Federal Office
for Information Security

Technical Guideline TR-03110

Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC)

Version 1.1

History

Version	Date	Comment
1.00	2006-02-08	Initial public version.
1.01	2006-11-02	Minor corrections and clarifications.
1.10	2007-08-20	Revised version.

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63, 53133 Bonn, Germany
Email: `ExtendedAccessControl@bsi.bund.de`
Internet: `http://www.bsi.bund.de/fachthem/epass`

Contents

List of Figures

List of Tables

1. Introduction

The International Civil Aviation Organization (ICAO) standardizes machine readable travel documents in ICAO Doc 9303. This standard consists of three parts:

Part 1: Machine Readable Passports

- Volume 1: Passports with Machine Readable Data Stored in Optical Character Recognition Format
- Volume 2: Specifications for Electronically Enabled Passports with Biometric Identification Capability

Part 2: Machine Readable Visa

Part 3: Machine Readable Official Travel Documents

This technical guideline mainly focuses on and extends the electronic security mechanisms for electronic passports described in Doc 9303 Part 1 Volume 2 [5] to protect the authenticity (including integrity), originality, and confidentiality of the data stored on the radio frequency chip embedded in the passport (MRTD chip). In a nutshell the security mechanisms specified in [5] are *Passive Authentication*, *Active Authentication*, and *Access Control* as summarized in Table 1.1.

Table 1.1.: ICAO Security Mechanisms

Mechanism	**Protection**	**Cryptographic Technique**
Passive Authentication	Authenticity	Digital Signature
Active Authentication	Originality	Challenge-Response
Access Control	Confidentiality	Authentication & Secure Channels

While the implementation of Passive Authentication is mandatory, Active Authentication and Access Control are both optional. It directly follows that without implementing those or equivalent mechanisms the originality and confidentiality of the stored data cannot be guaranteed. This guideline focuses on those aspects and specifies supplementary mechanisms for authentication and access control that are important for a secure MRTD chip.

1.1. Passive Authentication

The ICAO ePassport application basically consists of 16 data groups (DG1-DG16) and a Security Object for Passive Authentication. An overview on the usage of those data groups is given in Table 1.2.

Passive Authentication uses a digital signature to authenticate data stored in the data groups on the MRTD chip. This signature is generated by a Document Signer (e.g. the MRTD producer) in the personalization phase of the MRTD chip over a Security Object containing the hash values of all data groups stored on the chip. For details on the Security Object, Document Signers, and Country Signing CAs the reader is referred to [5].

To verify data stored on an MRTD chip using Passive Authentication the inspection system has to perform the following steps:

1. Read the Security Object from the MRTD chip.

DG	Content	Read / Write	Mandatory / Optional	Access Control
DG1	MRZ	R	M	BAC
DG2	Biometric: Face	R	M	BAC
DG3	Biometric: Finger	R	O	BAC + EAC
DG4	Biometric: Iris	R	O	BAC + EAC
...		R	O	BAC
DG14	SecurityInfos*	R	O	BAC
DG15	Active Authentication	R	O	BAC
DG16	...	R	O	BAC
SO_D	Security Object	R	M	BAC

*DG14 is defined in this specification (cf. Appendix A.1).

Table 1.2.: Data Groups of the ePassport Application

2. Retrieve the corresponding Document Signer Certificate and the trusted Country Signing CA Certificate.
3. Verify the Document Signer Certificate and the signature of the Security Object.
4. Compute hash values of read data groups and compare them to the hash values in the Security Object.

Passive Authentication enables an inspection system to detect manipulated data groups, but it does not prevent cloning of MRTD chips, i.e. copying the complete data stored on one MRTD chip to another MRTD chip.

NOTE: Even with Passive Authentication, an inspection system can detect a cloned MRTD chip by carefully comparing the machine readable zone (MRZ) printed on the datapage to the MRZ stored in data group DG1 on the MRTD chip. This test, however, assumes that it is impossible to physically copy the datapage – which exclusively relies on its physical security features.

1.2. Active Authentication

Active Authentication is a digital security feature that prevents cloning by introducing a chip-individual key pair:

- The public key is stored in data group DG15 and thus protected by Passive Authentication.
- The corresponding private key is stored in secure memory and may only be used internally by the MRTD chip and cannot be read out.

Thus, the chip can prove knowledge of this private key in a challenge-response protocol, which is called Active Authentication. In this protocol the MRTD chip digitally signs a challenge randomly chosen by the inspection system. The inspection system recognizes that the MRTD chip is genuine if and only if the returned signature is correct. Active Authentication is a straightforward protocol and prevents cloning very effectively, but introduces a privacy threat: Challenge Semantics (see Appendix F for a discussion on Challenge Semantics).

1.3. Access Control

Access Control is not only required for privacy reasons but also mitigates the risk of cloning attacks. The MRTD chip protects the stored data against unauthorized access by applying appropriate access control mechanisms as described below:

- Less-sensitive data (e.g. the MRZ, the facial image and other data that is relatively easy to acquire from other sources) required for global interoperable border crossing is protected by *Basic Access Control*. For the reader's convenience, Basic Access Control is described in Appendix E.
- Sensitive data (e.g. fingerprints and other data that cannot be obtained easily from other sources at a large scale) must only be available to authorized inspection systems. Such data is additionally protected by *Extended Access Control*.

Basic Access Control only checks that the inspection system has physical access to the travel document by requiring the MRZ to be read optically. Extended Access Control should additionally check that the inspection system is entitled to read sensitive data. Therefore, strong authentication of the inspection system is required. However, as Extended Access Control is not required for global interoperable border crossing, this protocol is not (yet) specified by ICAO.

1.4. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2]. The key word "CONDITIONAL" is to be interpreted as follows:

CONDITIONAL: The usage of an item is dependent on the usage of other items. It is therefore further qualified under which conditions the item is REQUIRED or RECOMMENDED.

1.5. Abbreviations

The following abbreviations are commonly used throughout this specification.

Name	**Abbrev.**
Card Verifiable	CV
Certification Authority	CA
Chip Authentication Public Key	PK_{PICC}
Chip Authentication Private Key	SK_{PICC}
Country Signing CA	CSCA
Country Verifying CA	CVCA
Country Verifying CA Certificate	C_{CVCA}
Security Object	SO_D
Document Verifier	DV
Document Verifier Certificate	C_{DV}
Domain Parameters	$\mathcal{D}$
International Civil Aviation Organization	ICAO
Inspection System	IS
Inspection System Certificate	C_{IS}
Key Derivation Function	KDF
Logical Data Structure	LDS
Machine Readable Travel Document	MRTD
Proximity Integrated Circuit Chip	PICC
Proximity Coupling Device	PCD
Terminal Authentication Public Key	PK_{PCD}
Terminal Authentication Private Key	SK_{PCD}

2. Advanced Security Mechanisms

This technical guideline specifies two advanced security mechanisms for machine readable travel documents: *Chip Authentication* and *Terminal Authentication*. While Chip Authentication can be used as a stand-alone protocol, e.g. to replace Active Authentication, Terminal Authentication can only be used in combination with Chip Authentication. The combination of both protocols provide an implementation of Extended Access Control.

Chip Authentication

This protocol is an alternative to the optional Active Authentication, i.e. it enables the inspection system to verify that the MRTD chip is genuine but has two advantages over the original protocol.

- Challenge Semantics are prevented because the transcripts produced by this protocol are non-transferable.
- Besides authentication of the MRTD chip this protocol also provides strong session keys for Secure Messaging.

An MRTD chip that supports Chip Authentication MUST also enforce Basic Access Control.

Terminal Authentication

This protocol enables the MRTD chip to verify that the inspection system is entitled to access sensitive data. As the inspection system MAY access sensitive data afterwards, all further communication MUST be protected appropriately. Therefore, Chip Authentication MUST have been successfully executed before starting this protocol – which is enforced by the protocol itself.

2.1. Inspection Procedure

Depending on whether or not a device (i.e. an MRTD chip or an inspection system) is compliant to this specification the device is called *compliant* or *non-compliant*, respectively. Depending on the combination of an inspection system and an MRTD chip, either the *standard inspection procedure* or the *advanced inspection procedure* is used:

- A non-compliant inspection system uses the standard inspection procedure. Less-sensitive data stored on a compliant MRTD chip MUST be readable by every non-compliant inspection system.
- A compliant inspection system SHALL use the advanced inspection procedure if the MRTD chip is compliant. Otherwise the standard inspection procedure SHALL be used.

Table 2.1 gives an overview on the inspection procedures to be used.

NOTE: As described in Section 1.1 Passive Authentication is a continuous process that requires the computation of a hash value of each data group read from the chip and its comparison to the corresponding hash value contained in the Security Object. While this continuous process is assumed to be applied in the following procedures, it is not explicitly described.

Table 2.1.: Inspection Procedures

Inspection system	**MRTD chip**	
	compliant	**non-compliant**
compliant	Advanced	Standard
non-compliant	Standard	Standard

2.1.1. Standard ePassport Inspection Procedure

The standard inspection procedure consists of the following steps:

1. **Select ePassport application** **(REQUIRED)**

   The MRTD chip performs the following:

   a) *BAC used:* It SHALL NOT grant access to any data (except general system data).

   b) *BAC unused:* It SHALL grant access to less-sensitive data (e.g. DG1, DG2, DG14, DG15, etc. and the Security Object).

2. **Basic Access Control** **(CONDITIONAL)**

   This step is REQUIRED if Basic Access Control is enforced by the MRTD chip.

   If successful, the MRTD chip performs the following:

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

3. **Passive Authentication (started)** **(REQUIRED)**

   The inspection system MUST read and verify the Security Object.

4. **Active Authentication** **(OPTIONAL)**

   If available, the inspection system MAY read and verify DG15 and perform Active Authentication.

5. **Read and authenticate data**

   The inspection system MAY read and verify read data groups containing less-sensitive data.

2.1.2. Advanced ePassport Inspection Procedure

The advanced inspection procedure consists of the following steps:

1. **Select ePassport application** **(REQUIRED)**

   The MRTD chip SHALL NOT grant access to any data (except general system data).

2. **Basic Access Control** **(REQUIRED)**

   If successful, the MRTD chip performs the following:

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

3. **Chip Authentication** **(REQUIRED)**

   The inspection system SHALL read DG14 and perform Chip Authentication.

   The MRTD chip performs the following:

   - It SHALL restart Secure Messaging.
   - It SHALL restrict access rights to require Secure Messaging established by Chip Authentication.

4. **Passive Authentication (started)** **(REQUIRED)**

   The inspection system performs the following:

   - It SHALL read and verify the Security Object.
   - It SHALL verify DG14.

5. **Active Authentication** **(OPTIONAL)**

   If available, the inspection system MAY read and verify DG15 and perform Active Authentication.

6. **Terminal Authentication** **(CONDITIONAL)**

   This step is REQUIRED to access sensitive ePassport data.

   If successful the MRTD chip performs the following:

   - It SHALL additionally grant access to data groups according the inspection system's access rights.
   - It SHALL restrict all access rights to require Secure Messaging established by Chip Authentication using the ephemeral public key authenticated by Terminal Authentication.

7. **Read and authenticate data**

   The inspection system MAY read and verify read data groups according to the inspection system's access rights.

NOTE: After a successful execution of Chip Authentication strong session encryption is established rendering the decryption of an eavesdropped communication computationally impossible. However, only after a successful Passive Authentication the MRTD chip may be considered genuine.

2.2. Public Key Infrastructure

Terminal Authentication requires the inspection system to prove to the MRTD chip that it is entitled to access sensitive data. Such an inspection system is equipped with at least one *Inspection System Certificate,* encoding the inspection system's public key and access rights, and the corresponding private key. After the inspection system has proven knowledge of this private key, the MRTD chip grants the inspection system access to sensitive data as indicated in the Inspection System Certificate.

The PKI required for issuing and validating Inspection System Certificates consists of the following entities:

1. Country Verifying CAs (CVCAs)
2. Document Verifiers (DVs)
3. Inspection systems (ISs)

This PKI forms the basis of Extended Access Control. It is illustrated in Figure 2.1.

