For Publication on the ICAO Website

TECHNICAL REPORT

LDS2 - Protocols

Version 0.8

April 2017

File: 2017_04_27 LDS2 – Protocols.odt
Author: ISO/IEC JTC1 SC17 WG3/TF5

Technical Report

LDS2 – Protocols

Release : 0.6
Date : 27 April 2017

Release Control

Release	*Date*	*Description*

Table of Contents

1 INTRODUCTION....4

2 ASSUMPTIONS AND NOTATIONS....4

3 SECURING ELECTRONIC DATA....4

4 ACCESS TO THE CONTACTLESS IC....4

4.1 Compliant Configurations....4
4.2 Chip Access Procedure....4
4.4 Password Authenticated Connection Establishment....6

5 AUTHENTICATION OF DATA....9

5.1 Passive Authentication....9

6 AUTHENTICATION OF THE CONTACTLESS IC....9

6.2 Chip Authentication....9

7 ADDITIONAL ACCESS CONTROL MECHANISMS....9

7.1 Terminal Authentication....9

8 INSPECTION SYSTEM....19

8.6 Terminal Authentication....19

9 COMMON SPECIFICATIONS....20

9.2 Information on Supported Protocols....20
9.3 APDUs....21
9.4 Public Key Data Objects....22

1 Introduction

This document extends the Inspection Procedure for eMTRDs by Terminal Authentication to facilitate granting certificate based access rights to terminals. This enables Issuing States or organizations to grant (and revoke) access rights, e.g. read access to sensitive data or write access to the chip, on a per-country basis. The protocol is based on Extended Access Control as used in the European Union to protect access to fingerprints stored in EF.DG3 of the eMRTD application, cf. [TR-03110].

Additionally, Chip Authentication is extended to allow execution in the Master File to enable a compact inspection procedure for eMRTDs with multiple applications.

[This document uses the terminology from Doc 9303 7th edition.]

2 Assumptions and Notations

[Add the following bullets to the list in 2.2]

- Signing a message *m* with private key SK_{PCD} is denoted by *s* = **Sign***(SK_{PCD}, m)*.
- Verifying the resulting signature *s* with public key PK_{PCD} is denoted by **Verify***(PK_{PCD}, s, m)*.

3 Securing Electronic Data

[Before Table 1, add the sentence:]

Inspection Procedures for different configurations of eMRTDs are described in Appendix A.

[In Table 1, Rename the row “Extended Access Control” to “Terminal Authentication” and replace “additional biometrics” by “sensitive data” in the description.]

4 Access to the Contactless IC

4.1 Compliant Configurations

[Add the following note at the end of 4.1:]

Note: For access to applications other than the eMRTD application, the IC MUST require the execution of PACE.

4.2 Chip Access Procedure

[Amend section 4.2 from Doc9303-11 as follows (changes in *italics* for readability)]

The chip access procedure to authenticate the inspection system consists of the following steps. If PACE is not supported by the inspection system, steps 1 and 3 are skipped.

1. **Read EF.CardAccess** **(REQUIRED)**

   If PACE is supported by the eMRTD, the eMRTD chip MUST provide the parameters to be used for PACE in the file EF.CardAccess.

If EF.CardAccess is available, the inspection system SHALL read the file EF.CardAccess (cf. Section 9.2.8) to determine the parameters (i.e. symmetric ciphers, key agreement algorithms, domain parameters, and mappings) supported by the eMRTD chip. The inspection system may select any of those parameters.

If the file EF.CardAccess is not available or does not contain parameters for PACE, the inspection system SHOULD try to read the eMRTD with Basic Access Control (skip to Step *3*).

2. Read EF.DIR (OPTIONAL)

The Inspection System MAY read EF.DIR (if present) to retrieve a list of applications present on the eMRTD chip.

3. PACE (CONDITIONAL)

This step is RECOMMENDED if PACE is supported by the eMRTD chip. *This step is REQUIRED if access to applications other than the eMRTD application is intended.*

- The inspection system SHOULD derive the key K_{π} from the MRZ. It MAY use the CAN instead of the MRZ if the CAN is known to the inspection system.
- The eMRTD chip SHALL accept the MRZ as passwords for PACE. It MAY additionally accept the CAN.
- The inspection system and the eMRTD chip mutually authenticate using K_{π} and derive session keys KS_{Enc} and KS_{MAC}. The PACE protocol as described in Section 4.4 SHALL be used.

If successful, the eMRTD chip performs the following:

- It SHALL start Secure Messaging.
- It SHALL grant access to less-sensitive data (e.g. EF.DG1, EF.DG2, EF.DG14, EF.DG15, etc. *of the eMRTD application* and the Document Security Object – for the definition of “sensitive data” see Doc 9303-1).
- It SHALL restrict access rights to require Secure Messaging.

4. Basic Access Control (CONDITIONAL)

This step is REQUIRED if Chip Access Control is enforced by the eMRTD chip and PACE has not been used. If PACE was successfully performed or if the eMRTD does not enforce Chip Access Control, this step is skipped.

The eMRTD Application MUST be selected before Basic Access Control is performed.

