MACHINE READABLE TRAVEL DOCUMENTS

TECHNICAL REPORT

Supplemental Access Control for Machine Readable Travel Documents

Version – 1.1

Date – 15 April 2014

Published by authority of the Secretary General

ISO/IEC JTC1 SC17 WG3/TF5

FOR THE

INTERNATIONAL CIVIL AVIATION ORGANIZATION

File	: 2014_04_15 TR-SAC 1_1.odt
Author	: ISO/IEC JTC1 SC17 WG3/TF5

Technical Report

Supplemental Access Control for Machine Readable Travel Documents

Release : 1.1
Date : 15 April 2014

Release Control

Release	*Date*	*Description*
1.00	2010-03-23	Initial public version with temporary restrictions on the usage of the elliptic curve integrated mapping.
1.01	2010-11-11	Removed restrictions on usage of the elliptic curve integrated mapping.
1.02	2011-03-08	Fixed bugs in the specification of the integrated mapping.
1.1	2014-04-15	Integration of Chip Authentication and Chip Authentication Mapping

Table of Contents

1 Introduction

This Technical Report specifies access control mechanisms that are supplementary to the mechanisms specified in [9], [10].

1.1 Password Authenticated Connection Establishment

This mechanism comprises Password Authenticated Connection Establishment (PACE) as specified in [5] and complementary contributions [8], [11], [4], [2] and [25]. This framework based on PACE v2 allows for various implementation options (cf. Section 3, e.g. mappings, algorithms, passwords, etc.), this specification fixes choices for the implementation in Machine Readable Travel Documents. A versioning of PACE is defining the evolutions that allow to the implementation of the initial or complete specifications:

- PACE v1 refers to the initial specification in TR-03110 v2.0 defining the generic mapping and
- PACE v2 refers to the extended version with complementary specifications for the generic and integrated mapping.

Throughout this document, the term PACE refers to PACE v2.

Note: *Although this document focuses on MRTDs/MRtds, PACE may also be used in other technology and/or application contexts. The free license for the variant using the elliptic curve integrated mapping [25] may only be available for implementations in the context of ISO/IEC 7501.*

1.1.1 Background

Doc 9303 [9], [10] introduces Basic Access Control as an OPTIONAL access control mechanism as follows:

Comparing a MRTD/MRtd that is equipped with a contactless chip with a traditional MRTD/MRtd shows two differences:

- *The data stored in the chip can be electronically read without opening the document (skimming).*
- *The communication between a chip and a reader, that is unencrypted, can be eavesdropped in a distance of several meters.*

While there are physical measures possible against skimming these don't address eavesdropping. Therefore, it is understood that States MAY choose to implement a Basic Access Control mechanism, i.e. an access control mechanism that requires the consent of the bearer of the MRTD that the data stored in the chip to be read in a secure way. This Basic Access Control Mechanism prevents skimming as well as eavesdropping.

This access control mechanism is OPTIONAL. Descriptions and specifications in this Technical Report on Basic Access Control and Secure Messaging only apply for MRTDs/MRtds and Inspection Systems that support this option. If supported, this mechanism MUST ensure that the contents of the chip can only be read after the bearer has willingly offered his MRTD/MRtd.

A chip that is protected by the Basic Access Control mechanism denies access to its contents unless the inspection system can prove that it is authorized to access the chip. This proof is given in a

challenge-response protocol, where the inspection system proves knowledge of the chip-individual Document Basic Access Keys (K_{ENC} and K_{MAC}) which are derived from information from the MRZ.

The inspection system MUST be provided with this information prior to reading the chip. The information has to be retrieved optically/visually from the MRTD/MRtd (e.g. from the MRZ). It also MUST be possible for an inspector to enter this information manually on the inspection system in case machine-reading of the MRZ is not possible.

Additionally, after the inspection system has been authenticated successfully, it is REQUIRED that the chip enforces encryption of the communication channel between the inspection system and the MRTD's/MRtd's chip by Secure Messaging techniques.

Assuming that the Document Basic Access Keys (K_{ENC} and K_{MAC}) cannot be obtained from a closed document (since they are derived from the optically read MRZ), it is accepted that the passport was willingly handed over for inspection. Due to the encryption of the channel, eavesdropping on the communication would require a considerable effort.

1.1.2 Operational experiences

Due to its simplicity Basic Access Control turned out to be a very successful protocol and it is implemented in almost every ePassport. Thus, the OPTIONAL Basic Access Control is now a RECOMMENDED feature for privacy protection.

The security provided by Basic Access Control is limited by the design of the protocol. The Document Basic Access Keys (K_{ENC} and K_{MAC}) are generated from printed data with very limited randomness. The data that is used for the generation of the keys are Document Number, Date of Birth, and Date of Expiry. As a consequence the resulting keys have a relatively low entropy and are cryptographically weak. The actual entropy mainly depends on the type of the Document Number. For 10 year valid travel document the **maximum** strength of the keys 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)

Especially in the second case this estimation requires the Document Number to be randomly and uniformly chosen which is usually not the case. 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.