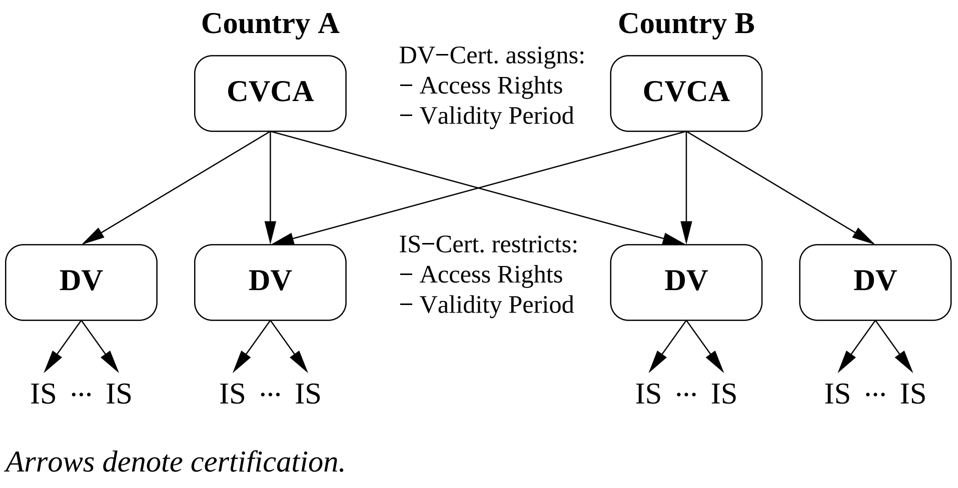

Figure 2.1.: Public Key Infrastructure

2.2.1. Country Verifying CA

Every State is required to set up one trust-point that issues Document Verifier Certificates: the *Country Verifying CA* (CVCA).

NOTE: The Country Signing CA issuing certificates for Document Signers (cf. [5]) and the Country Verifying CA MAY be integrated into a single entity, e.g. a Country CA. However, even in this case, separate key pairs MUST be used for different roles.

A CVCA determines the access rights to national MRTD chips for all DVs (i.e. national DVs as well as the DVs of other States) by issuing certificates for DVs entitled to access some sensitive data. The conditions under which a CVCA grants a DV access to sensitive data is out of the scope of this document and SHOULD be stated in a certificate policy (cf. Appendix A.4.3).

Document Verifier Certificates MUST contain information, such as which data a certain DV is entitled to access. To diminish the potential risk introduced by lost or stolen inspection systems Document Verifier Certificates MUST contain a short validity period. The validity period is assigned by the issuing CVCA at its own choice and this validity period may differ depending on the Document Verifier the certificate is issued to.

2.2.2. Document Verifiers

A *Document Verifier* (DV) is an organizational unit that manages inspection systems belonging together (e.g. inspection systems operated by a State's border police) by – inter alia – issuing Inspection System Certificates. A Document Verifier is therefore a CA, authorized by at least the national CVCA to issue certificates for national inspection systems. The Inspection System Certificates issued by a DV usually inherit both the access rights and the validity period from the Document Verifier Certificate, however, the Document Verifier MAY choose to further **restrict** the access rights or the validity period depending on the inspection system the certificate is issued for.

If a Document Verifier requires its inspection systems to access sensitive data stored on other States' MRTD chips, it MUST apply for a DV Certificate issued by the CVCA of the respective States. The Document Verifier MUST also ensure that all received Document Verifier Certificates are forwarded to the inspection systems within its domain.

2.2.3. Card Verifiable Certificates

CVCA Link Certificates, DV Certificates, and IS Certificates are to be validated by MRTD chips. Due to the computational restrictions of those chips, the certificates MUST be in a card verifiable format:

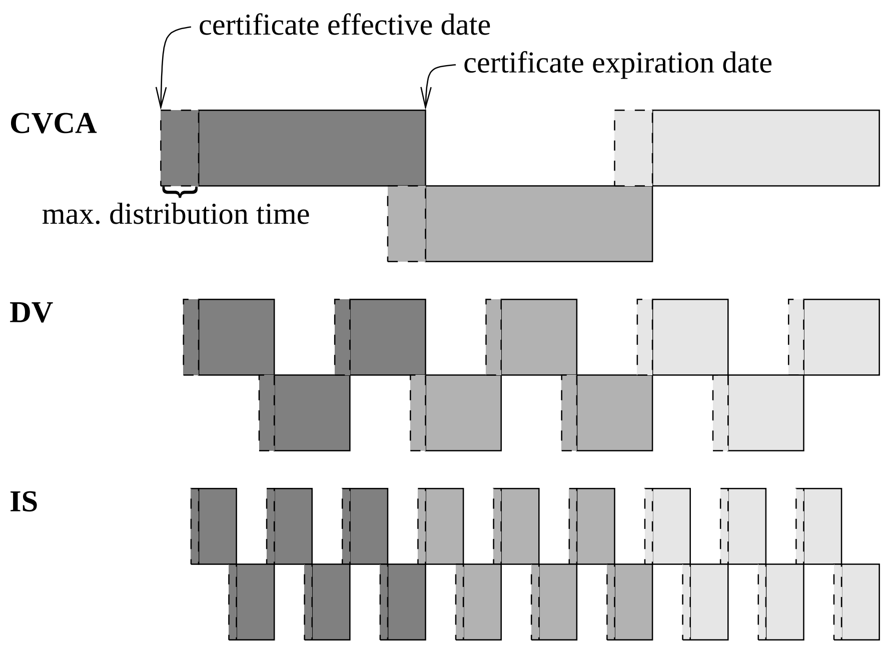

Figure 2.2.: Certificate Scheduling

- The certificate format and profile specified in Appendix A.4.1 SHALL be used.
- The signature algorithm, domain parameters, and key sizes to be used are determined by the CVCA of the issuing State, i.e. the same signature algorithm, domain parameters and key sizes MUST be used within a certificate chain.[1]
- CVCA Link Certificates MAY include a public key that deviates from the current parameters, i.e. the CVCA MAY switch to a new signature algorithm, new domain parameters, or key sizes.

2.2.3.1. Certificate Scheduling

Each certificate MUST contain a validity period. This validity period is identified by two dates, the *certificate effective date* and the *certificate expiration date.*

Certificate Effective Date: The certificate effective date SHALL be the date of the certificate generation.

Certificate Expiration Date: The certificate expiration date MAY be arbitrarily chosen by the certificate issuer.

When generating certificates the issuer MUST carefully plan the renewal of certificates, as sufficient time for propagation of certificates and set up of certificate chains MUST be provided. Obviously, a new certificate must be generated before the current certificate expires. The resulting *maximum distribution time* equals the certificate expiration date of the old certificate minus the certificate effective date of the new certificate. Certificate scheduling is illustrated in Figure 2.2.

2.2.4. Certificate Validation

To validate an IS Certificate, the MRTD chip MUST be provided with a certificate chain starting at a trust-point stored on the MRTD chip. Those trust-points are more or less recent public keys of the

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

MRTD chip's CVCA. The initial trust-point(s) SHALL be stored in the MRTD chip's secure memory in the production or (pre-) personalization phase.

As the key pair used by the CVCA changes over time, CVCA Link Certificates have to be produced. The MRTD chip is REQUIRED to internally update its trust-point(s) according to received valid link certificates.

NOTE: Due to the scheduling of CVCA Link Certificates (cf. Figure 2.2), at most two trust-points need to be stored on the MRTD chip.

The MRTD chip MUST only accept *recent* IS Certificates. If the MRTD chip has no internal clock, the *current date* SHALL be approximated as described below. Thus, the MRTD chip only verifies that a certificate is *apparently* recent (i.e. with respect to the approximated current date).

Current Date: The current date stored on the MRTD chip is initially the date of the (pre-) personalization. This date is then autonomously approximated by the MRTD chip using the most recent certificate effective date contained in a valid CVCA Link Certificate, a DV Certificate or a domestic IS Certificate.

An inspection system MAY send CVCA Link Certificates, DV Certificates, and IS Certificates to an MRTD chip to update the current date and the trust-point stored on the MRTD chip even if the inspection system does not intend to or is not able to continue with Terminal Authentication.

2.2.4.1. General Procedure

The certificate validation procedure consists of two steps:

1. **Certificate Verification:** the signature MUST be valid, the certificate MUST NOT be expired. If the verification fails, the procedure SHALL be aborted.
2. **Update of the internal status:** the current date MUST be *updated*, the public key and the attributes MUST be imported, new trust-points MUST be *enabled*, expired trust-points MUST be *disabled*.

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

2.2.4.2. Example Procedure

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

1. The MRTD chip verifies the signature on the certificate. If the signature is incorrect, the verification fails.
2. The certificate expiration date is compared to the MRTD chip's current date. If the expiration date is before the current date, the verification fails.
3. The certificate is valid and the public key and the relevant attributes contained in the certificate are imported.
   a) For CVCA Link Certificates:
      The new CVCA public key is added to the list of trust-points stored in the MRTD chip's secure memory. The new trust-point is then enabled.

   b) For DV and IS Certificates:

      The new DV or IS public key is temporarily imported for subsequent certificate verification or Terminal Authentication, respectively.

4. For CVCA, DV, and domestic IS Certificates:

   The certificate effective date is compared to the MRTD chip's current date. If the current date is before the effective date, the current date is updated to the effective date.

5. Expired trust-points stored in the MRTD chip's secure memory are disabled and may be removed from the list of trust-points.

3. Protocol Specifications

In this section cryptographic protocols for Chip Authentication and Terminal Authentication are specified assuming an arbitrary communication infrastructure. A mapping to ISO 7816 commands is given in Appendix B.

3.1. Cryptographic Algorithms and Notation

The protocols are executed between two parties: the MRTD chip (PICC) and the inspection system (PCD). The following cryptographic operations and notations are used.

3.1.1. Key Agreement

The keys and operations for key agreement are described in an algorithm-independent way. A mapping to DH and ECDH can be found in Appendix A.2.

3.1.1.1. Keys

The following key pairs are used:

- The MRTD chip has a static Diffie-Hellman key pair (or Chip Authentication Key Pair). The public key is PK_{PICC}, the corresponding private key is SK_{PICC}, the domain parameters are $\mathscr{D}_{PICC}$.
- The inspection system generates an ephemeral Diffie-Hellman key pair for every new communication using the MRTD chip's domain parameters $\mathscr{D}_{PICC}$. The ephemeral public key is $\widetilde{PK_{PCD}}$, the corresponding private key is $\widetilde{SK_{PCD}}$.

It is RECOMMENDED that the MRTD chip validates the ephemeral public key received from the inspection system.

3.1.1.2. Operations

Generating a shared secret K is denoted by $\mathbf{KA}(SK_{PICC}, \widetilde{PK_{PCD}}, \mathscr{D}_{PICC})$ for the MRTD chip and $\mathbf{KA}(\widetilde{SK_{PCD}}, PK_{PICC}, \mathscr{D}_{PICC})$ for the inspection system.

3.1.2. Signatures

The keys and operations for signatures are described in an algorithm-independent way. A mapping to RSA and ECDSA can be found in Appendix A.3.

3.1.2.1. Keys

The inspection system has a static signature key pair (or Terminal Authentication Key Pair). The public key is PK_{PCD}, the corresponding private key is SK_{PCD}.

MRTD Chip (PICC)		Inspection System (PCD)
static key pair: $(SK_{PICC}, PK_{PICC}, \mathcal{D}_{PICC})$		
	$\xrightarrow[\mathcal{D}_{PICC}]{PK_{PICC}}$	choose random ephemeral key pair $(\widetilde{SK_{PCD}}, \widetilde{PK_{PCD}}, \mathcal{D}_{PICC})$
	$\xleftarrow{\widetilde{PK_{PCD}}}$	
$K = \mathbf{KA}(SK_{PICC}, \widetilde{PK_{PCD}}, \mathcal{D}_{PICC})$		$K = \mathbf{KA}(\widetilde{SK_{PCD}}, PK_{PICC}, \mathcal{D}_{PICC})$

Figure 3.1.: Chip Authentication

3.1.2.2. Operations

The operations for signing and verifying a message are denoted as follows:

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

3.2. Chip Authentication

Chip Authentication is an ephemeral-static Diffie-Hellman key agreement protocol that provides secure communication and implicit unilateral authentication of the MRTD chip.

3.2.1. Protocol Specification

The following steps are performed by the inspection system and the MRTD chip, a simplified version is also shown in Figure 3.1:

1. The MRTD chip sends its static Diffie-Hellman public key PK_{PICC}, and the domain parameters $\mathcal{D}_{PICC}$ to the inspection system.
2. The inspection system generates an ephemeral Diffie-Hellman key pair $(\widetilde{SK_{PCD}}, \widetilde{PK_{PCD}}, \mathcal{D}_{PICC})$, and sends the ephemeral public key $\widetilde{PK_{PCD}}$ to the MRTD chip.
3. Both the MRTD chip and the inspection system compute the following:
   a) The shared secret $K = \mathbf{KA}(SK_{PICC}, \widetilde{PK_{PCD}}, \mathcal{D}_{PICC}) = \mathbf{KA}(\widetilde{SK_{PCD}}, PK_{PICC}, \mathcal{D}_{PICC})$
   b) The session keys K_{MAC} and K_{Enc} derived from K for Secure Messaging.
   c) The hash of the inspection system's ephemeral public key $H(\widetilde{PK_{PCD}})$ for Terminal Authentication.

