_key_parameters;
- the certificate validity period includes the current time;
- at the current time, the certificate is not revoked (see 6.3 for details); and
- the certificate issuer name is the working_issuer_name.

b) Assign the certificate `subjectPublicKey` to working_public_key.

c) If the `subjectPublicKeyInfo` field of the certificate contains an algorithm field with non-null parameters, assign the parameters to the working_public_key_parameters variable. If the `subjectPublicKeyInfo` field of the certificate contains an algorithm field with null parameters or parameters are omitted, compare the certificate `subjectPublicKey` algorithm to the working_public_key_algorithm. If the certificate `subjectPublicKey` algorithm and the working_public_key_algorithm are different, set the working_public_key_parameters to null.

d) Assign the certificate `subjectPublicKey` algorithm to the working_public_key_algorithm variable.

e) Recognize and process any other critical extensions present in the certificate.

f) Process any other recognized non-critical extensions present in the certificate.

If any of the checks in step a) fail or if there are any unrecognized critical extensions in the certificate that cannot be processed, the path validation procedure fails. Otherwise the procedure succeeds.

D.1.1.4 Outputs

If path validation succeeds, the procedure terminates, returning a success indication together with the working_public_key, the working_public_key_algorithm, and the working_public_key_parameters.

If path validation fails, the procedure terminates, returning a failure indication and an appropriate reason.

D.1.2 CRL Validation and Revocation Checking

The CRL validation algorithm in [REC 5280] covers various types of CRLs including delta CRLs, partitioned CRLs, indirect CRLs, etc. The CRL profile for the eMRTD application is very restrictive and prohibits use of any of these features. Use of the `issuingDistributionPoint` extension as well as all of the standardized CRL-entry extensions is also prohibited. As a result, CRL validation and revocation checking for the eMRTD application is relatively simple.

D.1.2.1 Inputs

[RFC 5280] defines two inputs to the CRL validation algorithm. Only the following one of these is relevant to the eMRTD application. If an implementation requires that the additional input be supplied, a recommendation for this is provided in D.2.

- certificate: certificate serial number and issuer name

D.1.2.2 Initialization

There are three State variables defined in [RFC 5280]. Only the following one of these is relevant to the eMRTD application. If an implementation requires that the additional two variables be initialized, recommendations for these are provided in D.2.

- cert_status : initialize to the value UNREVOKED.

D.1.2.3 CRL Processing

All CRLs in the eMRTD application are complete CRLs that cover all current certificates issued by the CSCA that issued the CRL. There are no partitioned, delta or indirect CRLs. The steps in the CRL processing algorithm for the eMRTD application are:

a) Obtain the current CRL for the CSCA that issued the certificate. If the CRL cannot be obtained, the cert_status variable is set to UNDETERMINED, and processing is stopped.

b) Verify that the CRL issuer is the same CSCA that issued the certificate in question. Because there is a single CSCA in each country, and the eMRTD application is a closed application with Inspection Systems retaining a cache of CRLs that is unique to this application, verifying that the country name is the same in the issuer field of the CRL and the issuer field of the certificate is sufficient.

- If the CSCA has not undergone a name change since the certificate was issued, the issuer field in the CRL and the issuer field in the certificate will be identical.

- If the CSCA has undergone a name change since the certificate was issued, the country attribute of the name in the issuer field of the certificate and in the issuer field of the CRL will be the same, but some other attributes may be different.

- If the relying party wishes to verify that substitution of some non eMRTD CRL has not happened, it may optionally verify that it has Trust Anchors for both CSCA names and that those Trust Anchors are for the same CSCA. If the CSCA has undergone a name change and has included the optional `issuerAltName` extension in the CRL, the relying party MAY optionally verify that the issuer field in the certificate is identical to one of the values in this extension.