1.1.3 Supplemental Access Control

There is no straightforward way to strengthen Basic Access Control as its limitations are inherent to the design of the protocol based on symmetric ("secret key") cryptography. A cryptographically strong access control mechanism must (additionally) use asymmetric ("public key") cryptography.

This Technical Report specifies PACE v2 as an access control mechanism that is supplemental to Basic Access Control. PACE MAY be implemented in addition to Basic Access Control, i.e.

- States MUST NOT implement PACE without implementing Basic Access Control if global interoperability is required.
- Inspection Systems SHOULD implement and use PACE if provided by the MRTD chip.

Note: *Basic Access Control will remain the "default" access control mechanism for globally interoperable machine readable travel documents as long as Basic Access Control provides sufficient security. Basic Access Control may however become deprecated in the future. In this case PACE will become the default access control mechanism.*

The inspection system SHALL use either BAC or PACE but not both in the same session.

1.2 Chip Authentication

As an additional protocol, this Technical Report specifies the Chip Authentication protocol for Machine Readable Travel Documents.

Chip Authentication Version 1 (designated Chip Authentication for short throughout this Report, originally specified in version 1.00 of [5] and extended by support of AES in version 2.00 of [5]) is an alternative to the optional ICAO Active Authentication, i.e. it enables the terminal to verify that the MRTD chip is genuine, based on a static key pair stored in the chip. Additionally, Chip Authentication establishes Secure Messaging between the MRTD chip and the terminal.

The main differences to Active Authentication are:

- 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.

Details on Challenge Semantics are described in Appendix B

Additionally, this version of the Technical Report defines a new Mapping for PACE, the Chip Authentication Mapping, which extends the Generic Mapping. This Mapping combines PACE and Chip Authentication into one protocol PACE-CAM, which allows faster execution than the separate protocols.

1.3 Requirements for MRTD Chips and Terminals

This Technical Report specifies requirements for implementations of MRTD chips and terminals. While MRTD chips must comply with those requirements according to the terminology described in Section 1.5.1, requirements for terminals are to be interpreted as guidance, i.e. interoperability of MRTD chip and terminal are only guaranteed if the terminal complies with those requirements, otherwise the interaction with the MRTD chip will either fail or the behavior of the MRTD chip is undefined. In general, the MRTD chip need not enforce requirements related to terminals unless the security of the MRTD chip is directly affected.

1.4 Assumptions

It is assumed that the reader is familiar with the concepts and mechanisms offered by asymmetric cryptography.

It is assumed that the reader is familiar with the contents of [9], [10] and any other official documents issued by ICAO regarding Machine Readable Travel Documents.

1.5 Terminology

1.5.1 Technical Report Terminology

The key words "MUST", “MUST NOT”, "SHALL", “SHALL NOT”, "REQUIRED", "SHOULD", “SHOULD NOT”, "RECOMMENDED", "MAY", and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [3].

In case OPTIONAL features are implemented, they MUST be implemented as described in this Technical Report.

Note: *Notes are part of the normative text and used to emphasise requirements or additional information.*

1.5.2 Notations

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

1.5.2.1 Hash Algorithms

The operation for computing a cryptographic hashis described in an algorithm-independent way.

- The operation for computing a hash over a message m is denoted by $\mathbf{H}(m)$.

1.5.2.2 Symmetric Key Algorithms

The keys and operations for symmetric key encryption and authentication are described in an algorithm-independent way.

Symmetric keys are derived from a shared secret K or from a password π using a Key Derivation Function (KDF):

- Deriving a key for message encryption is denoted by $KS_{Enc} = \mathbf{KDF_{Enc}}(K)$.
- Deriving a key for message authentication is denoted by $KS_{MAC} = \mathbf{KDF_{MAC}}(K)$.
- Deriving a key from a password is denoted by $K_{\pi} = \mathbf{KDF_{\pi}}(\pi)$.

The operations for encrypting and decrypting a message are denoted as follows:

- Encrypting a plaintext m with key Key_{Enc} is denoted by $c = \mathbf{E}(Key_{Enc}, m)$.
- Decrypting a ciphertext c with key Key_{Enc} is denoted by $m = \mathbf{D}(Key_{Enc}, c)$.

The operation for computing an authentication code T on message m with key Key_{MAC} is denoted as $T = \mathbf{MAC}(Key_{MAC}, m)$.

1.5.2.3 Key Agreement

The keys and operations for key agreement are described in an algorithm-independent way.