To verify the authenticity of the PK_{PICC} the inspection system SHALL perform Passive Authentication directly after Chip Authentication.

3.2.2. Security Status

If Chip Authentication was successfully performed, Secure Messaging is restarted using the derived session keys K_{MAC} and K_{Enc}. Otherwise, Secure Messaging is continued using the previously established session keys (Basic Access Control).

MRTD Chip (PICC)		**Inspection System (PCD)**
choose r_{PICC} randomly	$\xrightarrow{r_{PICC}}$	
	$\xleftarrow{s_{PCD}}$	$s_{PCD} = \mathbf{Sign}(SK_{PCD}, ID_{PICC} \Vert r_{PICC} \Vert H(\widetilde{PK_{PCD}}))$
$\mathbf{Verify}(PK_{PCD}, s_{PCD}, ID_{PICC} \Vert r_{PICC} \Vert H(\widetilde{PK_{PCD}}))$		

Figure 3.2.: Terminal Authentication

NOTE: Passive Authentication MUST be performed directly after Chip Authentication. Only after a successful validation of the Security Object read from the MRTD chip using the new session keys the MRTD chip may be considered genuine.

3.3. Terminal Authentication

Terminal Authentication is a two move challenge-response protocol that provides explicit unilateral authentication of the inspection system.

3.3.1. Protocol Specification

The following steps are performed by the inspection system and the MRTD chip, a simplified version is also shown in Figure 3.2:

1. The inspection system sends a certificate chain to the MRTD chip. The chain starts with a certificate verifiable with a CVCA public key stored on the chip and ends with the inspection system's IS Certificate.

2. The MRTD chip verifies the certificates and extracts the inspection system's public key PK_{PCD}. Then it sends the challenge r_{PICC} to the inspection system.

3. The inspection system responds with the signature

$$s_{PCD} = \mathbf{Sign}(SK_{PCD}, ID_{PICC}||r_{PICC}||H(\widetilde{PK_{PCD}})).$$

4. The MRTD chip checks that

$$\mathbf{Verify}(PK_{PCD}, s_{PCD}, ID_{PICC}||r_{PICC}||H(\widetilde{PK_{PCD}})) = \text{true}.$$

In this protocol ID_{PICC} is the MRTD chip's Document Number including the check digit (similar to Basic Access Control) as contained in the MRZ and $H(\widetilde{PK_{PCD}})$ is the hash value of the inspection system's ephemeral Diffie-Hellman public key from Chip Authentication.

NOTE: All messages MUST be transmitted with Secure Messaging in Encrypt-then-Authenticate mode using session keys derived from Chip Authentication.

3.3.2. Security Status

If Terminal Authentication was successfully performed, the MRTD chip SHALL grant access to stored sensitive data according to the effective authorization level of the authenticated inspection system.

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).

4. Security & Privacy

In this section the formal correctness of the protocols is shown. Following the ideas proposed in [1] a transition from the *Authenticated Link Model* to the *Unauthenticated Link Model* is used to prove the security of the protocols.

Authenticated Link Model: The Authenticated Link Model is an idealized setting where all messages are a priori authenticated.

Unauthenticated Link Model: The Unauthenticated Link Model is the real-world setting where messages are unauthenticated.

The Authenticated Link Model restricts the adversary to attacks on the cryptographic primitive itself and to attacks that do not have impact on the security of the protocol (e.g. denial of service attacks). In this model key agreement would be sufficient to set up the secure channel. The security of the underlying Diffie-Hellman protocol is directly based on the assumption that the Computational Diffie-Hellman Problem is hard.

The transition from the Authenticated Link Model to the Unauthenticated Link Model is done by applying appropriate *Authenticators*, turning unauthenticated messages into authenticated messages. Actually, Chip Authentication and Terminal Authentication are such authenticators. Unfortunately, there is no clear definition of the properties of an authenticator in the literature and the corresponding security proofs are quite blurred. To make such proofs more transparent, we give a definition of an authenticator:

Authenticator: A message sent from an originator to a recipient shall be authenticated. It directly follows that the following three properties are sufficient for authentication of the message:

- **Origin:** The recipient must be able to identify the sender of the message.
- **Destination:** The originator must be able to indicate the intended recipient of the message.
- **Freshness:** The recipient must be able to check that the message is not a copy of a previous message.

4.1. Chip Authentication

Chip Authentication is similar to the cipher-based authenticator proposed in [1], also shown in Figure 4.1. It is a two-move protocol that is used to protect the message m_{PICC} sent from the MRTD chip to the inspection system by authenticating the message with a MAC.

To make the transition resulting from the application of the cipher-based authenticator to the basic Diffie-Hellman protocol more clear, consider that the encryption $e_{PCD} = \mathbf{E}(PK_{PICC}, K)$ can be safely replaced by the ephemeral key $\widetilde{PK_{PCD}}$, because this is actually an encryption of $K = \mathbf{KA}(\widetilde{SK_{PCD}}, PK_{PICC}, \mathscr{D}_{PICC})$ (see also Proposition 5 and Remark 1 in [1]).

- **Origin:** Computation of the MAC requires knowledge of the authentication key K. It directly follows from the Computational Diffie-Hellman assumption that only the MRTD chip (and the inspection system) can generate K from $\widetilde{PK_{PCD}}$.

MRTD Chip (PICC)		**Inspection System (PCD)**
	$\xleftarrow{e_{PCD}}$	choose K randomly $e_{PCD} = \mathbf{E}(PK_{PICC}, K)$
$s_{PICC} = \mathbf{MAC}(K, m_{PICC} \| \| ID_{PCD})$	$\xrightarrow[s_{PICC}]{m_{PICC}}$	

Figure 4.1.: Cipher-based Authenticator

MRTD Chip (PICC)		**Inspection System (PCD)**
	$\xleftarrow{m_{PCD}}$	
choose r_{PICC} randomly	$\xrightarrow{r_{PICC}}$	
	$\xleftarrow{s_{PCD}}$	$s_{PCD} =$ $\mathbf{Sign}(SK_{PCD}, ID_{PICC} \| \| r_{PICC} \| \| m_{PCD})$

Figure 4.2.: Signature-based Authenticator

- **Destination:** The chip includes the identity of the inspection system in the MAC. If the inspection system remains anonymous, the distinguishing identifier ID_{PCD} can be removed from the MAC. In this case the message is intended for the inspection system that is able to verify the MAC (and thus has knowledge of K).
- **Freshness:** If the the inspection system chooses the ephemeral key pair $(\widetilde{SK_{PCD}}, \widetilde{PK_{PCD}})$ randomly and uniformly, the authentication key K is also generated randomly and uniformly.

4.1.1. Summarized Properties

Chip Authentication has the following properties:

1. Implicit authentication of the MRTD chip.
2. Secure messaging with forward secrecy.[1]

4.1.2. Remaining Risks

Chip Authentication alone does not necessarily guarantee that the MRTD chip contained in a presented document is genuine. To preclude sophisticated attacks (e.g. the *"Grandmaster Chess Attack"*) the authenticity and integrity of the printed MRZ and DG1 of the ePassport application MUST be verified and it MUST be checked that both are identical.

This implies that in addition to Chip Authentication and Passive Authentication the physical security features of the document MUST be additionally checked to verify the integrity and authenticity of the printed MRZ.

4.2. Terminal Authentication

Terminal Authentication is the signature-based authenticator proposed in [1], also shown in Figure 4.2. It is a three-move protocol that is used to protect the integrity of the message m_{PCD} sent from the inspection system to the chip by authenticating the message with a signature.

[1]Assuming that the inspection system chooses $\widetilde{PK_{PCD}}$ randomly and erases the secret key $\widetilde{SK_{PCD}}$ directly after generating the session keys, a compromise of the inspection system's static key pair does not affect the secrecy of past sessions.

- **Origin:** Computation of the signature s_{PCD} requires knowledge of the private key SK_{PCD}. Thus, only the inspection system can generate the signature.
- **Destination:** The inspection system includes the identity of the chip in the signature.
- **Freshness:** If the chip chooses the challenge randomly and uniformly it is guaranteed that the signature s_{PCD} is recent, as the challenge is included in the signed data.

4.2.1. Summarized Properties

Terminal Authentication has the following properties:

1. Explicit authentication of the inspection system.
2. Key confirmation for Secure Messaging.

4.2.2. Remaining Risks

Terminal Authentication mitigates the risk introduced by lost or stolen inspection systems by authorizing an inspection system to access sensitive data only for a short period of time. Due to the approximation of the current date, sensitive data may be theoretically read by an already expired inspection system.

On the one hand, an infrequently used travel document is obviously more affected by such an attack. On the other hand, the attack is more difficult to mount on an infrequently used travel document, as access to an MRTD chip still requires consent of the bearer which is enforced by Basic Access Control.

Furthermore, what cannot be prevented is an attacker being able to subvert an inspection system and gain access to sensitive data.

4.3. Challenge Semantics

Terminal Authentication is a challenge-response protocol based on digital signatures, which is obviously not free from challenge semantics. This potential attack is however less important, as the inspection system is usually not concerned about its privacy.

Therefore, we only have to show that Chip Authentication and the cipher-based authenticator provide a non-transferable proof of knowledge. This can be done by showing that the protocol is simulatable without the chip's private key, and that the simulated transcript is indistinguishable from a real transcript. The simulation is trivial:

Input: The chip's static public key PK_{PICC}, the domain parameters $\mathscr{D}_{PICC}$, and a message m_{PICC}.

Output: The authenticated message $s_{PICC} = \mathbf{MAC}(K, m_{PICC}||ID_{PCD})$, where the authentication key is $K = \mathbf{KA}(\widetilde{SK_{PCD}}, PK_{PICC}, \mathscr{D}_{PICC})$ and $\widetilde{SK_{PCD}}$ is a randomly chosen ephemeral private key of the inspection system.

In other words, Chip Authentication is free from challenge semantics because the MAC is based on symmetric cryptography. Any party being able to verify the MAC is also able to compute the MAC.

Appendix A.
Key Management (Normative)

The object identifiers used in the following appendices are contained in the subtree of `bsi-de`:

```
bsi-de OBJECT IDENTIFIER ::= {
  itu-t(0) identified-organization(4) etsi(0)
  reserved(127) etsi-identified-organization(0) 7
}
```

A.1. Information on Supported Security Protocols

The ASN.1 data structure `SecurityInfos` SHALL be provided by the MRTD chip in DG14 to indicate supported security protocols. The data structure is specified as follows:

```
SecurityInfos ::= SET of SecurityInfo

SecurityInfo  ::= SEQUENCE {
  protocol      OBJECT IDENTIFIER,
  requiredData  ANY DEFINED BY protocol,
  optionalData  ANY DEFINED BY protocol OPTIONAL
}
```

The elements contained in a `SecurityInfo` data structure have the following meaning:

- The object identifier `protocol` identifies the supported protocol.
- The open type `requiredData` contains protocol specific mandatory data.
- The open type `optionalData` contains protocol specific optional data.

A.1.1. Supported Protocols

The ASN.1 specifications for the protocols provided in this specification are described in the following.

NOTE: The following data structures are introduced in this version of this specification:

- `ChipAuthenticationInfo`
- `TerminalAuthenticationInfo`

MRTD chips implemented according to Version 1.0.x of this specification will only provide a `ChipAuthenticationPublicKeyInfo`. In this case the inspection system SHOULD assume the following:

- The MRTD chip supports Chip Authentication.
- The MRTD chip may support Terminal Authentication.

To determine whether or not sensitive data protected by Terminal Authentication is stored on the MRTD chip, the inspection system may consult the Security Object and the elementary file EF.CVCA.

A.1.1.1. Chip Authentication

To indicate support for Chip Authentication `SecurityInfos` may contain the following entries:

- At least one `ChipAuthenticationPublicKeyInfo` MUST be present. The data structure provides the MRTD chip's Chip Authentication Public Key. Multiple entries containing different public keys (i.e. with different algorithms and/or key lengths) MAY be present. In this case the optional `keyId` MUST be used to indicate the local key identifier.
- At least one `ChipAuthenticationInfo` SHOULD be present. This data structure provides detailed information on the protocol itself (i.e. protocol version and Secure Messaging). If more than one Chip Authentication Public Key is present the optional `keyId` MUST be used to indicate the local key identifier.