- The inspection system SHOULD derive the Document Basic Access Keys (K_{Enc} and K_{MAC}) from the MRZ.

- The inspection system and the eMRTD chip mutually authenticate using the Document Basic Access Keys and derive session keys KS_{Enc} and KS_{MAC}.

If successful, the eMRTD chip performs the following:

- It SHALL start Secure Messaging.
- It SHALL grant access to less-sensitive data (e.g. EF.DG1, EF.DG2, EF.DG14, EF.DG15, etc. *of the eMRTD application* and the Document Security Object).
- It SHALL restrict access rights to require Secure Messaging.

The inspection system MUST verify the authenticity of the contents of the file EF.CardAccess *and EF.DIR* (see above) using EF.DG14 *or EF.CardSecurity*.

Note: As a result of the Chip Access Procedure, the Current DF can be either the Master File (if PACE was used) or the eMRTD Application (if BAC was used).

4.4 Password Authenticated Connection Establishment

4.4.2 Application Protocol Data Units

[Amend section 4.4.4 from Doc 9303-11 as follows (changes in *italics* for readability)]

The following sequence of commands SHALL be used to implement PACE:

1. MSE:Set AT
2. General Authenticate

4.4.2.1 MSE:Set AT

The command MSE:Set AT is used to select and initialize the PACE protocol.

Command			
CLA		Context specific	
INS	0x22	Manage Security Environment	
P1/P2	0xC1A4	Set Authentication Template for mutual authentication	
Data	0x80	*Cryptographic mechanism reference* Object Identifier of the protocol to select (value only, Tag 0x06 is omitted).	REQUIRED
	0x83	*Reference of a public key / secret key* *T*he password to be used is indicated as follows: 0x01: MRZ_information 0x02: CAN	REQUIRED
	0x84	*Reference of a private key / Reference for computing a session key*	CONDITIONAL

		This data object is REQUIRED to indicate the identifier of the domain parameters to be used if the domain parameters are ambiguous, i.e. more than one set of domain parameters is available for PACE.	
	0x7F4C	*Certificate Holder Authorization Template* *This data object (defined in Doc 9303-12) MUST be present if the terminal requests Certification Authority Reference(s) for use in Terminal Authentication to be returned as part of PACE (cf. Section 4.4.3).* *The Object Identifier contained in this data object SHALL be set to* `id-IS`*. The access bits in the discretionary data template SHALL all be set to 1 by the terminal.*	*CONDITIONAL*

Note: For the MSE:Set command, the IC SHOULD ignore data objects with tags not specified for this command. The terminal SHOULD NOT include data objects with tags not known to be understood by the IC.

4.4.3 Exchanged Data

[Amend section 4.4.5 from Doc 9303-11 as follows (changes in *italics* for readability)]

The protocol specific data objects SHALL be exchanged in a chain of General Authenticate commands, with protocol specific command and response data encapsulated in a Dynamic Authentication data object (see section 4.4.4.2) with context specific tags as shown in the table below:

Table 1: Exchanged data for PACE

Step	***Description***	***Protocol Command Data***		***Protocol Response Data***	
1.	Encrypted Nonce	-	Absent[1]	0x80	Encrypted Nonce
2.	Map Nonce	0x81	Mapping Data	0x82	Mapping Data
3.	Perform Key Agreement	0x83	Ephemeral Public Key	0x84	Ephemeral Public Key
4.	Mutual Authentication	0x85	Authentication Token	0x86	Authentication Token
				0x87	*Certification Authority Reference (CONDITIONAL)*
				0x88	*Certification Authority Reference (CONDITIONAL)*
				0x8A	Encrypted Chip Authentication Data (CONDITIONAL)

- *Certification Authority Reference(s) MUST be present if a data object 0x7F4C was transmitted to the IC during set up of PACE (cf. Section 4.4.2.1) and Terminal Authentication is supported by the IC. In this case the data object 0x87 SHALL*

[1] This implies an empty Dynamic Authentication Data Object

contain the most recent Certification Authority Reference. The data object 0x88 MAY contain the previous Certification Authority Reference.

- Encrypted Chip Authentication Data (cf. Section 4.4.3.5) MUST be present if Chip Authentication Mapping is used and MUST NOT be present otherwise.

4.4.3.5 Certification Authority Reference

[Add before the current section 4.4.5.5]

The Certification Authority Reference (CAR) data objects SHALL be encoded as specified in Doc 9303-12.

5 Authentication of Data

5.1 Passive Authentication

[Add new section as follows]

5.1.2 Inspection Process for LDS2-data

Data written after issuance of the eMRTD are not protected by the Document Security Object, which is signed by the issuer of the document. To verify the authenticity of data written after issuance, the following steps MUST be performed by the inspection system for each written data object:

1. The inspection system SHALL build and validate a certification path from a Trust Anchor to the Signer Certificate used to sign the data object according to Doc 9303-12. The inspection system MAY use both certificates known beforehand and certificates retrieved from the chip to build the path (see Doc 9303-10).
2. The inspection system SHALL use the verified Signer Public Key to verify the signature of the data object.

Note: This procedure can be skipped for data objects whose authenticity is not relevant for the inspection process.