The following key pairs are used for PACE and Chip Authentication:

- For PACE both the MRTD chip and the terminal generate ephemeral Diffie-Hellman key pairs based on the ephemeral domain parameters D_{Map}.
  - The chip's ephemeral public key is $PK_{DH,PICC}$, the corresponding private key is $SK_{DH,PICC}$.
  - The terminal's ephemeral public key is $PK_{DH,PCD}$, the corresponding private key is $SK_{DH,PCD}$.
- For Chip Authentication the MRTD chip uses a static Diffie-Hellman key pair and the terminal generates an ephemeral key pair based on the MRTD chip's static domain parameters D_{PICC}.
  - The chip's static public key is PK_{PICC}, the corresponding private key is SK_{PICC}.
  - The terminal's ephemeral public key is $PK_{DH,PCD}$, the corresponding private key is $SK_{DH,PCD}$.

It is RECOMMENDED that the MRTD chip validates public keys received from the terminal.

Note: *The terminal will have to use different ephemeral public keys for PACE and Chip Authentication. As the ephemeral keys are context specific, the same notation is used. Ephemeral keys were noted by a ~ in previous version of this document. This was changed for clarity.*

The operation for generating a shared secret K is denoted by $K = \mathbf{KA}(SK, PK, D)$, where SK is an (ephemeral or static) secret key, PK is an (ephemeral or static) public key and D are the (ephemeral or static) domain parameters.

1.5.3 Passwords

The following passwords (and keys derived from passwords) are relevant within the scope of this Technical Report:

Name	***Abbreviation***	***Comments***
Document Basic Access Keys	K_{ENC}, K_{MAC}	Document Basic Access Keys are symmetric keys, both are derived from the MRZ
PACE Key	K_{π}	PACE keys are derived from a password (CAN or MRZ).
Card Access Number	CAN	Password derived from a short number printed on the front side of the datapage.
Machine Readable Zone	MRZ	Password derived from the printed Machine Readable Zone as defined by Doc 9303.

2 Inspection procedure

The inspection procedure of the electronic part of an ICAO-compliant eMRTD consists of the following steps:

1. **Access eMRTD** **(REQUIRED)**

   See section 2.1

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

   The terminal MUST read and verify the Document Security Object, see Doc 9303 [9], [10].

3. **Authentication of the Chip** **(OPTIONAL)**

   See section 2.2.

4. **Read and authenticate data** **(OPTIONAL)**

   The terminal MAY read and verify read data groups, see Doc 9303 [9], [10].

2.1 Access eMRTD

When a MRTD with OPTIONAL Supplemental Access Control is offered to the inspection system, optically or visually read information is used to derive a PACE Key K_π to gain access to the chip and to set up a secure channel for communications between the MRTD chip and the inspection system.

An MRTD chip that supports Supplemental Access Control SHALL respond to unauthenticated read attempts (including selection of (protected) files in the LDS) with “Security status not satisfied” (0x6982). To authenticate the inspection system the following steps SHALL be performed:

The ePassport application MUST be opened as part of the ePassport inspection procedure. Opening the ePassport application consists of selecting the ePassport application and performing access control as required by the MRTD chip, i.e. Basic Access Control or PACE. If the MRTD chip supports both PACE and Basic Access Control the inspection system SHOULD use PACE instead of Basic Access Control.

The opening procedure consists of the following steps, the master file being selected as a precondition:

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

   The inspection system SHALL read the file CardAccess (cf. Section 5.4) to determine the parameters (i.e. symmetric ciphers, key agreement algorithms, domain parameters, and mappings) supported by the MRTD chip. The inspection system may select any of those parameters.

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

   If the file CardAccess is not available, the inspection system MAY try to read the ePassport with Basic Access Control.

2. **PACE** **(CONDITIONAL)**

   This step is RECOMMENDED if PACE is supported by the MRTD chip.

- The inspection system SHOULD derive the key K_π from the MRZ. It MAY use the CAN instead of the MRZ if the CAN is known to the inspection system.
- The MRTD chip SHALL accept the MRZ as passwords for PACE. It MAY additionally accept the CAN.
- The inspection system and the MRTD chip mutually authenticate using K_π and derive session keys KS_{ENC} and KS_{MAC}. The PACE protocol as described in Section 3 SHALL be used.

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. **Select ePassport Application** **(REQUIRED)**
4. **Basic Access Control** **(CONDITIONAL)**

   This step is REQUIRED if access control is enforced by the MRTD chip and PACE has not been used.

   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.

After successful authentication, subsequent communication SHALL be protected by Secure Messaging using the session keys KS_{ENC} and KS_{MAC}.

The inspection system then continues with the inspection as described in Doc 9303 [9], [10], e.g. Passive Authentication MUST be performed. In addition the inspection system MUST verify the authenticity of the content of the file CardAccess (see above) using DG14.

2.2 Authentication of the Chip

The following mechanisms to verify the authenticity of the chip are available.