The ASN.1 definition of Chip Authentication is as follows:

```
id-PK OBJECT IDENTIFIER ::= {
  bsi-de protocols(2) smartcard(2) 1
}

id-CA OBJECT IDENTIFIER ::= {
  bsi-de protocols(2) smartcard(2) 3
}

id-PK-DH              OBJECT IDENTIFIER ::= {id-PK 1}
id-PK-ECDH            OBJECT IDENTIFIER ::= {id-PK 2}

id-CA-DH              OBJECT IDENTIFIER ::= {id-CA 1}
CA-DH-3DES-CBC-CBC    OBJECT IDENTIFIER ::= {id-CA-DH 1}

id-CA-ECDH            OBJECT IDENTIFIER ::= {id-CA 2}
CA-ECDH-3DES-CBC-CBC  OBJECT IDENTIFIER ::= {id-CA-ECDH 1}

ChipAuthenticationInfo ::= SEQUENCE {
  protocol CA-DH-3DES-CBC-CBC||
           CA-ECDH-3DES-CBC-CBC,
  version  INTEGER, -- MUST be 1
  keyId    INTEGER OPTIONAL
}

ChipAuthenticationPublicKeyInfo ::= SEQUENCE {
  protocol                     id-PK-DH||
                               id-PK-ECDH,
  chipAuthenticationPublicKey SubjectPublicKeyInfo,
  keyId                        INTEGER OPTIONAL
}

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm        AlgorithmIdentifier,
    subjectPublicKey BIT STRING
  }

  AlgorithmIdentifier ::= SEQUENCE {
    algorithm  OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL
  }
```

The `chipAuthenticationPublicKey` contains the public key in encoded form. More details on the specification of `SubjectPublicKeyInfo` and `AlgorithmIdentifier` can be found in [4].

A.1.1.2. Terminal Authentication

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

- At least one `TerminalAuthenticationInfo` SHOULD be present. This data structure provides detailed information on the protocol itself (i.e. protocol version and location of the elementary file EF.CVCA).

The ASN.1 definition of Terminal Authentication is as follows:

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

  id-TA-RSA            OBJECT IDENTIFIER ::= {id-TA 1}
  TA-RSA-v1_5-SHA-1    OBJECT IDENTIFIER ::= {id-TA-RSA 1}
  TA-RSA-v1_5-SHA-256  OBJECT IDENTIFIER ::= {id-TA-RSA 2}
  TA-RSA-PSS-SHA-1     OBJECT IDENTIFIER ::= {id-TA-RSA 3}
  TA-RSA-PSS-SHA-256   OBJECT IDENTIFIER ::= {id-TA-RSA 4}

  id-TA-ECDSA          OBJECT IDENTIFIER ::= {id-TA 2}
  TA-ECDSA-SHA-1       OBJECT IDENTIFIER ::= {id-TA-ECDSA 1}
  TA-ECDSA-SHA-224     OBJECT IDENTIFIER ::= {id-TA-ECDSA 2}
  TA-ECDSA-SHA-256     OBJECT IDENTIFIER ::= {id-TA-ECDSA 3}

  TerminalAuthenticationInfo ::= SEQUENCE {
    protocol id-TA,
    version  INTEGER, -- MUST be 1
    efCVCA   FileID OPTIONAL
  }

  FileID ::= SEQUENCE {
    fid  INTEGER
    sfid INTEGER OPTIONAL
  }
```

A.1.1.3. Other Protocols

`SecurityInfos` MAY contain references to protocols that are not contained in this specification (including Active Authentication and Basic Access Control).

Table A.1.: Algorithms and Formats for Chip Authentication

Algorithm / Format	DH	ECDH
Key Agreement Algorithm	PKCS#3 [17]	KAEG [7, 9, 3]
Public Key Format	PKCS#3 [17]	ECC [3]
Key Derivation Function	ICAO 3DES KDF [5, 3]	
Ephemeral Public Key Hash	SHA-1 [15]	X-Coordinate
Ephemeral Public Key Validation	RFC 2631[16]	ECC [3]

A.2. Chip Authentication

A.2.1. Storage on the Chip

The Chip Authentication Key Pair MUST be stored on the MRTD chip.

- The private key SHALL be stored in the MRTD chip's secure memory.
- The public key SHALL be provided in the `ChipAuthenticationPublicKeyInfo` structure.

The MRTD chip MAY support more than one Chip Authentication Key Pair (i.e. the chip may support different algorithms and/or key lengths). In this case the local key identifier MUST be disclosed in the corresponding `ChipAuthenticationPublicKeyInfo` (cf. Appendix B.1.1).

A.2.2. Chip Authentication with DH

For Chip Authentication with DH the respective algorithms and formats from Table A.1 MUST be used.

A.2.3. Chip Authentication with ECDH

For Chip Authentication with ECDH the respective algorithms and formats from Table A.1 MUST be used. The elliptic curve domain parameters MUST be described explicitly in the `ChipAuthenticationPublicKeyInfo` structure, i.e. named curves and implicit domain parameters MUST NOT be used.

A.2.4. Ephemeral Public Keys

The domain parameters contained in the `ChipAuthenticationPublicKeyInfo` structure MUST be used by the inspection system for the generation of an ephemeral public key. Ephemeral public keys MUST be exchanged as plain public key values. More information on the encoding can be found in Appendix C.3.

According to Section 3.1.1 the validation of ephemeral public keys is RECOMMENDED. For DH, the validation algorithm requires the MRTD chip to have a more detailed knowledge of the domain parameters (i.e. the order of the used subgroup) than usually provided by PKCS#3.

A.3. Terminal Authentication

A.3.1. Public Key Import

Public keys imported by the certificate validation procedure (cf. Section 2.2.4) are either *permanently* or *temporarily* stored on the MRTD chip.

A.3.1.1. Permanent Import

Public keys contained in CVCA Link Certificates SHALL be permanently imported by the MRTD chip and MUST be stored in secure memory. A permanently imported public key SHALL fulfill the following conditions:

- It MAY be overwritten *after expiration* by a subsequent permanently imported public key.
- It MUST NOT be overwritten by a temporarily imported public key.

Enabling and disabling a permanently imported public key MUST be an atomic operation. The internal state MUST be reflected in the file EF.CVCA (cf. Table A.2).

A.3.1.2. Temporary Import

Public keys contained in DV and IS Certificates SHALL be temporarily imported by the MRTD chip. A temporarily imported public key SHALL fulfill the following conditions:

- It SHALL NOT be selectable or usable after a power down of the MRTD chip.
- 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. An inspection system MUST NOT make use of any temporarily imported public key but the most recently imported.

A.3.1.3. Imported Metadata

For each permanently or temporarily imported public key the following additional data contained in the certificate (cf. Appendix A.4.1) MUST be stored:

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

The calculation of the effective role (CVCA, DV, or IS) and the effective authorization of the certificate holder is described in Appendix A.5.

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

A.3.1.4. EF.CVCA

The MRTD chip MUST make the references of trusted CVCA public keys available to inspection systems in a transparent elementary file EF.CVCA as specified in Table A.2. This file contains a sequence of Certification Authority Reference (CAR) data objects (cf. Appendix A.4.1.2) structured as follows:

- 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 zeros.

Table A.2.: Elementary File EF.CVCA

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

Table A.3.: Object Identifiers for Terminal Authentication with RSA

OID	**Signature**	**Hash**	**Parameters**
TA-RSA-v1_5-SHA-1	RSASSA-PKCS1-v1_5	SHA-1	N/A
TA-RSA-v1_5-SHA-256	RSASSA-PKCS1-v1_5	SHA-256	N/A
TA-RSA-PSS-SHA-1	RSASSA-PSS	SHA-1	default
TA-RSA-PSS-SHA-256	RSASSA-PSS	SHA-256	default

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

A.3.2. Terminal Authentication with RSA

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

A.3.2.1. Signature Algorithm

RSA [14, 18] as specified in Table A.3 SHALL be used. The default parameters to be used with RSA-PSS are defined as follows:

- Hash Algorithm: The hash algorithm is selected according to Table A.3.
- Mask Generation Algorithm: MGF1 [14, 18] using the selected hash algorithm.
- Salt Length: Octet length of the output of the selected hash algorithm.
- Trailer Field: 0xBC

A.3.2.2. Public Key Format

The TLV-Format [12] as described in Appendix C.3.1 SHALL be used.

- The object identifier SHALL be taken from Table A.3.
- The bit length of the modulus SHALL be 1024, 1280, 1536, 2048, or 3072.

A.3.3. Terminal Authentication with ECDSA

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

Table A.4.: Object Identifiers for Terminal Authentication with ECDSA

OID	**Signature**	**Hash**
TA-ECDSA-SHA-1	ECDSA	SHA-1
TA-ECDSA-SHA-224	ECDSA	SHA-224
TA-ECDSA-SHA-256	ECDSA	SHA-256

Table A.5.: CV Certificate Profile

Data Object	**Cert**
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
Signature	m

m: mandatory, x: must not be used

A.3.3.1. Signature Algorithm

ECDSA with plain signature format [7, 8, 3] as specified in Table A.4 SHALL be used.

A.3.3.2. Public Key Format

The TLV-Format [12] as described in Appendix C.3.2 SHALL be used.

- The object identifier SHALL be taken from Table A.4.
- The bit length of the curve SHALL be 160, 192, 224, or 256.
- Domain Parameters SHALL be compliant to [3].

A.4. CV Certificates

A.4.1. Certificate Profile

Self-descriptive card verifiable (CV) certificates according to ISO 7816 [10, 11, 12] and the certificate profile specified in Table A.5 SHALL be used. Details on the encoding of the data objects used in the certificate profile can be found in Appendix C.2.

A.4.1.1. Certificate Profile Identifier

The version of the profile is indicated by the Certificate Profile Identifier. Version 1 as specified in Table A.5 is identified by a of value 0.

A.4.1.2. Certification Authority Reference

The Certification Authority Reference is used to identify to public key to be used to verify the signature of the certification authority (CVCA or DV). The Certificate Authority Reference MUST be equal to the Certificate Holder Reference in the corresponding certificate of the certification authority (CVCA Link Certificate or DV Certificate).

A.4.1.3. Public Key

Details on the encoding of public keys can be found in Appendix C.3.

A.4.1.4. Certificate Holder Reference

The Certificate Holder Reference SHALL identify a public key of the certificate holder. It MUST be a unique identifier *relative* to the issuing certification authority. It SHALL consist of the following concatenated elements[1]:

1. The ISO 3166-1 ALPHA-2 country code of the certificate holder's country.
2. A mnemonic that represents the certificate holder.
3. A numeric or alphanumeric sequence number.

NOTE: It is **not** guaranteed that the Certificate Holder Reference is a unique identifier in general.

A.4.1.5. 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:

1. An object identifier that specifies the format of the template.
2. A discretionary data object that encodes the role and authorization of the certificate holder.

For the ePassport application the content and evaluation of the Certificate Holder Authorization Template is described in Appendix A.7.

A.4.1.6. Certificate Effective/Expiration Date

Indicates the validity period of the certificate. The Certificate Effective Date MUST be the date of the certificate generation.

A.4.1.7. 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.

A.4.2. Certificate Requests

Certificate requests are reduced CV certificates that may carry an additional signature. The certificate request profile specified in Table A.6 SHALL be used. Details on the encoding of the data objects used in the certificate request profile can be found in Appendix C.2.

[1]The Certificate Holder Reference is an identifier that need not be parsed by inspection systems and MRTD chips.

Table A.6.: CV Certificate Request Profile

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

m: mandatory, x: must not be used, c: conditional

A.4.2.1. Certificate Profile Identifier

The version of the profile is identified by the Certificate Profile Identifier. Version 1 as specified in Table A.6 is identified by a of value 0.

A.4.2.2. Public Key

Details on the encoding of public keys can be found in Appendix C.3.

A.4.2.3. Certificate Holder Reference

In a certificate request the Certificate Holder Reference SHALL only identify the certificate holder **not** the public key. It SHALL consist of the following concatenated elements:

1. The ISO 3166-1 ALPHA-2 country code of the certificate holder's country.
2. A mnemonic that represents the certificate holder.

The sequence number SHALL be assigned by the receiving certification authority.

A.4.2.4. Signature(s)

A certificate request may have 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 Certificate 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 Certificate Authority Reference (i.e. both including tag and length).

A.4.3. Certificate Policy

It is RECOMMENDED that each CVCA and every DV publishes a certificate policy and/or a certification practice statement.

A.4.3.1. Naming Scheme

The certificate policy SHOULD provide details on the naming scheme:

- Composition of the certificate holder mnemonic.
- Handling of the sequence number (e.g. format and wrap-around).

A.4.3.2. Procedures

The certificate policy SHOULD specify the following procedures:

- Entity identification, authentication, and registration.
- Certificate application, issuance, and distribution.
- Compromise and disaster recovery.
- Auditing.

A.4.3.3. Usage Restrictions

The certificate policy SHOULD imply restrictions on the devices used to store/process corresponding private keys and other sensitive (personal) data:

- Physical and operational security.
- Access control mechanisms.
- Evaluation and certification (e.g. Common Criteria Protection Profiles).
- Data Protection.

A.5. Roles and Authorization Levels

The following object identifier SHALL be used to identify roles and authorization levels for EAC-protected ePassports:

```
id-EAC-ePassport OBJECT IDENTIFIER ::= {
  bsi-de applications(3) mrtd(1) roles(2) 1
}
```

Table A.7.: Encoding of Roles and Access Rights

7	6	5	4	3	2	1	0	Description
x	x	-	-	-	-	-	-	**Role**
1	1	-	-	-	-	-	-	CVCA
1	0	-	-	-	-	-	-	DV (domestic)
0	1	-	-	-	-	-	-	DV (foreign)
0	0	-	-	-	-	-	-	IS
-	-	x	x	x	x	x	x	**Access Rights**
-	-	0	0	0	0	-	-	RFU
-	-	-	-	-	-	1	-	Read access to DG 4 (Iris)
-	-	-	-	-	-	-	1	Read access to DG 3 (Fingerprint)

A.5.1. Relative Authorization

The *relative authorization* of the certificate holder is encoded in one byte which is to be interpreted as binary bit map as shown in Table A.7. In more detail, this bit map contains a role and access rights. Both are relative to the authorization of all previous certificates in the chain.

A.5.2. Effective Authorization

To determine the *effective authorization* of a certificate holder, the MRTD chip MUST calculate a bitwise Boolean 'and' of the *relative authorization* contained in the current certificate and effective authorization of the previous certificate in the chain. As the certificate chain always starts with a CVCA public key stored on the MRTD chip, the initial value for the effective authorization is set to the (relative) authorization of the CVCA stored on the chip.

A.5.3. Access Rights

The effective authorization is to be interpreted by the MRTD chip as follows:

- The effective role is a CVCA:
  - This link certificate was issued by the national CVCA.
  - The MRTD chip MAY 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 MRTD chip MUST update its current date using the Certificate Effective Date.
  - The MRTD chip MUST NOT grant the CVCA extended access to sensitive data (i.e. the effective access rights 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 MRTD chip MUST update its current date using the Certificate Effective Date.
  - The MRTD chip MUST NOT grant a DV extended access to sensitive data (i.e. the effective access rights SHOULD be ignored).
- The effective role is an IS:

- The certificate was issued by either a domestic or a foreign DV.
- If the certificate was issued by a domestic DV, the issuer is a trusted source of time and the MRTD chip MUST update its current date using the Certificate Effective Date.
- The MRTD chip MUST grant the **authenticated** IS extended access to sensitive data according to the effective access rights.

Appendix B. ISO 7816 Mapping (Normative)

In this Appendix the protocols for Chip Authentication and Terminal Authentication are mapped to ISO 7816 APDUs (Application Program Data Units) [10].

B.1. Chip Authentication

The following command SHALL be used to implement Chip Authentication: MSE:Set Kat.

B.1.1. MSE:Set KAT

Command			
CLA	0x0C	Secure Messaging with authenticated header, no chaining	
INS	0x22	Manage Security Environment	
P1/P2	0x41A6	Set for computation / Key Agreement Template	
Data	0x91	*Ephemeral Public Key* Ephemeral public key $\widetilde{PK_{PCD}}$ (cf. Section A.2) encoded as plain public key value.	REQUIRED
	0x84	*Reference of a private key* This data object is REQUIRED if the private key is ambiguous, i.e. more than one key pair is available for Chip Authentication (cf. Appendix A.2.1).	CONDITIONAL

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal operation* The key agreement operation was successfully performed. New session keys have been derived.
	0x6A80	*Incorrect Parameters in the command data field* The validation of the ephemeral public key failed. The previously established session keys remain valid.
	other	*Operating system dependent error* The previously established session keys remain valid.

B.2. Terminal Authentication

The following sequence of commands SHALL be used 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, IS Certificate).

B.2.1. MSE:Set DST

Command			
CLA	0x0C	Secure Messaging with authenticated header, no chaining	
INS	0x22	Manage Security Environment	
P1/P2	0x81B6	Set for verification: Digital Signature Template	
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.

B.2.2. PSO: Verify Certificate

Command			
CLA	0x0C	Secure Messaging with authenticated header, no chaining	
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).	

B.2.3. MSE:Set AT

Command			
CLA	0x0C	Secure Messaging with authenticated header, no chaining	
INS	0x22	Manage Security Environment	
P1/P2	0x81A4	Set for external authentication: Authentication Template	
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.

B.2.4. Get Challenge

Command		
CLA	0x0C	Secure Messaging with authenticated header, no chaining
INS	0x84	Get Challenge
P1/P2	0x0000	
Data	–	Absent
Le	0x08	

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

B.2.5. External Authenticate

Command			
CLA	0x0C	Secure Messaging with authenticated header, no chaining	
INS	0x82	External Authenticate	
P1/P2	0x0000	Keys and Algorithms implicitly known	
Data		Signature generated by the inspection system.	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.
	other	*Operating system dependent error* The authentication failed.

B.3. Secure Messaging

B.3.1. Send Sequence Counter

Only **after** a successful MSE:Set KAT command Secure Messaging is restarted using the new session keys derived from the key agreement operation, i.e.

- The old session keys and the old SSC are used to protect the response of the MSE:Set KAT command.
- The Send Sequence Counter is set to zero (SSC=0).
- The new session keys and the new SSC are used to protect subsequent commands/responses.

B.3.2. Secure Messaging Errors

The MRTD chip MUST abort Secure Messaging if and only if a Secure Messaging error occurs:

- If expected Secure Messaging data objects are missing, the MRTD chip SHALL respond with status bytes 0x6987.
- If Secure Messaging data objects are incorrect, the MRTD chip SHALL respond with status bytes 0x6988.

If Secure Messaging is aborted, the MRTD chip SHALL delete the session keys and reset the inspection system's access rights.

B.4. Reading Data Groups

The APDUs for selecting and reading EAC-protected data groups already specified by ICAO [5] SHALL be used (i.e. Select File and Read Binary). In accordance with ICAO specifications any unauthorized access to EAC-protected data groups SHALL be denied and the MRTD chip MUST respond with status bytes 0x6982 ("Security status not satisfied").

B.5. Command Flow

The sequence of ISO 7816 commands required to implement the Advanced Inspection Procedure described in Section 2.1 is illustrated in Figure B.1. In this example the MRZ (DG1), the facial image (DG2), and the fingerprints (DG3) are read from the MRTD chip. It is assumed that the LDS application is already selected and Basic Access Control was successfully performed.

B.6. Extended Length

Depending on the size of the cryptographic objects (e.g. public keys, signatures), APDUs with extended length fields MUST be used to send this data to the MRTD chip. For details on extended length see [10].

B.6.1. MRTD Chips

For MRTD chips support of extended length is CONDITIONAL. If the cryptographic algorithms and key sizes selected by the issuing state require the use of extended length, the MRTD chips SHALL support extended length. If the MRTD chip supports extended length this MUST be indicated in the ATR/ATS or in EF.ATR as specified in [10].

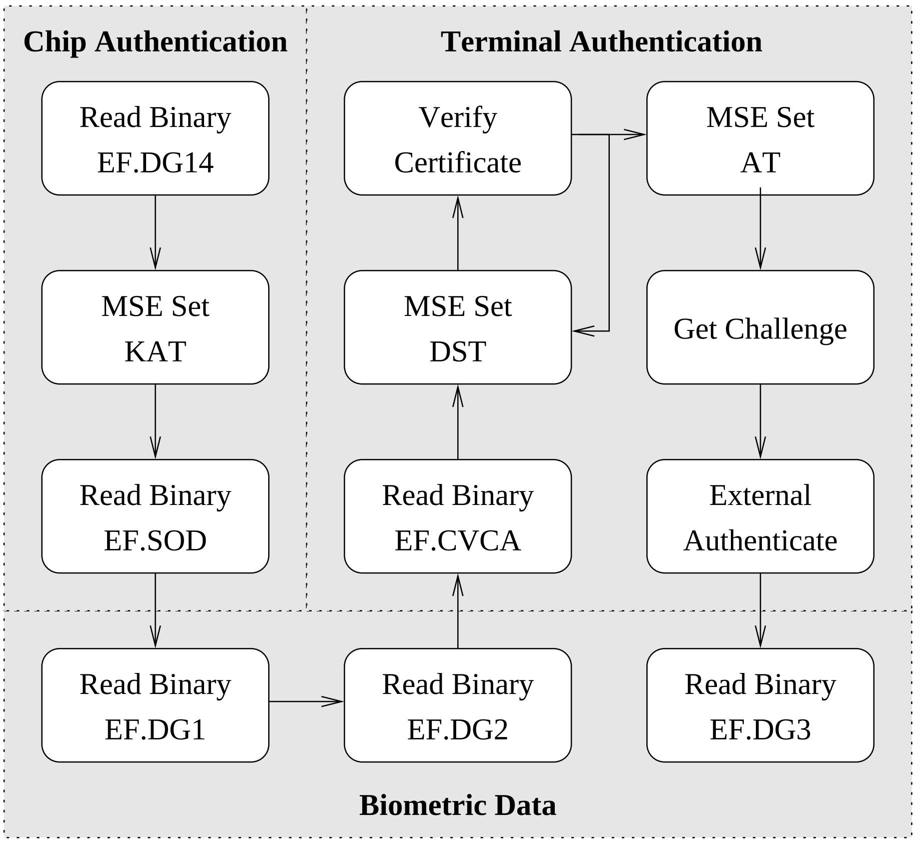

Figure B.1.: Command Flow

B.6.2. Inspection Systems

For inspection systems support of extended length is REQUIRED. An inspection system SHOULD examine whether or not support for extended length is indicated in the MRTD chip's ATR/ATS or in EF.ATR before using this option. The inspection system MUST NOT use extended length for APDUs other than the following commands unless the exact input and output buffer sizes of the MRTD chip are explicitly stated in the ATR/ATS or in EF.ATR.

- PSO:Verify Certificate
- MSE:Set KAT
- External Authenticate

B.6.3. Errors

The MRTD chip SHALL indicate extended length errors with status bytes 0x6700.

Appendix C.

DER Encoding (Normative)

The Distinguished Encoding Rules (DER) according to X.690 [13] SHALL be used to encode both ASN.1 data structures and (application specific) data objects. The encoding results in a Tag-Length-Value (TLV) structure as follows:

Tag: The tag is encoded in one or two octets and indicates the content.

Length: The length is encoded as unsigned integer in one, two, or three octets resulting in a maximum length of 65535 octets. The minimum number of octets SHALL be used.

Value: The value is encoded in zero or more octets.

C.1. ASN.1

The encoding of data structures defined in ASN.1 syntax is described in X.690 [13].

C.2. Data Objects

C.2.1. Overview

Table C.1 gives an overview on the tags, lengths, and values of the data objects used in this specification.

NOTE: The tag 0x7F4C is not yet defined by ISO/IEC 7816. The allocation is requested.

C.2.2. 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.

C.2.2.1. 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 0x00 octets MUST NOT be used.

NOTE: In contrast the ASN.1 type `INTEGER` is always a signed integer.

C.2.2.2. Elliptic Curve Points

The conversion of Elliptic Curve Points to octet strings is specified in [3]. The uncompressed format SHALL be used.

Table C.1.: Overview on Data Objects

Name	**Tag**	**Len**	**Value**	**Comment**
Object Identifier	0x06	V	Object Identifier	–
Discretionary Data	0x53	V	Octet String	Contains arbitrary data.
Public Key	0x7F49	V	Sequence	Nests the public key value and the domain parameters.
CV Certificate	0x7F21	V	Sequence	Nests certificate body and signature.
Certificate Body	0x7F4E	V	Sequence	Nests data objects of the certificate body.
Certificate Profile Identifier	0x5F29	1F	Unsigned Integer	Version of the certificate and certificate request format.
Certification Authority Reference	0x42	16V	Character String	Identifies the public key of the issuing certification authority in a certificate.
Certificate Holder Reference	0x5F20	16V	Character String	Associates the public key contained in a certificate with an identifier.
Certificate Holder Authorization Template	0x7F4C	V	Sequence	Encodes the role of the certificate holder (i.e. CVCA, DV, IS) and assigns read/write access rights to data groups storing sensitive data.
Certificate Effective Date	0x5F25	6F	Date	The date of the certificate generation.
Certificate Expiration Date	0x5F24	6F	Date	The date after which the certificate expires.
Signature	0x5F37	V	Octet String	Digital signature produced by an asymmetric cryptographic algorithm.
Authentication	0x67	V	Sequence	Nests certificate request with additional authentication data.

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

C.2.2.3. Dates

A date is encoded in 6 digits $d_1 \cdots d_6$ in the format YYMMDD using timezone GMT. It is converted to an octet string $o_1 \cdots o_6$ by encoding each digit d_j to an octet o_j as unpacked BCDs ($1 \leq j \leq 6$).

NOTE: The year YY is encoded in two digits and to be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

C.2.2.4. Character Strings

A character string $c_1 \cdots c_n$ is a concatenation of n characters c_j with $1 \leq j \leq n$. It SHALL be converted to an octet string $o_1 \cdots o_n$ by converting each character c_j to an octet o_j using the ISO/IEC 8859-1 character set. For informational purposes the character set can be found in Table C.3.

NOTE: 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.

Table C.3.: ISO/IEC 8859-1 Character Set

Code	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0																
1																
2	SP	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

SP: Space, NBSP: Non-breaking Space, SHY: Soft Hyphen

C.2.2.5. Octet Strings

An octet string $o_1 \cdots o_n$ is a concatenation of n octets o_j with $1 \leq j \leq n$. Every octet o_j consists of 8 bits.

C.2.2.6. Object Identifiers

An object identifier $i_1.i_2.\cdots.i_n$ is an ordered list of n unsigned integers i_j with $1 \leq j \leq n$. It SHALL be converted to an octet string $o_1 \cdots o_{n-1}$ using the following procedure:

1. The first two integers i_1 and i_2 are packed into a single integer i that is then converted to the octet string o_1. The value i is calculated as follows:

$$i = i_1 \cdot 40 + i_2$$

2. The remaining integers i_j are directly converted to octet strings o_{j-1} with $3 \leq j \leq n$.

More details on the encoding can be found in [13].

C.2.2.7. Sequences

A sequence $D_1 \cdots D_n$ is an ordered list of n data objects D_j with $1 \leq j \leq n$. The sequence SHALL be converted to a concatenated list of octet strings $O_1 \cdots O_n$ by DER encoding each data object D_j to an octet string O_j.

C.3. Public Key Data Objects

A public key data object is a sequence consisting 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.

Table C.4.: RSA Public Key

Data Object	Tag	Type	CV Certificate
Object Identifier	0x06	Object Identifier	m
Composite modulus n	0x81	Unsigned Integer	m
Public exponent e	0x82	Unsigned Integer	m

m: mandatory, o: optional (domain parameter)

Table C.5.: ECDSA Public Key

Data Object	Tag	Type	CV Certificate
Object Identifier	0x06	Object Identifier	m
Prime modulus p	0x81	Unsigned Integer	o
First coefficient a	0x82	Unsigned Integer	o
Second coefficient b	0x83	Unsigned Integer	o
Base point G	0x84	Elliptic Curve Point	o
Order of the base point r	0x85	Unsigned Integer	o
Public Point Y	0x86	Elliptic Curve Point	m
Cofactor f	0x87	Unsigned Integer	o

m: mandatory, o: optional (domain parameter)

- 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.

NOTE: Ephemeral public keys are only used in the MSE:Set KAT command (cf. Appendix B.1.1). As the public key format and the domain parameters are already known, only the plain public key value, i.e. **only** the context specific data object 0x91, is used to convey the ephemeral public key.

C.3.1. RSA Public Keys

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

C.3.2. ECDSA Public Keys

The data objects contained in an ECDSA public key are shown in Table C.5. The order of the data objects is fixed.

NOTE: In CV certificates the optional domain parameters MUST be either all present or all absent. Only CVCA Link Certificates MAY contain domain parameters. The domain parameters for DV and IS public keys are inherited from the respective CVCA public key and MUST be absent.

C.3.3. Ephemeral DH Public Keys

An ephemeral DH public key is an unsigned integer y contained in a context specific data object 0x91.

C.3.4. Ephemeral ECDH Public Keys

An ephemeral ECDH public key is an elliptic curve point Y contained in a context specific data object 0x91.

Appendix D.

Worked Examples (Informative)

This appendix provides worked examples for data structures (DG14 and CV certificates) defined by this specification.

NOTE: All numbers contained in the examples below are hexadecimal unless the notation is self-evident or explicitly stated.

D.1. Data Group 14

This section provides two examples for DG14. The first example is based on Elliptic Curve Diffie Hellman and the second example is based on Diffie Hellman.

D.1.1. ECDH-based Example

The hexdump of the encoded DG14 can be found in Figure D.1. The hexdump of the corresponding private key can be found in Figure D.2 at the end of the section.

D.1.1.1. General Structure

The general structure of the worked example is examined in the following table. Details on the encoding in the `AlgorithmIdentifier` and the `SubjectPublicKey` contained in the `SubjectPublicKeyInfo` are given in the subsequent sections.

Tag	Length	Value	ASN.1 Type	Comment
6E	82 01 49		-	Application specific tag “14”
31	82 01 45		SET	Set of SecurityInfos
30	82 01 22		SEQUENCE	SecurityInfo
06	09	04 00 7F 00 07 02 02 01 02	OBJECT IDENTIFIER	ChipAuthenticationPublicKeyInfo CA with ECDH
30	82 01 13		SEQUENCE	SubjectPublicKeyInfo
30	81 D4	see below	SEQUENCE	AlgorithmIdentifier
03	3A	see below	BIT STRING	SubjectPublicKey
–	–	–		optional keyId is unused
30	0E		SEQUENCE	SecurityInfo
06	09	04 00 7F 00 07 02 03 01 02	OBJECT IDENTIFIER	ChipAuthenticationInfo CA with ECDH
02	01	01	INTEGER	Version 1
–	–	–		optional keyId is unused
30	0D		SEQUENCE	SecurityInfo
06	08	04 00 7F 00 07 02 02 02	OBJECT IDENTIFIER	TerminalAuthenticationInfo
02	01	01	INTEGER	Version 1
–	–	–		optional FileID is unused

D.1.1.2. Encoding of the AlgorithmIdentifier

The `AlgorithmIdentifier` is not only used to identify the type of public key contained in the `SubjectPublicKeyInfo` but does also contain the domain parameters to be used. The encoding of the group generator as part of the domain parameter can be found in Section D.1.1.3.

Tag	Length	Value	ASN.1 Type	Comment			
30	81 D4		SEQUENCE	AlgorithmIdentifier			
06	07	2A 86 48 CE 3D 02 01	OBJECT IDENTIFIER		Elliptic Curve Public Key		
30	81 C8		SEQUENCE		Domain parameter		
02	01	01	INTEGER			Version	
30	28		SEQUENCE			Underlying field	
06	07	2A 86 48 CE 3D 01 01	OBJECT IDENTIFIER				Prime field
02	1D	00 D7 C1 ... C8 C0 FF	INTEGER				Prime p
30	3C		SEQUENCE			Curve equation	
04	1C	68 A5 E6 ... D2 9F 43	OCTET STRING				Parameter a
04	1C	25 80 F6 ... 6C 40 0B	OCTET STRING				Parameter b
–	–	–					Optional seed is unused
04	39	see below	OCTET STRING			Group generator G	
02	1D	00 D7 C1 ... A7 93 9F	INTEGER			Group order n	
02	01	01	INTEGER			Cofactor f	

D.1.1.3. Encoding of Elliptic Curve Points

The group generator as part of the elliptic curve domain parameters and the public key are both points on an elliptic curve. According to [3] elliptic curve points are not specified as ASN.1 data structures.

Encoding of the Group Generator: The group generator is part of the elliptic curve domain parameters contained in the `AlgorithmIdentifier`. The encoded elliptic curve point is embedded in an OCTET STRING.

Tag	Length	Value	ASN.1 Type	Comment	
04	39		OCTET STRING	Encoded group generator	
		04	–		Uncompressed point
		0D 90 29 ... 12 C0 7D	–		x-coordinate
		58 AA 56 ... 14 02 CD	–		y-coordinate

Encoding of the SubjectPublicKey: The public key contained in the `SubjectPublicKeyInfo` is an elliptic curve point embedded in a BIT STRING.

Tag	Length	Value	ASN.1 Type	Comment
03	3A		BIT STRING	Encoded public key
		00		Number of unused bits
		04	–	Uncompressed point
		68 0E C4 ... 46 4D 76	–	x-coordinate
		5C 38 C2 ... EB 21 3C	–	y-coordinate

D.1.1.4. Session Key Generation

The following table shows the ephemeral key pair randomly chosen by the inspection system and the shared secret resulting from the key agreement. Note that the domain parameters are omitted.

Private Key	`7756F0C5 D1AB06C0 03672668 2B720C2F B1D5F789 B58244A6 DC07E5A2`
Public Key	`69D489F6 8A99ABC8 7106B3E1 3A52C6AF 2C57CEE5 72755FE3 712C8AC3,` `8A6A3E9F E0694482 31BDC1BE FC826035 67E72602 EBA5C3EE EEAC3F15`
Shared Secret	`A770F66A CC78ED59 0581CC82 033C79F3 3BECE0A2 0C280244 79A4E97C`

Using the shared secret and the ICAO KDF, the following two session keys are derived:

K_{Enc}	`DF94ED65 8A5CCB35 5FFFE612 8BADB584`
K_{MAC}	`1297F052 5DD9DE75 917DCD90 848465C1`

D.1.1.5. Hashed Ephemeral Public Key

For ECDH the MRTD chip stores the x-coordinate of the ephemeral public key received from the inspection system.

$H(\widetilde{PK_{PCD}})$	`69D489F6 8A99ABC8 7106B3E1 3A52C6AF 2C57CEE5 72755FE3 712C8AC3`

D.1.2. DH-based Example

The hexdump of the encoded DG14 is shown in Figure D.3. The corresponding private key can be found in Figure D.4 at the end of this section.

D.1.2.1. General Structure

The general structure of the second worked example is examined in the following table. Details on the encoding of the `AlgorithmIdentifier` and the `SubjectPublicKey` contained in the `SubjectPublicKeyInfo` are given in the following sections.

Tag	Length	Value	ASN.1 Type	Comment
6E	82 01 DB		-	Application specific tag “14”
31	82 01 D7		SET	Set of SecurityInfos
30	82 01 B4		SEQUENCE	SecurityInfo
06	09	04 00 7F 00 07 02 02 01 01	OBJECT IDENTIFIER	ChipAuthenticationPublicKeyInfo CA with DH
30	82 01 A5		SEQUENCE	SubjectPublicKeyInfo
30	82 01 1A	see below	SEQUENCE	AlgorithmIdentifier
03	81 84	see below	BIT STRING	SubjectPublicKey
–	–	–	–	optional keyId unused
30	0E		SEQUENCE	SecurityInfo
06	09	04 00 7F 00 07 02 03 01 01	OBJECT IDENTIFIER	ChipAuthenticationInfo CA with DH

Tag	Length	Value	ASN.1 Type	Comment			
02	01	01	INTEGER				Version 1
–	–	–					optional keyId unused
30	0D		SEQUENCE			SecurityInfo	
06	08	04 00 7F 00 07 02 02 02	OBJECT IDENTIFIER				TerminalAuthenticationInfo
02	01	01	INTEGER				Version 1
–	–	–					optional FileID is unused

D.1.2.2. Encoding of the AlgorithmIdentifier

The `AlgorithmIdentifier` is not only used to identify the type of public key contained in the `SubjectPublicKeyInfo` but does also contain the domain parameters to be used.

Tag	Length	Value	ASN.1 Type	Comment		
30	82 01 1A		SEQUENCE	AlgorithmIdentifier		
06	09	2A 86 48 86 F7 0D 01 03 01	OBJECT IDENTIFIER		PKCS#3 dhKeyAgreement	
30	82 01 0B		SEQUENCE		Domain parameter	
02	81 81	00 DC B5 ... 59 CA D7	INTEGER			Prime *p*
02	81 80	2E 69 FE ... F0 52 10	INTEGER			Group generator *g*
02	02	03 F9	INTEGER			Private key length

D.1.2.3. Encoding of the SubjectPublicKey

The public key contained in the `SubjectPublicKey` is a DER encoded INTEGER embedded in a BIT STRING. In the example below the value of the BIT STRING is decoded. The decoded structure is put in brackets.

Tag	Length	Value	ASN.1 Type	Comment	
03	81 84		BIT STRING	Encoded public key	
		00			Number of unused bits
(02)	(81 80)	(55 3C E7 ... 72 48 A1)	(INTEGER)		Public key

D.1.2.4. Session Key Generation

The following table shows the ephemeral key pair randomly chosen by the inspection system and the shared secret resulting from the key agreement. Note that the domain parameters are omitted.

Private Key	`0170A377 AA4B612B 69A6762E CD71A91C 3D7CD149 A870F37 F357A196F F1134BF7`
	`E0B33DDC EC645560 54EA9959 23189BDB 3893656F E05F8DA BE67F8998 3799E16F`
	`9BF7A9CA 8050C949 31BAB4D8 CAA5F84B 33D71ACA 77A817C BC44CA92C 4B8960A2`
	`034FBC31 999E7DEE 025E1001 EAF96113 BD06EFED FBBD5F2 E916ADC73 1971F019`

Public Key	`8EBC4457 EABBEF83 65D6EF83 9A1A3672 449486B2 779EF88E B0198ADD C64A096B`
	`0AFC3C26 4D64EDFD F543C03B AEC7ED5D 58C2F2A1 4B63BB5D 280E62FE CAC0A6DD`
	`F255CEB9 2AB51C0D B672A251 68934F86 7B95552A 189A3244 4AF890B7 7509ED92`
	`EC5A81A7 D787F5F5 51B37EB2 FA3A49C7 787B5F61 3527649C 151C1786 8417E9CA`

Shared Secret	`C30AAE5F DC23EFF6 E477734A C318D32D F128AF25 2542087F 1FA239DD 3734DE5C`
	`C7E154ED 93BDEC78 E87CD691 6307976F B2603425 133A61D4 10F7F050 EA0797B3`
	`59A5009F 20C9BF0D 227C8866 B2C701FB 04ADF646 B138B1D6 D2623C17 AB3A910A`
	`5A2C72E6 2B554B3F B5B2C310 A1F3334E D4C6AA77 919EA912 5A147C64 9C9E6556`

Using the shared secret and the ICAO KDF, the following two session keys are derived:

K_{Enc}	`EFF63AC6 29184F19 99C69B7C 3BFA4F17`
K_{MAC}	`7AD463F3 6997CB2B CB3D1B88 2CE8E4A7`

D.1.2.5. Hashed Ephemeral Public Key

For DH the MRTD chip stores the SHA-1 hash of the ephemeral public key received from the inspection system.

$H(\widetilde{PK_{PCD}})$	`97D9AC36 0DCA6BB0 F2699B85 2DE37793 C29458CD`

D.2. Card Verifiable Certificates

This section provides two examples for CV certificates. Both examples are self-signed CVCA certificates. The first example is based on ECDSA and the second example is based on RSA.

D.2.1. Example 1: CVCA Certificate with ECDSA

The hexdump of the encoded certificate can be found in Figure D.5. The hexdump of the corresponding private key can be found in Figure D.6 at the end of the section.

D.2.1.1. General Structure

The general structure of the first worked example is examined in the following table. Details on the encoding of the public key and the Certificate Holder Authorization Template are given in the following sections.

Tag	Length	Value	Comment		
7F 21	82 01 89		CV Certificate		
7F 4E	82 01 49			Certificate Body	
5F 29	01	00			Certificate Profile Identifier
42	0E	44 45 43 ... 30 30 31			Certificate Authority Reference: DECVCAEPASS001
7F 49	81 FD	see below			Public key
5F 20	0E	64 65 63 ... 30 30 31			Certificate Holder Reference: DECVCAEPASS001
7F 4C	0E	see below			Certificate Holder Authorization Template
5F 25	06	00 07 00 04 00 01			Certificate Effective Date: 2007-APR-01
5F 24	06	00 09 00 04 00 01			Certificate Expiration Date: 2009-APR-01
5F 37	38	see below		Digital Signature: ECDSA in Plain Format	

D.2.1.2. Public Key

The encoding of the public key and the domain parameters contained in the certificate is explained in the following table. The corresponding private key is shown in Figure D.6.

Tag	Length	Value	Comment	
7F 49	81 FD		Public key	
06	0A	04 00 7F ... 02 02 02		Terminal Authentication with ECDSA-SHA-224
81	1C	D7 C1 34 ... C8 C0 FF		Prime modulus p
82	1C	68 A5 E6 ...		First coefficient a

Tag	Length	Value	Comment	
		D2 9F 43		
83	1C	25 80 F6 ... 6C 40 0B		Second coefficient *b*
84	39	04 0D 90 ... 14 02 CD		Base point *G*
85	1C	D7 C1 34 ... A7 93 9F		Order of the base point *r*
86	39	04 04 4F ... C6 6E F1		Public point *Y*
87	01	01		Cofactor *f*

D.2.1.3. Certificate Holder Authorization Template

The discretionary data object contains in the CHAT identifies a CVCA that allows read access to DG3.

Tag	Length	Value	Comment	
7F 4C	0E		Certificate Holder Authorization Template	
06	09	04 00 7F ... 01 02 01		Extended Access Control for ePassports
53	01	C1		Discretionary Data: CVCA granting access to DG3.

D.2.1.4. ECDSA Plain Signature

According to [3] ECDSA signatures in plain format are specified as direct concatenation of two octet strings $R||S$. For ECDSA-224, each octet string has length 28 (decimal).

R	`4CCF25C5 9F3612EE E18875F6 C5F2E2D2 1F039568 3B532A26 E4C189B7`
S	`1EFE659C 3F26E0EB 9AEAE998 63107F9B 0DADA164 14FFA204 516AEE2B`

NOTE: The X9.62 signature format based on ASN.1 MUST NOT be used.

D.2.2. Example 2: CVCA Certificate with RSA

The hexdump of the encoded certificate can be found in Figure D.7. The hexdump of the corresponding private key can be found in Figure D.8 at the end of the section.

D.2.2.1. General Structure

The general structure of the second worked example is examined in the following table. Details on the encoding of the public key are given in the following section. The encoding of the Certificate Holder Authorization Template is described in Appendix D.2.1.3.

Tag	Length	Value	Comment		
7F 21	82 02 6C		CV Certificate		
7F 4E	82 01 62			Certificate Body	
5F 29	01	00			Certificate Profile Identifier
42	0E	44 45 43 ... 30 30 31			Certificate Authority Reference: DECVCAEPASS001
7F 49	82 01 15	see below			Public key
5F 20	0E	44 45 43 ... 30 30 31			Certificate Holder Reference: DECVCAEPASS001
7F 4C	0E	see above			Certificate Holder Authorization Template
5F 25	06	00 07 00 04 00 01			Certificate Effective Date: 2007-APR-01
5F 24	06	00 09 00 04 00 01			Certificate Expiration Date: 2009-APR-01
5F 37	82 01 00	6D 7E E7 ... 99 19 DB		Digital Signature: RSA PKCS#1 version 1.5	

D.2.2.2. Public Key

The encoding of the public key and the domain parameters contained in the certificate is explained in the following table. The corresponding private key (private exponent d) is show in Figure D.8.

Tag	Length	Value	Comment	
7F 49	82 01 15		Public key	
06	0A	04 00 7F... 02 01 02		Terminal Authentication with RSA-v1_5-SHA-256
81	82 01 00	82 72 E1 ... B4 E3 15		Composite Modulus n
82	03	01 00 01		Exponent e

D.3. Encoding of the Document Number

The MRTD chip's document number is used as ID_{PICC} in Terminal Authentication. Let the document number be "123456789", thus, the check digit is "7". The corresponding Octet String to be used in Terminal Authentication is 3132333435363738393937.

Figure D.1.: DG14 (ECDH)