6 Authentication of the Contactless IC

6.2 Chip Authentication

[Replace last paragraph “as the eMRTD application … application” by the following]

If the IC supports Chip Authentication, the IC MAY support Chip Authentication in the Master File and/or MAY support Chip Authentication in the eMRTD application. If Chip Authentication is used in conjunction with accessing data groups in other applications than the eMRTD application, the IC MUST support Chip Authentication in the Master File.

Note: If compatibility with European Union Extended Access Control [TR-03110] is required, the IC MUST support Chip Authentication in the eMRTD application.

7 Additional Access Control Mechanisms

[rename 7.1 “Extended Access Control for Additional Biometrics” to “Terminal Authentication” and replace with the following text]

7.1 Terminal Authentication

The Terminal Authentication Protocol is a two move challenge-response protocol that provides explicit unilateral authentication of the terminal. The protocol is based on Extended Access Control as specified in [TR-03110]. If this protocol is supported by the IC, it MUST support Chip Authentication or PACE with Chip Authentication Mapping.

This protocol enables the IC to verify that the terminal is entitled to access sensitive data. As the terminal may access sensitive data afterwards, all further communication MUST be protected appropriately. Terminal Authentication therefore also authenticates an ephemeral public key chosen by the terminal that was used to set up Secure Messaging with Chip Authentication or PACE with Chip Authentication Mapping. The IC MUST bind the terminal's access rights to Secure Messaging established by the authenticated ephemeral public key of the terminal.

The IC MAY support Terminal Authentication in the Master File and/or the eMRTD application. If Terminal Authentication is used to protect data groups in other applications than the eMRTD application, the IC MUST support Terminal Authentication in the Master File.

Note: If compatibility with European Union Extended Access Control [TR-03110] is required, the IC MUST support Terminal Authentication in the eMRTD application.

7.1.2 Protocol Specification

The following steps are performed by the terminal and the IC:

1. The terminal sends a certificate chain to the IC. The chain starts with a certificate verifiable with the CVCA public key stored on the chip and ends with the Terminal Certificate.
2. The IC verifies the certificates and extracts the terminal's public key PK_{PCD}.
3. The IC randomly chooses a challenge r_{IC} and sends it to the terminal.
4. The terminal responds with the signature s_{PCD} = **Sign**$(SK_{PCD}, ID_{IC}||r_{IC}||$**Comp**$(PK_{DH,PCD}))$.
5. The IC checks that **Verify**$(PK_{PCD}, s_{PCD}, ID_{IC}||r_{IC}||$**Comp**$(PK_{DH,PCD}))$ = *true*.

Note: The key $PK_{DH,PCD}$ is generated during Chip Authentication or PACE with Chip Authentication Mapping. If more than one key is generated (e.g. Chip Authentication is performed after PACE with Chip Authentication Mapping), the newest key MUST be used.

In this protocol ID_{IC} is an identifier of the IC:

- If BAC is used ID_{IC} is the eMRTD's Document Number as contained in the MRZ including the check digit.
- If PACE is used, ID_{IC} is computed using the IC's ephemeral PACE public key, i.e. ID_{IC} = **Comp**$(PK_{DH,IC})$.

Note: A successful execution of the PACE protocol is REQUIRED before Terminal Authentication can be performed in the MF.

A simplified version is shown below:

IC (chip)		***PCD (inspection system)***
Choose r_{IC} randomly		
	–- r_{IC} →	
		s_{PCD} = **Sign**$(SK_{PCD}, ID_{IC} \| \| r_{IC} \| \|$ **Comp**$(PK_{DH,PCD}))$
	←- s_{PCD} –-	
Verify$(PK_{PCD}, s_{PCD},$ $ID_{IC} \| \| r_{IC} \| \|$ **Comp**$(PK_{DH,PCD})) = true$		

Figure 1: Terminal Authentication

7.1.3 Security Status

If Terminal Authentication was successfully performed, the IC SHALL grant access to stored sensitive data according to the effective authorization of the authenticated terminal. If the effective authorization does not grant access rights to an application, selecting this application MUST be rejected by the IC.

The IC SHALL however restrict the terminal's access rights to Secure Messaging established by the authenticated ephemeral public key, i.e. the ephemeral public key provided by the terminal as part of Chip Authentication or PACE with Chip Authentication Mapping. The IC MUST NOT accept more than one execution of Terminal Authentication within the same session (cf. Section 9.8 of Doc 9303-11 on the definition of "session").

Note: Access rights are valid as long as the Secure Messaging established by the authentivated ephemeral public keys is active, therefore the security status is not affected by selecting or deselecting applications.

Note: Secure Messaging is not affected by Terminal Authentication. The MRTD chip SHALL retain Secure Messaging even if Terminal Authentication fails (unless a Secure Messaging error occurs).

7.1.4 Cryptographic Specifications

7.1.4.1 Terminal Authentication with RSA

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

7.1.4.1.1 Signature Algorithm

RSA [RFC-RSA], [PKCS#1] as specified in Table 2 SHALL be used.