1. *Active Authentication*, as defined in Doc 9303 [9], [10]. Support of Active Authentication is indicated by the presence of DG15. If available, the terminal MAY read and verify DG15 and perform Active Authentication
2. *Chip Authentication*, as defined in section 4 of this Report. Support of Chip Authentication is indicated by the presence of corresponding `SecurityInfos` in DG14. If available, the terminal MAY read and verify DG14 and perform Chip Authentication.
3. PACE with *Chip Authentication Mapping* (PACE-CAM) as defined in section 3. Support is indicated by the presence of a corresponding `PACEInfo` structure in CardAccess. If PACE-CAM was performed successfully in the Open ePassport step of the inspection procedure, the terminal MAY perform the following to authenticate the chip:

- read and verify CardSecurity
- use the Public Key from CardSecurity together with the Mapping Data and the Chip Authentication Data received as part of PACE-CAM to authenticate the chip (section 3.4.4.2).

3 Password Authenticated Connection Establishment (PACE)

This section describes a supplemental access control mechanism based on Password Authenticated Connection Establishment (PACE) as described in [5].

The PACE Protocol is a password authenticated Diffie-Hellman key agreement protocol that provides secure communication and password-based authentication of the MRTD chip and the inspection system (i.e. MRTD chip and inspection system share the same password π).

PACE establishes Secure Messaging between an MRTD chip and an inspection system based on weak (short) passwords. The security context is established in the master file.

The protocol enables the MRTD chip to verify that the inspection system is authorized to access stored data and has the following features:

- Strong session keys are provided independent of the strength of the password.
- The entropy of the password(s) used to authenticate the inspection system can be very low (e.g. 6 digits are sufficient in general).

PACE uses keys K_π derived from passwords. For globally interoperable machine readable travel documents the following two passwords and corresponding keys are available as follows[1]:

MRZ: The key K_π = $\mathbf{KDF}_\pi$(MRZ) is REQUIRED. It is derived from the printed Machine Readable Zone (MRZ) similar to Basic Access Control, i.e. the key is derived from the Document Number, the Date of Birth and the Date of Expiry.

CAN: The key K_π= $\mathbf{KDF}_\pi$(CAN) is OPTIONAL. It is derived from the Card Access Number (CAN). The CAN is a number printed on the *front side* of the datapage and MUST be chosen randomly or pseudo-randomly (e.g. using a cryptographicly strong PRF).

***Note:** In contrast to the MRZ (Document Number, Date of Birth, Data of Expiry) the CAN has the advantage that it can easily be typed in manually.*

PACE supports different Mappings as part of the protocol execution:

- *Generic Mapping* based on a Diffie-Hellman Key Agreement;
- *Integrated Mapping* based on a direct mapping of a field element to the cryptographic group;
- *Chip Authentication Mapping* extends the Generic Mapping and integrates Chip Authentication into the PACE protocol.

If the chip supports Chip Authentication Mapping, at least one of Generic Mapping or Integrated Mapping and Chip Authentication MUST also be supported by the chip. This implies that for inspection systems supporting PACE, only support for Generic Mapping and Integrated Mapping is REQUIRED. Support for Chip Authentication Mapping is OPTIONAL.

[1] States MAY implement additional passwords for national purposes, e.g. a secret Personal Identification Number (PIN) and/or a PIN Unblock Key (PUK).

3.1 Protocol Setup

The inspection system reads the parameters for PACE supported by the MRTD chip from the file CardAccess (cf. Section 5.4) and selects the parameters to be used.

The following commands SHALL be used:

- Read Binary as specified in [9], [10].
- MSE:Set AT as specified in Section 3.5.1.

3.2 Protocol Specification

The following steps SHALL be performed by the inspection system and the MRTD chip using a chain of General Authenticate commands as specified in Section 3.5.2. A simplified version of the protocol is also shown in Figure 1.

1. The MRTD chip randomly and uniformly chooses a nonce s, encrypts the nonce to $z = \mathbf{E}(K_\pi, s)$, where $K_\pi = \mathbf{KDF}_\pi(\pi)$ is derived from the shared password π, and sends the ciphertext z to the inspection system.
2. The inspection system recovers the plaintext $s = \mathbf{D}(K_\pi, z)$ with the help of the shared password π.
3. Both the MRTD chip and the inspection system perform the following steps:

MRTD Chip (PICC)		***Inspection System (PCD)***
static domain parameters D_{PICC}		
choose random nonce $s \in_R Dom(E)$		
$z = \mathbf{E}(K_\pi, s)$	$\underline{z}\rangle$	$s = \mathbf{D}(K_\pi, z)$
additional data required for $\mathbf{Map}()$	$\langle - \rangle$	additional data required for $\mathbf{Map}()$
$D_{Map} = \mathbf{Map}(D_{PICC}, s, [t])$		$D_{Map} = \mathbf{Map}(D_{PICC}, s, [t])$
choose random ephemeral key pair $(SK_{DH,PICC}, PK_{DH,PICC}, D_{Map})$		choose random ephemeral key pair $(SK_{DH,PCD}, PK_{DH,PCD}, D_{Map})$
check that $PK_{DH,PCD} \neq PK_{DH,PICC}$	$\langle \frac{PK_{DH,PCD}}{PK_{DH,PICC}} \rangle$	check that $PK_{DH,PICC} \neq PK_{DH,PCD}$
$K = \mathbf{KA}(SK_{DH,PICC}, PK_{DH,PCD}, D_{Map})$		$K = \mathbf{KA}(SK_{DH,PCD}, PK_{DH,PICC}, D_{Map})$
$T_{PICC} = \mathbf{MAC}(KS_{MAC}, PK_{DH,PCD})$ [compute CA_{PICC} and encrypt as $A_{PICC} = \mathbf{E}(KS_{ENC}, CA_{PICC})$]	$\langle \frac{T_{PCD}}{T_{PICC}, [A_{PICC}]} \rangle$	$T_{PCD} = \mathbf{MAC}(KS_{MAC}, PK_{DH,PICC})$
verify T_{PCD}		verify T_{PICC} [decrypt A_{PICC} and authenticate chip]

Figure 1: PACE

a) They exchange additional data required for the mapping of the nonce (cf. Section 3.5.3.2):
   - For the Generic Mapping and the Chip Authentication Mapping the MRTD chip and the inspection system exchange ephemeral public keys.
   - For the Integrated Mapping the inspection system sends an additional nonce to the MRTD chip.

b) They compute the ephemeral domain parameters $D_{Map} = \mathbf{Map}(D_{PICC}, s, [t])$ as described in Section 3.4.2.

c) They perform an anonymous Diffie-Hellman key agreement (cf. Section 5.6) based on the ephemeral domain parameters and generate the shared secret

$$K = \mathbf{KA}(SK_{DH,PICC}, PK_{DH,PCD}, D_{Map}) = \mathbf{KA}(SK_{DH,PCD}, PK_{DH,PICC}, D_{Map}).$$

   During Diffie-Hellman key agreement, each party SHOULD check that the two public keys $PK_{DH,PICC}$ and $PK_{DH,PCD}$ differ.

d) They derive session keys $KS_{MAC} = \mathbf{KDF_{MAC}}(K)$ and $KS_{Enc} = \mathbf{KDF_{Enc}}(K)$ as described in Section 5.8.

e) They exchange and verify the authentication token $T_{PCD} = \mathbf{MAC}(KS_{MAC}, PK_{DH,PICC})$ and $T_{PICC} = \mathbf{MAC}(KS_{MAC}, PK_{DH,PCD})$ as described in Section 3.4.3.

4. Conditionally, the MRTD chip computes Chip Authentication Data CA_{PICC}, encrypts them $A_{PICC} = \mathbf{E}(KS_{ENC}, CA_{PICC})$ and sends them to the terminal (cf. Section 3.4.4.1). The terminal decrypts A_{PICC} and verifies the authenticity of the chip using the recovered Chip Authentication Data CA_{PICC} (cf. Section 3.4.4.2).

3.3 Security Status

If PACE was successfully performed then the MRTD chip has verified the used password. Secure Messaging is started using the derived session keys KS_{MAC} and KS_{Enc}.

3.4 Cryptographic Specifications

3.4.1 Algorithms

Particular algorithms are selected by the issuer of the MRTD. The inspection system MUST support all combinations described in the following. The MRTD chip MAY support more than one combination of algorithms.

Note: *Some algorithms are not available for the Chip Authentication Mapping: For security reasons, the use of 3DES is not longer recommended. DH-variants are not available to reduce the number of variants to be implemented by Terminals.*

3.4.1.1 DH

For PACE with DH the respective algorithms and formats from Table 6 and Table 1 MUST be used.

OID	*Mapping*	*Sym. Cipher*	*Keylen k*	*Secure Messaging*	*Auth. Token*
id-PACE-DH-GM-3DES-CBC-CBC	Generic	3DES	112	CBC / CBC	CBC
id-PACE-DH-GM-AES-CBC-CMAC-128		AES	128	CBC / CMAC	CMAC
id-PACE-DH-GM-AES-CBC-CMAC-192		AES	192	CBC / CMAC	CMAC
id-PACE-DH-GM-AES-CBC-CMAC-256		AES	256	CBC / CMAC	CMAC
id-PACE-DH-IM-3DES-CBC-CBC	Integrated	3DES	112	CBC / CBC	CBC
id-PACE-DH-IM-AES-CBC-CMAC-128		AES	128	CBC / CMAC	CMAC
id-PACE-DH-IM-AES-CBC-CMAC-192		AES	192	CBC / CMAC	CMAC
id-PACE-DH-IM-AES-CBC-CMAC-256		AES	256	CBC / CMAC	CMAC

Table 1: Object Identifiers for PACE with DH

3.4.1.2 ECDH

For PACE with ECDH the respective algorithms and formats from Table 6 and Table 2 MUST be used. Only prime curves with uncompressed points SHALL be used.