```
0000 :  6E820149 31820145 30820122 06090400 7F000702 02010230 82011330 81D40607
0020 :  2A8648CE 3D020130 81C80201 01302806 072A8648 CE3D0101 021D00D7 C134AA26
0040 :  4366862A 18302575 D1D787B0 9F075797 DA89F57E C8C0FF30 3C041C68 A5E62CA9
0060 :  CE6C1C29 9803A6C1 530B514E 182AD8B0 042A59CA D29F4304 1C2580F6 3CCFE441
0080 :  38870713 B1A92369 E33E2135 D266DBB3 72386C40 0B043904 0D9029AD 2C7E5CF4
00A0 :  340823B2 A87DC68C 9E4CE317 4C1E6EFD EE12C07D 58AA56F7 72C0726F 24C6B89E
00C0 :  4ECDAC24 354B9E99 CAA3F6D3 761402CD 021D00D7 C134AA26 4366862A 18302575
00E0 :  D0FB98D1 16BC4B6D DEBCA3A5 A7939F02 0101033A 0004680E C4FF3851 12D9A401
0100 :  76D36733 157B11FC 08B4A280 CE9B8246 4D765C38 C21CB883 6EE05724 3C1EBC7B
0120 :  B80EC484 41107C38 E4F545EB 213C300E 06090400 7F000702 03010202 0101300D
0140 :  06080400 7F000702 02020201 01......................................
```

Figure D.2.: Chip Authentication Private Key (ECDH)