If the CRL issuer is not the CSCA that issued the certificate, the cert_status variable is set to UNDETERMINED, and processing is stopped.

c) Validate the certification path for the issuer of the CRL. Note that in the eMRTD application all CRLs are issued by CSCAs that are the Trust Anchors for the respective paths. Unlike the algorithm in [RFC 5280], the eMRTD application does NOT require that the Trust Anchor used to validate the CRL certification path be the same Trust Anchor that was used to validate the target certificate. However, if the Trust Anchors are different, they MUST both be Trust Anchors for the same CSCA. Unlike [RFC 5280], the eMRTD application has multiple Trust Anchors for a given CSCA that are valid at the same time. If the certification path cannot be successfully validated, the cert_status variable is set to UNDETERMINED, and processing is stopped.

d) Verify the signature on the CRL. If the signature cannot be successfully verified, the cert_status variable is set to UNDETERMINED, and processing is stopped.

e) Search for the certificate on the CRL. If an entry is found that matches the certificate issuer and serial number, then the cert_status variable is set to UNSPECIFIED.

D.1.2.4 Output

Return the cert_status. If steps a), b), c) or d) failed, the status will be UNDETERMINED. If the certificate was listed as revoked on the CRL, the status will be UNSPECIFIED. If CRL validation succeeded, but the certificate was not listed on the CRL, the status will be UNREVOKED.

D.2 STEPS NOT REQUIRED BY eMRTD

D.2.1 Certification Path Validation

Settings for additional inputs that are not relevant to eMRTD validation include:

- initial-policy-mapping-inhibit: Set to inhibit policy mapping;
- initial-any-policy-inhibit: Set to inhibit processing of the any-policy value;
- initial-permitted-subtrees: Set to permit all subtrees;
- initial-excluded-subtrees: Set to exclude no subtrees;
- initial-explicit-policy: This should NOT be set; and
- user-initial-policy-set: Set to the special value “any-policy”.

Initialization of State variables that are not relevant to the eMRTD application include:

- permitted_subtrees: Initialize to permit all subtrees;
- excluded_subtrees: Initialize to exclude no subtrees;
- inhibit_any_policy: If initial-any-policy-inhibit is set, initialize to “0”. Otherwise, set to the value 1 or any value greater than that;
- policy_mapping: Initialize to “0”;
- explicit_policy: Initialize to “2”; and
- valid_policy_tree: Initialize the valid_policy element to “anyPolicy”, the qualifier_set element to empty and the expected_policy_set to “anyPolicy”.

D.2.2 CRL Validation

Settings for additional inputs that are not relevant to eMRTD validation include:

- use-deltas: Set to prohibit use of deltas.

Initialization of State variables that are not relevant to the eMRTD application include:

- reasons_mask: Initialize to an empty set; and
- Interim_reasons_mask: Initialize to the special value “all-reasons”.

D.3 MODIFICATIONS REQUIRED TO PROCESS CRLS

CRL validation systems that comply with the CRL validation procedure in [RFC 5280] are not intended to support environments where a CA has undergone a name change, such as the eMRTD application environment. Therefore these systems require some modification to handle this special case, as described below:

a) In clause 6.3.3, step a) of the [RFC 5280] CRL validation procedure, the name in the distribution point field of the CRL Distribution Points extension of the certificate in question is used to update the local cache with the relevant CRL(s). For the eMRTD application, this step would need to be modified and only the `countryName` attribute of the distribution point field should be used to identify and obtain the appropriate CRL.

b) In clause 6.3.3, step f) of the [RFC 5280] CRL validation procedure, there is a requirement that the same Trust Anchor be used to validate the certification path for the CRL issuer that was used to validate the target certificate. This is NOT a requirement for the eMRTD application because independent Trust Anchors are established for each public key of the CSCA.

The Trust Anchor used for validation of the CRL issuer will be the one for the CSCA's public key that corresponds to the private key used to sign the CRL. The Trust Anchor used to validate the certification path for the target certificate may be for an earlier CSCA key pair.

— — — — — — — —

Appendix E to Part 12

LDS2 EXAMPLE (INFORMATIVE)

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.

The preliminaries are as follows:

- Utopia has installed an LDS2 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 Utopia and Dystopia 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 Utopia 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.

The recurring process in order to enable Dystopia to electronically stamp Utopia's eMRTDs is as follows:

- 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.

The process to electronically stamp an eMRTD is as follows:

- 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. If 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.

— END —

ISBN 978-92-9275-422-8