OID	*Mapping*	*Sym. Cipher*	*Keylen k*	*Secure Messaging*	*Auth. Token*
id-PACE-ECDH-GM-3DES-CBC-CBC	Generic	3DES	112	CBC / CBC	CBC
id-PACE-ECDH-GM-AES-CBC-CMAC-128		AES	128	CBC / CMAC	CMAC
id-PACE-ECDH-GM-AES-CBC-CMAC-192		AES	192	CBC / CMAC	CMAC
id-PACE-ECDH-GM-AES-CBC-CMAC-256		AES	256	CBC / CMAC	CMAC
id-PACE-ECDH-IM-3DES-CBC-CBC	Integrated	3DES	112	CBC / CBC	CBC
id-PACE-ECDH-IM-AES-CBC-CMAC-128		AES	128	CBC / CMAC	CMAC
id-PACE-ECDH-IM-AES-CBC-CMAC-192		AES	192	CBC / CMAC	CMAC
id-PACE-ECDH-IM-AES-CBC-CMAC-256		AES	256	CBC / CMAC	CMAC
id-PACE-ECDH-CAM-AES-CBC-CMAC-128	Chip Authenti-cation	AES	128	CBC / CMAC	CMAC
id-PACE-ECDH-CAM-AES-CBC-CMAC-192		AES	192	CBC / CMAC	CMAC
id-PACE-ECDH-CAM-AES-CBC-CMAC-256		AES	256	CBC / CMAC	CMAC

Table 2: Object Identifiers for PACE with ECDH

3.4.2 Encrypting and Mapping Nonces

The MRTD chip SHALL randomly and uniformly select the nonce $s \in_R \{0 \ldots 2^l - 1\}$ as a binary bit string of length l, where l is a multiple of the block size in bits of the respective block cipher $\mathbf{E}()$ chosen by the MRTD chip.

- The nonce s SHALL be encrypted in CBC mode according to ISO/IEC 10116 [12] using the key $K_\pi = \mathbf{KDF}_\pi(\pi)$ derived from the password π and IV set to the all-0 string.
- The nonce s SHALL be converted to a group generator using an algorithm-specific mapping function **Map**, which maps a static generator G (ECDH case) / g (DH case) to an ephemeral generator $\tilde{G}$ / $\tilde{g}$, respectively.
- For the Integrated Mapping the additional nonce $t \in_R \{0 \ldots 2^k - 1\}$ SHALL be sent in clear. In this case k is the key size in bits of the respective block cipher $\mathbf{E}()$ and l is the smallest multiple of the block size of $\mathbf{E}()$ such that $l \geq k$.

To map the nonce s or the nonces s, t into the cryptographic group one of the following mappings SHALL be used:

- *Generic Mapping* (section 3.4.2.1);
- *Integrated Mapping* (section 3.4.2.2);
- *Chip Authentication Mapping* (section 3.4.2.3).

3.4.2.1 Generic Mapping

3.4.2.1.1 ECDH

The function $\mathbf{Map}: G \mapsto G_{Map}$ is defined as $G_{Map} = s \cdot G + H$, where $H \in \langle G \rangle$ is chosen such that $\log_G H$ is unknown. The point H SHALL be calculated by an anonymous Diffie-Hellman Key Agreement [6] as $H = \mathbf{KA}(SK_{Map,PICC}, PK_{Map,PCD}, D_{PICC}) = \mathbf{KA}(SK_{Map,PCD}, PK_{Map,PICC}, D_{PICC})$.

Note: *The key agreement algorithm ECKA as specified in [6] prevents small subgroup attacks by using compatible cofactor multiplication.*

3.4.2.1.2 DH

The function $\mathbf{Map}: g \mapsto g_{Map}$ is defined as $g_{Map} = g^s \cdot h$, where $h \in \langle g \rangle$ is chosen such that $\log_g h$ is unknown. The group element h SHALL be calculated by an anonymous Diffie-Hellman Key Agreement as $h = \mathbf{KA}(SK_{Map,PICC}, PK_{Map,PCD}, D_{PICC}) = \mathbf{KA}(SK_{Map,PCD}, PK_{Map,PICC}, D_{PICC})$.