Table 2: Object Identifiers for Terminal Authentication with RSA

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

id-TA-RSA-PSS-SHA-512	RSASSA-PSS	SHA-512	default

The default parameters to be used with RSA-PSS are defined as follows:

- Hash Algorithm: The hash algorithm is selected according to Table 2.
- Mask Generation Algorithm: MGF1 [RFC-RSA], [PKCS#1] using the selected hash algorithm.
- Salt Length: Octet length of the output of the selected hash algorithm.
- Trailer Field: 0xBC

7.1.4.1.2 Public Key Format

The TLV-Format [7816-8] as described in [TODO: Reference to PKI-part] SHALL be used.

- The object identifier SHALL be taken from Table 2.
- The bit length of the modulus SHALL be 2048, or 3072.
- The bit length of the exponent SHALL be at most 32.

7.1.4.2 Terminal Authentication with ECDSA

For Terminal Authentication with ECDSA the following algorithms and formats MUST be used.

7.1.4.2.1 Signature Algorithm

ECDSA with plain signature format [TR-03111] as specified in Table 3 SHALL be used.

Table 3: Object Identifiers for Terminal Authentication with ECDSA

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

7.1.4.2.2 Public Key Format

The TLV-Format [7816-8] as described in [TODO Reference to PKI part] SHALL be used.

- The object identifier SHALL be taken from Table 3.
- The bit length of the curve SHALL be 224, 256, 320, 384 or 512.
- Domain Parameters SHALL be compliant to [TR-03111].

7.1.4.3 Certificate Validation

To validate a Terminal Certificate, the IC MUST be provided with a certificate chain starting at a trust-point stored on the IC. Those trust-points are more or less recent public keys of the IC's CVCA.

7.1.4.3.1 Initial State of the IC's trust-point(s)

The initial trust-point(s) SHALL be stored securely in the IC'S memory in the production or (pre-) personalization phase.

The (pre-)personalization agent SHALL

- set the current date of the IC to the date of the (pre-)personalization, and
- personalize the CVCA key with the most recent effective date as trust-point.

The (pre-)personalization agent MAY additionally personalize the previous CVCA key as trust-point.

7.1.4.3.2 Link Certificates

As the key pair used by the CVCA changes over time, CVCA Link Certificates have to be produced. CVCA Link Certificates MUST be signed with the previous CVCA key, i.e. the CVCA key with the most recent effective date. The IC is REQUIRED to internally update its trust-point(s) according to received valid link certificates.

The IC MUST be able to store up to two trust-points.

Note: Due to the scheduling of CVCA Link Certificates (see Doc 9303-12), at most two trust-points need to be stored on the IC.

7.1.4.3.3 Current Date

The IC MUST accept expired CVCA Link Certificates but it MUST NOT accept expired DV and Terminal Certificates. To determine whether a certificate is expired, the IC SHALL use its *current date*.

Current Date: If the IC has no internal clock, the current date of the IC SHALL be approximated as described in the following. The date is autonomously approximated by the IC using the most recent certificate effective date contained in a valid CVCA Link Certificate, a DV Certificate or an *Accurate Terminal Certificate*.

Accurate Terminal Certificate: A Terminal Certificate is accurate, if the issuing Document Verifier is trusted by the IC to produce Terminal Certificates with the correct certificate effective date. CVCA Link Certificates, DV Certificates and Terminal Certificates issued by a domestic DV SHALL be considered accurate by the IC. Other certificates MUST NOT be considered accurate.

A terminal MAY send CVCA Link Certificates, DV Certificates, and Terminal Certificates to an IC to update the current date and the trust-point stored on the IC even if the terminal does not intend to or is not able to continue with Terminal Authentication.

Note: The IC only verifies that a certificate is apparently recent (i.e. with respect to the approximated current date), unless the IC contains an internal clock.

7.1.4.3.4 General Validation Procedure

The certificate validation procedure consists of three steps:

1. **Certificate Verification:** The signature MUST be valid and unless the certificate is a CVCA Link Certificate, the certificate MUST NOT be expired. If the verification fails, the procedure SHALL be aborted.

Note: The case of an expired CVCA Link Certificate can only occur if the IC has a source of time beyond the approximated current date described above.

2. **Internal Status Update:** The current date MUST be *updated*, the public key and the attributes (including relevant certificate extensions) MUST be imported, new trust-points MUST be *enabled*, expired trust-points MUST be *disabled* for the verification of DV Certificates.
3. **Cleanup:** The chip SHALL provide at most two enabled trust-points per application. If more than two trust-points for an application remain enabled after the internal status update, the trust-point with the least recent effective date SHALL be *disabled*.

The operation of *updating* the current date and the operations of *enabling* and *disabling* a trust-point MUST be implemented as an atomic operation.

Enabling a trust-point: The new trust-point SHALL be added to the list of trust-points.

Disabling a trust-point: Expired trust-points MUST NOT be used for the verification of DV Certificates. In case of ICs where the current date may be advanced beyond the expiry date of a trust-point, e.g. ICs using an internal clock, expired trust-points MUST remain usable for the verification of CVCA Link Certificates. Disabled trust-points MAY be deleted after the successful import of the successive Link Certificate.