```
0000 :  12528622 D8947E85 E4988853 69ECDCAB F10E343A F7B95A99 DF610031
```

Figure D.3.: DG14 (DH)

```
0000 :  6E8201DB 318201D7 308201B4 06090400 7F000702 02010130 8201A530 82011A06
0020 :  092A8648 86F70D01 03013082 010B0281 8100DCB5 54DF8C69 31E865C1 B588273D
0040 :  80A2D87A B539C5E4 A074B402 49FF655A 9AB83063 3B457C4C F885E31C D79F8114
0060 :  8C8A68D1 DBFC2F7B 70ED55C0 387C23A0 479A9572 E8A6714F 418A6BF9 B00EC5BC
0080 :  4DEF255A 9485058A 4271008B A694AA62 CC18385E F9D7B6E8 33A7088A C817AA1F
00A0 :  9B93A86B 983EAB73 C15884E7 33665659 CA7D0281 802E69FE 94D3C0A4 378C8A47
00C0 :  9D83091A ED419234 25C10300 8C6AB3F6 E83E20CB 16C4AE0B 0E28ED9B C79CD7D7
00E0 :  E9DFD39D D0A39141 F2DD5714 9AB688DB AD177C68 6F771828 E5A04408 512F1564
0100 :  74B0BFD4 30CBBF91 C01589E7 21DDDFFC DF450043 EB771E61 084C597F 7AEA9048
0120 :  420A2180 EBFEC1B3 B93C1A6C B1AD38B3 984FF052 10020203 F9038184 00028180
0140 :  553CE735 ECF5CBF2 029D30FA A4F97335 DF404047 E4F8586D 76A7D221 A09E7F55
0160 :  BBE255C6 587BF288 5D41B786 BCEF2177 D52BF3CD BA785D37 D70B88D6 AB4E1CA6
0180 :  6A63B601 1376ED44 444A662B D0DC9524 176E9712 87AD41D2 9BED3D35 EAC7D39C
01A0 :  A73ECB2A 3B4D3967 1CE4125C 92658C5B F3DEDA91 5ED71B88 FC031BAB 887248A1
01C0 :  300E0609 04007F00 07020301 01020101 300D0608 04007F00 07020202 020101..
```

Figure D.4.: Chip Authentication Private Key (DH)

```
0000 :  01CD4A70 FFDC3D42 C862FD5E 3D781EB6 DE97677B 4FF61319 242E1499 B5CD1908
0020 :  A9B54221 135D1EDB AD787A5C E37586CF E86A61C4 78187157 267C97B4 0A7F2727
0040 :  B9B92FAC EC267CC0 1C883FA3 783BA07D C090EE04 99C9CE88 C684C874 0FEB84F4
0060 :  49B0F544 C1747716 46BDB7A3 0B0E3AB5 6C655F0E 83A98A4B A99EB9F1 0B0C0FBF
```

Figure D.5.: CVCA Certificate (ECDSA)

```
0000 :  7F218201 897F4E82 01495F29 0100420E 44454356 43414550 41535330 30317F49
0020 :  81FD060A 04007F00 07020202 0202811C D7C134AA 26436686 2A183025 75D1D787
0040 :  B09F0757 97DA89F5 7EC8C0FF 821C68A5 E62CA9CE 6C1C2998 03A6C153 0B514E18
0060 :  2AD8B004 2A59CAD2 9F43831C 2580F63C CFE44138 870713B1 A92369E3 3E2135D2
0080 :  66DBB372 386C400B 8439040D 9029AD2C 7E5CF434 0823B2A8 7DC68C9E 4CE3174C
00A0 :  1E6EFDEE 12C07D58 AA56F772 C0726F24 C6B89E4E CDAC2435 4B9E99CA A3F6D376
00C0 :  1402CD85 1CD7C134 AA264366 862A1830 2575D0FB 98D116BC 4B6DDEBC A3A5A793
00E0 :  9F863904 393EE8E0 6DB6C7F5 28F8B426 0B49AA93 309824D9 2CDB1807 E5437EE2
0100 :  E26E29B7 3A711153 0FA86B35 0037CB94 15E15370 43944637 97139E14 8701015F
0120 :  200E4445 43564341 45504153 53303031 7F4C0E06 0904007F 00070301 02015301
0140 :  C15F2506 00070004 00015F24 06000900 0400015F 37384CCF 25C59F36 12EEE188
0160 :  75F6C5F2 E2D21F03 95683B53 2A26E4C1 89B71EFE 659C3F26 E0EB9AEA E9986310
0180 :  7F9B0DAD A16414FF A204516A EE2B....................................
```

Figure D.6.: CVCA Private Key (ECDSA)

```
0000 :  867B8CC4 3D5B55DD 6907F215 74B94F37 466769C9 0662C9A0 A5A2C3E3
```

Figure D.7.: CVCA Certificate (RSA)