Note: *The public key validation method described in RFC 2631 [22] MUST be used to prevent small subgroup attacks.*

3.4.2.2 Integrated Mapping

3.4.2.2.1 ECDH

The function $\mathbf{Map}: G \mapsto G_{Map}$ is defined as $G_{Map} = f_G(\mathbf{R}_p(s,t))$, where $\mathbf{R}_p()$ is a pseudo-random function that maps octet strings to elements of $GF(p)$ and $f_G()$ is a function that maps elements of

$GF(p)$ to $\langle G \rangle$. The random nonce t SHALL be chosen randomly by the inspection system and sent to the MRTD chip. The pseudo-random function $\mathbf{R}_p()$ is described in Section 3.4.2.2.3. The function $f_G()$ is defined in [4] and [25].

3.4.2.2.2 DH

The function $\mathbf{Map}: g \mapsto g_{Map}$ is defined as $g_{Map} = f_g(\mathbf{R}_p(s,t))$, where $\mathbf{R}_p()$ is a pseudo-random function that maps octet strings to elements of $GF(p)$ and $f_g()$ is a function that maps elements of $GF(p)$ to $\langle g \rangle$. The random nonce t SHALL be chosen randomly by the inspection system and sent to the MRTD chip. The pseudo-random function $\mathbf{R}_p()$ is described in Section 3.4.2.2.3. The function $f_g()$ is defined as $f_g(x) = x^a \bmod p$, and $a = (p-1)/q$ is the cofactor. Implementations MUST check that $\tilde{g} \neq 1$.

3.4.2.2.3 Pseudorandom Number Mapping

The function $\mathbf{R}_p(s,t)$ is a function that maps octet strings s (of bit length l) and t (of bit length k) to an element $\text{int}(x_1 \| x_2 \| ... \| x_n) \bmod p$ of $GF(p)$. The function $\mathbf{R}(s,t)$ is specified in Figure 2. The construction is based on the respective block cipher $\mathbf{E}()$ in CBC mode according to ISO/IEC 10116 [12] with $IV = 0$, where k is the key size (in bits) of $\mathbf{E}()$. Where required, the output k_i MUST be truncated to key size k. The value n SHALL be selected as smallest number, such that $n \cdot l \geq \log_2 p + 64$.

Note: *The truncation is only necessary for AES-192: Use octets 1 to 24 of k_i; additional octets are not used.*

The constants c_0 and c_1 are defined as follows:

- For 3DES and AES-128 ($l = 128$):
  - c_0=0xa668892a7c41e3ca739f40b057d85904
  - c_1=0xa4e136ac725f738b01c1f60217c188ad
- For AES-192 and AES-256 ($l = 256$):
  - c_0=0xd463d65234124ef7897054986dca0a174e28df758cbaa03f240616414d5a1676

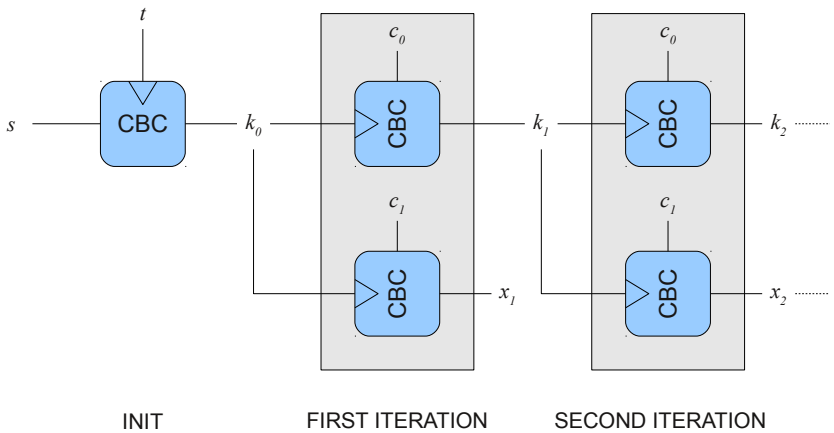

Figure 2: The function R(s,t) using the block cipher E() in CBC mode

3.4.2.3 Chip Authentication Mapping

The mapping phase of the PACE-CAM is identical to the mapping phase of PACE-GM (cf. Section 3.4.2.1).

3.4.3 Authentication token

The authentication token SHALL be computed over a public key data object (cf. Section 5.9) containing the object identifier as indicated in MSE:Set AT (cf. Section 3.5.1), and the received ephemeral public key (i.e. excluding the domain parameters, cf. Section 5.9.4) using an authentication code and the key KS_{MAC} derived from the key agreement.

Note: *Padding is performed internally by the message authentication code, i.e. no application specific padding is performed.*

3.4.3.1 3DES

3DES [20] SHALL be used in Retail-mode according to ISO/IEC 9797-1 [14] MAC algorithm 3 / padding method 2 with block cipher DES and $IV = 0$.

3.4.3.2 AES

AES [19] SHALL be used in CMAC-mode [21] with a MAC length of 8 bytes.