7.1.4.3.5 Example Validation Procedure

The following validation procedure, provided as an example, MAY be used to validate a certificate chain. For each received certificate the IC performs the following steps:

1. The IC verifies the signature on the certificate. If the signature is incorrect, the verification fails.
2. If the certificate is not a CVCA Link Certificate, the certificate expiration date is compared to the IC's current date. If the expiration date is before the current date, the verification fails.
3. The certificate is accepted as valid and the public key and the attributes (including relevant certificate extensions) contained in the certificate are imported.
   - For CVCA, DV, and Accurate Terminal Certificates: The certificate effective date is compared to the IC’s current date. If the current date is before the effective date, the current date is updated to the effective date.
   - For CVCA Link Certificates: The new CVCA public key is added to the list of trust-points stored securely in the IC’s memory. The new trust-point is then enabled.
   - For DV and Terminal Certificates: The new DV or terminal public key is temporarily imported for subsequent certificate verification or Terminal Authentication, respectively.
4. Expired trust-points stored securely in the IC’s memory are disabled for the verification of DV Certificates and may be removed from the list of trust-points.

7.1.4.3.6 Effective Authorization

Each certificate SHALL contain a *Certificate Holder Authorization Template* ([REF to PKI-Part]) and MAY contain *Authorization Extensions* ([REF to PKI-Part]).

- The Certificate Holder Authorization Template identifies the terminal type (this specification only considers Inspection Systems, but other specifications may use different terminal types).
- The Certificate Holder Authorization Template and the Authorization Extensions determine the *relative authorization* of the certificate holder assigned by the issuing certificate authority.

To determine the *effective authorization* of a certificate holder, the IC MUST calculate a bitwise Boolean 'and' of the relative authorization contained in the Terminal Certificate, the referenced Document Verifier Certificate, and the referenced CVCA Certificate.

The effective authorization SHALL be interpreted by the IC as follows:

- The effective role is a CVCA:
  - This link certificate was issued by the national CVCA.
  - The IC MUST update its internal trust-point, i.e. the public key and the effective authorization.
  - The certificate issuer is a trusted source of time and the IC MUST update its current date using the Certificate Effective Date.
  - The IC MUST NOT grant the CVCA access to sensitive data (i.e. the effective authorization SHOULD be ignored).
- The effective role is a DV:
  - The certificate was issued by the national CVCA for an authorized DV.
  - The certificate issuer is a trusted source of time and the IC MUST update its current date using the Certificate Effective Date.
  - The IC MUST NOT grant a DV access to sensitive data (i.e. the effective authorization SHOULD be ignored).
- The effective role is a Terminal:
  - The certificate was issued by either a domestic or a foreign DV.
  - If the certificate is an accurate terminal certificate (cf. Section 7.1.4.3.3), the issuer is a trusted source of time and the IC MUST update its current date using the Certificate Effective Date.
  - The IC MUST grant the authenticated terminal access to sensitive data according to the effective authorization.

Note: The Certificate Holder Authorization Template and the Authorization Extensions can contain bits not allocated to an access right (RFU bits). The IC MUST ignore these bits during evaluation of access rights.

7.1.4.3.7 Public Key Import

Public keys imported by the certificate validation procedure are either *permanently* or *temporarily* stored on the IC.

The IC SHOULD reject to import a public key, if the Certificate Holder Reference is already known to the IC.

Permanent Import

Public keys contained in CVCA Link Certificates SHALL be permanently imported by the IC and MUST be stored securely in the IC's memory. A permanently imported public key and its metadata SHALL fulfill the following conditions:

- It MAY be overwritten *after expiration* by a subsequent permanently imported public key.
- It either MUST be overwritten by a subsequent permanently imported public key with the same Certificate Holder Reference or the import MUST be rejected.
- It MUST NOT be overwritten by a temporarily imported public key.

Enabling and disabling a permanently imported public key MUST be an atomic operation.

Temporary Import

Public keys contained in DV and Terminal Certificates SHALL be temporarily imported by the IC. A temporarily imported public key and its metadata SHALL fulfill the following conditions:

- It SHALL NOT be selectable or usable after a power down of the IC.
- It MUST remain usable until the subsequent cryptographic operation is successfully completed (i.e. PSO:Verify Certificate or External Authenticate).
- It MAY be overwritten by a subsequent temporarily imported public key.

A terminal MUST NOT make use of any temporarily imported public key but the most recently imported.

Imported Metadata

For each permanently or temporarily imported public key the following additional data contained in the certificate (see Doc 9303-12) MUST be stored:

- Certificate Holder Reference
- Certificate Holder Authorization (effective role and effective authorization)
- Certificate Effective Date
- Certificate Expiration Date
- Certificate Extensions (where applicable)

The calculation of the effective role (CVCA, DV, or Terminal) and the effective authorization of the certificate holder is described in Section 7.1.4.3.6.

Note: The format of the stored data is operating system dependent and out of the scope of this specification.

7.1.5 Application Protocol Data Units

The following sequence of commands SHALL be used with secure messaging to implement Terminal Authentication:

1. MSE:Set DST
2. PSO:Verify Certificate
3. MSE:Set AT
4. Get Challenge
5. External Authenticate

Steps 1 and 2 are repeated for every CV certificate to be verified (CVCA Link Certificates, DV Certificate, Terminal Certificate).