```
0000 :  7F218202 6C7F4E82 01625F29 0100420E 44454356 43414550 41535330 30317F49
0020 :  82011506 0A04007F 00070202 02010281 82010082 72E1A553 8FC15183 9BB97AD7
0040 :  205DB251 D6AC4495 DA2C5035 6F768961 190F7DB2 77908CC9 7D55DB8F F5C8C3D7
0060 :  4FBBBAF6 901C52E1 5E4A8C7C 3C46BE4C 8DC5103A 497532B2 7B8257C0 8236426D
0080 :  73FEC79E CD165C93 33C1D938 661D1512 94F0AD31 FF6258C7 A4C7D26C 4928EBEA
00A0 :  D4E60E95 97740E57 354F98CC 1571289C 899FF786 F3E36246 D04540AA 575FB33C
00C0 :  E04CD130 E4418889 C21D97B6 18245AB6 53B9A1EF 91DEDFB5 02A77377 CDE1BA26
00E0 :  C35B76D6 C0A96102 2E978928 C73972F8 6276FC53 D775C991 9AD0F2EF 383569BF
0100 :  3EAD7992 123ABB93 B5DDDA6A 1F87AB13 7F7FE5F4 4F379CEF A08C19D0 2C456FF4
0120 :  A375D786 C87F44AF 1F38D155 8381870C B4E31582 03010001 5F200E44 45435643
0140 :  41455041 53533030 317F4C0E 06090400 7F000703 01020153 01C15F25 06000700
0160 :  0400015F 24060009 00040001 5F378201 006D7EE7 DA282000 F534FBA0 41104AF6
0180 :  0FD52FD6 1A51E8A2 91B8C0E2 DD4A2D60 2231438D 1D7BFA9D FF925670 1C335A08
01A0 :  A1E36929 8479EF20 56A3AA6E 5D011685 BC401C66 B42EA8A7 BF65D452 E5BC4D54
01C0 :  ECB33ED7 8504EE96 657CE280 52C98B83 29C5C0BF 7B3F51FF 04DE46D4 78C722B3
01E0 :  7AB925EB 7E22E85D E5D60017 F497F911 C88BDA6E 55037CDB BC712E5E A9FF2234
0200 :  61487A3C 6A68BDED 9499B537 4BFB8A15 B9A3CE83 7C0FAFFE 4CFD402C D7B59CC6
0220 :  B2F41557 2A03A067 9B5D0D40 EE567054 CEB22C2B 7A79FBD4 ECD20AC4 8292AB96
0240 :  8D9FFFA7 85FD938E AE408559 C8769104 6412B9E7 F6908FB5 D5DA4736 EC314C2E
0260 :  70138DC1 7F6631B5 BD53B8FD AEA59919 DB..............................
```

Figure D.8.: CVCA Private Key (RSA)

```
0000 :  710C5406 6D8B2852 97066E50 497CB016 681BED40 A6C8E31C 43A2AC0C 544C5E83
0010 :  65157EFC B364DDF2 2029356A 9AFE6B47 9483C138 063D2BC7 E1A66EFC 5DE58281
0020 :  14B23251 D4217B0C C7355106 724A52E2 F5F8EFC7 1C7E7F20 DB628B13 86FDF231
0040 :  7621C73D FF0E0B69 8D143DD5 6DB64105 24989328 166589EC 75F8A30F 39D17F01
00A0 :  C2E0DD3B 256632B1 87DC22D3 D1283BCD 12138082 8CA53BBA 524072A4 927D95AE
00C0 :  8AAA9B87 F84BA6C1 6359B553 0BD2132C B565AF78 19AB48CF 0022562F E41465C0
00E0 :  0C3A3EC2 7B906A61 EEEA382F 3BFA5A40 FA76F6DC 8729B965 EE7D3B32 BEDF5783
0100 :  C10E714B E5A6D148 18087711 284E04FC 5E94D43A 2C701DFD 97E4C0D1 22F35FE5
```

Appendix E.

Basic Access Control (Informative)

The protocol for Basic Access Control is specified by ICAO [5]. Basic Access Control checks that the inspection system has *physical* access to the MRTD's data page. This is enforced by requiring the inspection system to derive an authentication key from the *optically* read MRZ of the MRTD. The protocol for Basic Access Control is based on ISO/IEC 11770-2 [6] key establishment mechanism 6. This protocol is also used to generate session keys that are used to protect the confidentiality (and integrity) of the transmitted data.

E.1. Document Basic Access Keys

The Document Basic Access Keys KB_{Enc} and KB_{MAC} stored on the RF-chip in secure memory, have to be derived by the inspection system from the MRZ of the MRTD prior to accessing the RF-chip. Therefore, the inspection system optically reads the MRZ and generates the Document Basic Access Keys by applying the ICAO KDF [5] to the most significant 16 bytes of the SHA-1 [15] hash of some fields of the MRZ. As reading the MRZ optically is error-prone, only the fields protected by a check-digit are used to generate the Basic Access Key(s): Document Number, Date of Birth, and Date of Expiry. As a consequence the resulting authentication key has a relatively low entropy. The actual entropy mainly depends on the type of the Document Number. For 10 year valid travel document the **maximum** strength of the authentication key is approximately:

- 56 Bit for a numeric Document Number ($365^2 \cdot 10^{12}$ possibilities)
- 73 Bit for an alphanumeric Document Number ($365^2 \cdot 36^9 \cdot 10^3$ possibilities)

NOTE: Especially in the second case this estimation requires the Document Number to be randomly and uniformly chosen. Depending on the knowledge of the attacker, the actual entropy of the Document Basic Access Key may be lower, e.g. if the attacker knows all Document Numbers in use or is able to correlate Document Numbers and Dates of Expiry.

Given that in the first case the maximum entropy (56 Bit) is relatively low, calculating the authentication key from an eavesdropped session is possible. On the other hand, this still requires more effort than to obtain the same (less-sensitive) data from another source.

E.2. Protocol Specification

Basic Access Control is shown in Figure E.1. For better readability encryption and message authentication are combined into a single authenticated encryption primitive

$$\mathbf{E}_K(S) = \mathbf{E}'_{KB_{Enc}}(S)||\mathbf{MAC}_{K_{MAC}}(\mathbf{E}'_{KB_{Enc}}(S)),$$

where $K = \{KB_{Enc}, KB_{MAC}\}$.

1. The MRTD chip sends the nonce r_{PICC} to the inspection system.

MRTD Chip (PICC)		**Inspection System (PCD)**
		read MRZ optically and derive K
choose r_{PICC} and K_{PICC} randomly		choose r_{PCD} and K_{PCD} randomly
	$\xrightarrow{r_{PICC}}$	
	$\xleftarrow{e_{PCD}}$	$e_{PCD} = \mathbf{E}_K(r_{PCD} \Vert r_{PICC} \Vert K_{PCD})$
$r'_{PCD} \Vert r'_{PICC} \Vert K'_{PCD} = \mathbf{D}_K(e_{PCD})$		
check $r'_{PICC} = r_{PICC}$		
$e_{PICC} = \mathbf{E}_K(r_{PICC} \Vert r'_{PCD} \Vert K_{PICC})$	$\xrightarrow{e_{PICC}}$	
		$r'_{PICC} \Vert r''_{PCD} \Vert K'_{PICC} = \mathbf{D}_K(e_{PICC})$
		check $r''_{PCD} = r_{PCD}$

Figure E.1.: Basic Access Control

2. The inspection system sends the encrypted challenge

$$e_{PCD} = \mathbf{E}_K(r_{PCD}||r_{PICC}||K_{PCD})$$

to the MRTD chip, where r_{PICC} is the MRTD chip's nonce, r_{PCD} is the inspection system's randomly chosen nonce, and K_{PCD} is keying material for the generation of the session keys.

3. The MRTD chip performs the following actions:

   a) It decrypts the received challenge to $r'_{PCD}||r'_{PICC}||K'_{PCD} = \mathbf{D}_K(e_{PCD})$ and verifies that $r'_{PICC} = r_{PICC}$.

   b) It responds with the encrypted challenge $e_{PICC} = \mathbf{E}_K(r_{PICC}||r'_{PCD}||K_{PICC})$, where r_{PICC} is the MRTD chip's randomly chosen nonce and K_{PICC} is keying material for the generation of the session keys.

4. The inspection system decrypts the encrypted challenge to $r'_{PICC}||r''_{PCD}||K'_{PICC} = \mathbf{D}_K(e_{PICC})$ and verifies that $r''_{PCD} = r_{PCD}$.

After a successful authentication all further communication MUST be protected by Secure Messaging in Encrypt-then-Authenticate mode using session keys K_{Enc} and K_{MAC} derived according to [5] from the common master secret $K_{Master} = K_{PICC} \oplus K_{PCD}$ and a Send Sequence Counter SSC derived from r_{PICC} and r_{PCD}.

Appendix F.

Challenge Semantics (Informative)

Consider a signature based challenge-response protocol between an MRTD chip (PICC) and an inspection system (PCD), where the MRTD chip wants to prove knowledge of its private key SK_{PICC}:

1. The inspection system sends a randomly chosen challenge c to the MRTD chip.
2. The MRTD chip responds with the signature $s = \mathbf{Sign}(SK_{PICC}, c)$.

While this is a very simple and efficient protocol, the MRTD chip in fact signs the message c without knowing the semantic of this message. As signatures provide a transferable proof of authenticity, any third party can – in principle – be convinced that the MRTD chip has indeed signed this message.

Although c should be a random bit string, the inspection system can as well generate this bit string in an unpredictable but (publicly) verifiable way, e.g. let SK_{PCD} be the inspection system's private key and

$$c = \mathbf{Sign}(SK_{PCD}, ID_{PICC}||\text{Date}||\text{Time}||\text{Location}),$$

be the challenge generated by using a signature scheme with message recovery. The signature guarantees that the inspection system has indeed generated this challenge. Due to the transferability of the inspection system's signature, any third party having trust in the inspection system and knowing the corresponding public key PK_{PCD} can check that the challenge was created correctly by verifying this signature. Furthermore, due to the transferability of MRTD chip's signature on the challenge, the third party can conclude that the assertion became true: The MRTD chip was indeed at a certain date and time at a certain location.

On the positive side, countries may use Challenge Semantics for their internal use, e.g. to prove that a certain person indeed has immigrated. On the negative side such proves can be misused to track persons. In particular since Active Authentication is not restricted to authorized inspection systems misuse is possible. The worst scenario would be MRTD chips that provide Active Authentication without Basic Access Control. In this case a very powerful tracking system may be set up by placing secure hardware modules at prominent places. The resulting logs cannot be faked due to the signatures. Basic Access Control diminishes this problem to a certain extent, as interaction with the bearer is required. Nevertheless, the problem remains, but is restricted to places where the travel document of the bearer is read anyway, e.g. by airlines, hotels etc.

One might object that especially in a contactless scenario, challenges may be eavesdropped and reused at a different date, time or location and thus render the proof at least unreliable. While eavesdropping challenges is technically possible, the argument is still invalid. By assumption an inspection system is trusted to produce challenges correctly and it can be assumed that it has checked the MRTD chip's identity before starting Active Authentication. Thus, the eavesdropped challenge will contain an identity different from the prover's identity who signs the challenge.

Bibliography

[1] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *30th Annual ACM Symposium on the Theory of Computing*, pages 419–428. ACM Press, 1998.

[2] Scott Bradner. Key words for use in RFCs to indicate requirement levels. RFC 2119, 1997.

[3] BSI. Technical Guideline: Elliptic Curve Cryptography (ECC) based on ISO 15946, Version 1.0. TR-03111, 2007.

[4] Russel Housley, Tim Polk, Warwick Ford, and David Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 3280, 2002.

[5] ICAO Doc 9303. Specifications for electronically enabled passports with biometric identification capabilities. In *Machine Readable Travel Documents – Part 1: Machine Readable Passport*, volume 2. ICAO, 6th edition, 2006.

[6] ISO 11770-2. Information technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques, 1996.

[7] ISO 15946-1. Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General, 2002.

[8] ISO 15946-2. Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital signatures, 2002.

[9] ISO 15946-3. Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 3: Key establishment, 2002.

[10] ISO/IEC 7816-4:2005. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange, 2005.

[11] ISO/IEC 7816-6:2004. Identification cards – Integrated circuit cards – Part 6: Interindustry data elements for interchange, 2004.

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

[13] ITU-T. Information Technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). X.690, 2002.

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

[15] NIST. Secure hash standard. FIPS PUB 180-2 (+ Change Notice to include SHA-224), 2002.

[16] Eric Rescorla. Diffie-Hellman key agreement method. RFC 2631, 1999.

[17] RSA Laboratories. PKCS#3: Diffie-Hellman key-agreement standard. RSA Laboratories Technical Note, 1993.

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