7.1.5.1 MSE:Set DST

The command MSE:Set DST is used to setup certificate verification.

Command			
CLA		Context Specific	
INS	0x22	Manage Security Environment	
P1/P2	0x81B6	Set Digital Signature Template for verification.	
Data	0x83	*Reference of a public key* ISO 8859-1 encoded name of the public key to be set	REQUIRED

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal Operation* The key has been selected for the given purpose.
	0x6A88	*Referenced data not found* The selection failed as the public key is not available.
	other	*Operating system dependent error* The key has not been selected.

Note: Some operating systems accept the selection of an unavailable public key and return an error only when the public key is used for the selected purpose.

7.1.5.2 PSO:Verify Certificate

The command PSO:Verify Certificate is used to verify and import certificates.

Command			
CLA		Context Specific	
INS	0x2A	Perform Security Operation	
P1/P2	0x00BE	Verify self-descriptive certificate.	
Data	0x7F4E	*Certificate body* The body of the certificate to be verified.	REQUIRED
	0x5F37	*Signature* The signature of the certificate to be verified.	REQUIRED

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal operation* The certificate was successfully validated and the public key has been imported.
	other	*Operating system dependent error* The public key could not be imported (e.g. the certificate was not accepted).

7.1.5.3 MSE:Set AT

Command			
CLA		Context Specific	
INS	0x22	Manage Security Environment	
P1/P2	0x81A4	*Terminal Authentication:* Set Authentication Template for external authentication.	
Data	0x83	*Reference of a public key / secret key* This data object is used to select the public key of the terminal by its ISO 8859-1 encoded name.	REQUIRED

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal operation* The protocol has been selected and initialized.
	0x6A80	*Incorrect parameters in the command data field* Algorithm not supported or initialization failed.
	0x6A88	*Referenced data not found* The referenced data is not available.
	other	*Operating system dependent error* The initialization of the protocol failed.

Note: Some operating systems accept the selection of an unavailable public key and return an error only when the public key is used for the selected purpose.

7.1.5.4 Get Challenge

Command			
CLA		Context Specific	
INS	0x84	Get Challenge	
P1/P2	0x0000		
Data	–	Absent	
Le	0x08		REQUIRED

Response		
Data	r_{IC}	8 bytes of randomness.
Status Bytes	0x9000	*Normal operation*
	other	*Operating system dependent error*

7.1.5.5 External Authenticate

Command			
CLA		Context Specific	
INS	0x82	External Authenticate	
P1/P2	0x0000	Keys and Algorithms implicitly known.	
Data		Signature generated by the terminal.	REQUIRED

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal operation* The authentication was successful. Access to data groups will be granted according to the effective authorization of the corresponding verified certificate.
	0x6300	*Warning* Signature verification failed.
	0x6982	*Security status not satisfied* The authentication failed as the current authentication level of the terminal does not allow to use Terminal Authentication (e.g. Terminal Authentication was already performed, etc.).
	other	*Operating system dependent error* The authentication failed.

8 Inspection System

8.6 Terminal Authentication

[rename Section 8.6 “Extended Access Control for Additional Biometrics” to “Terminal Authentication” and replace with the following]

Support of Terminal Authentication by inspection systems is OPTIONAL.

If the inspection system supports Terminal Authentication, it is REQUIRED that the inspection system has the capablity to securely store the private key of the inspection system. The inspection system requires access to its DV in regular intervals to renew the terminal certificate.

If the inspection system supports Terminal Authentication, the inspection system's software SHALL support the Terminal Authentication protocol as described in section 7.1.

9 Common Specifications

9.2 Information on Supported Protocols

[Add to Section 9.2]

To indicate support for Terminal Authentication `SecurityInfos` may contain the following entry:

- At least one `TerminalAuthenticationInfo` SHALL be present.

9.2.1 TerminalAuthenticationInfo

[Add as new Section after 9.2.7]

This data structure provides detailed information on an implementation of Terminal Authentication.

- The object identifier `protocol` SHALL identify the *general* Terminal Authentication Protocol as the specific protocol may change over time.
- The integer `version` SHALL identify the version of the protocol. Currently, version 1 is supported by this specification. Note that other specifications [TR-03110] define version 2 of this protocol, which is out of scope of this specification.

9.2.2 Terminal Authentication Object Identifiers

[Add as new Section after 9.2.7]

The following Object Identifier SHALL be used:

```
id-TA OBJECT IDENTIFIER ::= {
  bsi-de protocols(2) smartcard(2) 2
}

id-TA-RSA               OBJECT IDENTIFIER ::= {id-TA
1}id-TA-RSA-PSS-SHA-256   OBJECT IDENTIFIER ::= {id-TA-RSA
4}id-TA-RSA-PSS-SHA-512   OBJECT IDENTIFIER ::= {id-TA-RSA 6}id-TA-ECDSA
OBJECT IDENTIFIER ::= {id-TA 2}id-TA-ECDSA-SHA-224     OBJECT
IDENTIFIER ::= {id-TA-ECDSA 2}
id-TA-ECDSA-SHA-256     OBJECT IDENTIFIER ::= {id-TA-ECDSA 3}
id-TA-ECDSA-SHA-384     OBJECT IDENTIFIER ::= {id-TA-ECDSA 4}
id-TA-ECDSA-SHA-512     OBJECT IDENTIFIER ::= {id-TA-ECDSA 5}

TerminalAuthenticationInfo ::= SEQUENCE {
```

```
  protocol OBJECT IDENTIFIER(id-TA),
  version  INTEGER, -- MUST be 1
}
```

9.2.3 Storage on the Chip

[Replace the 2nd and 3rd bullet in Section 9.2.8 by the following]

- The file EF.CardSecurity contained in the Master File is REQUIRED if
  - PACE with Chip Authentication Mapping is suppported by the IC,
  - Terminal Authentication in the Master File is supported by the IC, or
  - Chip Authentication in the Master File is supported by the IC

  and SHALL contain
  - `ChipAuthenticationInfo` as required by Chip Authentication
  - `ChipAuthenticationPublicKeyInfo` as required by PACE-CAM/Chip Authentication
  - `TerminalAuthenticationInfo` as required by Terminal Authentication
  - the `SecurityInfo`s contained in EF.CardAccess.
- The file EF.DG14 contained in the eMRTD application is REQUIRED if
  - PACE with Generic/Integrated Mapping is suppported by the IC,
  - Terminal Authentication in the eMRTD application is supported by the IC, or
  - Chip Authentication in the eMRTD application is supported by the IC

  and SHALL contain
  - `ChipAuthenticationInfo` as required by Chip Authentication
  - `ChipAuthenticationPublicKeyInfo` as required by Chip Authentication
  - `TerminalAuthenticationInfo` as required by Terminal Authentication
  - the `SecurityInfo`s contained in EF.CardAccess.

9.3 APDUs

9.3.2 Data Objects

[Add as new subsection to Section 9.3]

The sender of a command or response APDU MUST transmit the data objects in the data field in the order as defined in the APDU descriptions.

Note: Accepting data objects in any order is not required, but enhances interoperability for some commands, e.g for MSE:Set AT/General Authenticate. But care is to be taken in case

of commands such as PSO:Verify Certifcate, where the ordering is fixed for cryptographic reasons.

9.4 Public Key Data Objects

9.4.2 RSA Public Keys

[add between sections 9.4.1 and 9.4.2, unless RSA is removed for TA]

The data objects contained in an RSA public key are shown in Table 4. The order of the data objects is fixed.

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

Table 4: RSA Public Key

A. Inspection Procedures (informativ)

[new Appendix to be added to Doc 9303-11]

A.1. Inspection Procedure for eMRTD application

This section describes an inspection procedure which contains only an eMRTD application (“LDS1-documents”).

1. Gain access to the contactless IC (see section 4.2)
   - If access to the IC is protected, PACE or BAC can be used in this step, although it is recommended to use PACE for security reasons. Beginning 1/1/2018 eMRTDs may support PACE only.
   - If supported by IC and terminal, PACE-CAM should be used for performance reasons.
   - The IC grants access to less sensitive data in the eMRTD application and to EF.CardSecurity in the Master File, if present.
   - As a result of this step, the eMRTD application is selected
2. Start authentication of data
   - Read the Document Security Object and verify the signature, including chain verification of the Document Signer Certificate.
3. Authentication of the chip
   - Depending on support by the IC, perform Chip Authentication or Active Authentication. Support of Active Authentication is indicated by the presence of

EF.DG15 in the eMRTD application, support for Chip Authentication by the presence of corresponding `SecurityInfo`s in EF.DG14.

- This step can also be performed as part of step 1, if PACE with Chip Authentication Mapping is used.
- Authentication is only complete in combination with authentication of the file containing the public key (EF.CardSecurity, EF.DG14 or EF.DG15) used for this step.

4. Additional access control
   - Performing Terminal Authentication is necessary, if the eMRTD is configured to require this for access to sensitive data, i.e. EF.DG3 and/or EF.DG4.
5. Read data
   - Reading data can be started as soon as the necessary access rights are granted, e.g. less sensitive data can be read after step 1.
   - Data must not be considered genuine without authentication of the read data (step 2).

A.2. Inspection Procedure for multi-application eMRTDs

This section describes an inspection procedure designed for eMRTDs containing one or more applications besides the eMRTD application (“LDS2-documents”). This procedure can also be used to access the eMRTD application only.

1. Gain access to the contactless IC (see section 4.2)
   - In this setting, only PACE is available to gain access to the IC.
   - If supported by IC and terminal, PACE-CAM should be used for performance reasons.
   - The IC grants access to less sensitive data in the eMRTD application and to EF.CardSecurity in the Master File.
2. Check presence of EF.CardSecurity
   - If EF.CardSecurity is not present, the eMRTD does not support authentication in the Master File (implying that the IC only contains an eMRTD application). In this case select the eMRTD application and continue with step 2 of the procedure in Appendix A.1.
3. Start authentication of data
   - Read EF.CardSecurity and verify the signature, including chain verification of the Document Signer Certificate.
   - Data from the eMRTD application are protected via the Document Security Object, which must be verified when data from this application is read. Data from other applications are protected by signatures of the data, which also must be verified upon reading these data.
4. Authentication of the chip

- Perform Chip Authentication in the Master File. If the necessary information are not contained in the `SecurityInfo`s in EF.CardSecurity, the IC does not support authentication in the Master File. In this case select the eMRTD application and continue with step 2 of the procedure in Appendix A.1.
- This step can also be performed as part of step 1, if PACE with Chip Authentication Mapping is used.
- Authentication is only complete in combination with authentication of the file containing the public key (EF.CardSecurity) used for this step.

5. Additional access control
   - Perform Terminal Authentication.
   - If only read access to less sensitive data in the eMRTD application is required, this step can be skipped.
6. Reading/writing data
   - Reading/writing data includes selection of the applications containing the files.
   - Reading data can be started as soon as the necessary access rights are granted, e.g. less sensitive data of the eMRTD application can be read after step 1.
   - Data must not be considered genuine without authentication of the read data (step 3).

B. European Extended Access Control (informative)

Terminal Authentication as defined in this document is based on Extended Access Control as used in the European Union (see [TR-03110]) to protect access to fingerprints stored in the LDS1-application. This Annex points out the differences between [TR-03110] and the protocols defined in this document.

The Advanced Inspection Procedure used to access eMRTDs equipped with EAC according to [TR-03110] comprises the following steps:

1. Perform the Chip Access Procedure (see section 4.2) and select the eMRTD application;
2. Perform Chip Authentication in the eMRTD application (see section 6.2) and start Passive Authentication (see section 5.1);
3. Perform Terminal Authentication (see below) in the eMRTD application (see section 7.1).

Note: Both Chip and Terminal Authentication are performed in the eMRTD application in the European Extended Access Control. The specifications in this document allow these protocols – depending on context – to be performed either in the eMRTD application or the Master File.

B.1. Access Rights

Access rights to data groups in applications other than the eMRTD application are conveyed via Authorization Extensions as defined in Parts 12 and 10 of Doc 9303. Access rights for fingerprints (and iris) are conveyed via the Certificate Holder Authorization Template:

7	***6***	***5***	***4***	***3***	***2***	***1***	***0***	***Description***
x	x	-	-	-	-	-	-	***Role (see Doc 9303-12)***
-	-	x	x	x	x	x	x	***Access Rights***
-	-	x	x	x	x	-	-	RFU
-	-	-	-	-	-	1	-	Read access to eMRTD application: DG 4 (Iris)
-	-	-	-	-	-	-	1	Read access to eMRTD application: DG 3 (Fingerprint)

Table 5: Authorization of Inspection Systems

For the computation of the effective access rights see section 7.1.4.3.6.

B.2. EF.CVCA

According to the specification, the trust points (Certificate Authority References) known to the ICC for certificate verification as part of Terminal Authentication are transmitted to the PCD as part of the PACE protocol (see section 4.4.3.5).

The European Extended Access Control defines a transparent file EF.CVCA in the eMRTD application instead. The specification is reproduced below:

If the IC supports Terminal Authentication in the eMRTD application, it MUST make the references of CVCA public keys suitable for inspection systems available in a transparent elementary file EF.CVCA in the eMRTD application as specified in Table 6.

File Name	***EF.CVCA***
File ID	0x011C (default)
Short File ID	0x1C (default)
Read Access	PACE
Write Access	NEVER (internally updated only)
Size	36 bytes (fixed) padded with octets of value 0x00
Content	$[CAR_i][\|\|CAR_{i-1}][\|\|0x00..00]$

Table 6: Elementary File EF.CVCA

This file SHALL contain a sequence of Certification Authority Reference (CAR) data objects (see Doc 9303-12) suitable for Terminal Authentication.

- It SHALL contain at most two Certification Authority Reference data objects.
- The most recent Certification Authority Reference SHALL be the first data object in this list.

- The file MUST be padded by appending octets of value 0x00.

The file EF.CVCA has a default EF identifier and short EF identifier. If the default values cannot be used, the (short) EF identifier SHALL be specified in the OPTIONAL parameter `efCVCA` of the `TerminalAuthenticationInfo`. If `efCVCA` is used to indicate the EF identifier to be used, the default EF identifier is overridden. If no short EF identifier is given in `efCVCA`, the file EF.CVCA MUST be explicitly selected using the given EF identifier.

```
TerminalAuthenticationInfo ::= SEQUENCE {
  protocol OBJECT IDENTIFIER(id-TA),
  version  INTEGER, -- MUST be 1
  efCVCA   FileID OPTIONAL
}

FileID ::= SEQUENCE {
  fid  OCTET STRING (SIZE(2)),
  sfid OCTET STRING (SIZE(1)) OPTIONAL
}
```

References

[TR-03110] BSI TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents,

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

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

[7816-8] ISO/IEC 7816-8:2004, Identification cards – Integrated circuit cards – Part 8: Commands for security operations, 2004

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