3.4.4 Encrypted Chip Authentication Data

The MRTD chip MUST provide static key pair(s) SK_{PICC}, PK_{PICC} as described in section 4.

Encrypted Chip Authentication Data is REQUIRED for PACE with Chip Authentication Mapping.

3.4.4.1 Generation by the MRTD chip

The Chip Authentication Data SHALL be computed as $CA_{PICC} = (SK_{PICC})^{-1} \cdot SK_{Map,PICC} \mod p$,

where SK_{PICC} is the static private key of the chip, $SK_{Map,PICC}$ is the ephemeral private key used by the chip in the mapping phase of PACE (cf. Section 3.4.2.3) and p is the order of the used cryptographic group. The Chip Authentication Data SHALL be encrypted using the key KS_{Enc} derived from the key agreement as $A_{PICC} = \mathbf{ENC}(KS_{Enc}, CA_{PICC})$ to yield the Encrypted Chip Authentication Data.

Note: $(SK_{PICC})^{-1}$ *can be precomputed during personalization of the MRTD chip and securely stored in the chip, avoiding the modular inversion during run-time.*

3.4.4.2 Verification by the terminal

The terminal SHALL decrypt A_{PICC} to recover CA_{PICC} and verify $PK_{Map,PICC} = \mathbf{KA}(CA_{PICC}, PK_{PICC}, D_{PICC})$, where PK_{PICC} is the static public key of the MRTD chip.

Note: *Passive Authentication MUST be performed in combination with the Chip Authentication Mapping. Only after a successful validation of the respective Security Object the MRTD chip may be considered genuine.*

3.4.4.3 Padding

The data to be encrypted SHALL be padded according to ISO/IEC 9797-1 [14] "Padding Method 2".

3.4.4.4 AES

AES [19] SHALL be used in CBC-mode according to ISO/IEC 10116 [12] with $IV = \mathbf{E}(KS_{Enc}, -1)$, where -1 is the bit string of length 128 with all bits set to 1.

3.5 Application Protocol Data Units

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

1. MSE:Set AT
2. General Authenticate

3.5.1 MSE:Set AT

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

Command			
CLA		Context specific	
INS	0x22	Manage Security Environment	
P1/P2	0xC1A4	Set Authentication Template for mutual authentication	
Data	0x80	*Cryptographic mechanism reference* Object Identifier of the protocol to select (value only, Tag 0x06 is omitted).	REQUIRED
	0x83	*Reference of a public key / secret key* The password to be used is indicated as follows: 0x01: MRZ 0x02: CAN	REQUIRED
	0x84	*Reference of a private key / Reference for computing a session key* This data object is REQUIRED to indicate the identifier of the domain parameters to be used if the domain parameters are ambiguous, i.e. more than one set of domain parameters is available for PACE.	CONDITIONAL

Response		
Data	–	Absent
Status Bytes	0x9000	*Normal operation* The protocol has been selected and initialized.
	0x6A80	*Incorrect parameters in the command data field* Algorithm not supported or initialization failed.

	0x6A88	*Referenced data not found* The referenced data (i.e. password or domain parameter) is not available.
	other	*Operating system dependent error* The initialization of the protocol failed.

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

3.5.2 General Authenticate

A chain of General Authenticate commands is used to perform the PACE protocol.

Command			
CLA		Context specific.	
INS	0x86	General Authenticate	
P1/P2	0x0000	Keys and protocol implicitly known	
Data	0x7C	*Dynamic Authentication Data* Protocol specific data objects	REQUIRED
Response			
Data	0x7C	*Dynamic Authentication Data* Protocol specific data objects as described in Section 3.5.3.	REQUIRED
Status Bytes	0x9000	*Normal operation* The protocol (step) was successful.	
	0x6300	*Authentication failed* The protocol (step) failed.	
	0x6A80	*Incorrect parameters in data field* Provided data is invalid.	
	other	*Operating system dependent error* The protocol (step) failed.	

3.5.3 Exchanged Data

The protocol specific data objects SHALL be exchanged in a chain of General Authenticate commands as shown below:

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

[2]This implies an empty Dynamic Authentication Data Object.

Encrypted Chip Authentication Data (cf. Section 3.4.4) MUST be present if Chip Authentication Mapping is used and MUST NOT be present otherwise.

3.5.3.1 Encrypted Nonce

The encrypted nonce (cf. Section 3.4.2) SHALL be encoded as octet string.

3.5.3.2 Mapping Data

The exchanged data is specific to the used mapping:

3.5.3.2.1 Generic Mapping

The ephemeral public keys (cf. Section 3.4.2 and Section 5.9.4) SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

3.5.3.2.2 Integrated Mapping

The nonce *t* SHALL be encoded as octet string.

Note: *The context specific data object 0x82 SHALL be empty.*

3.5.3.2.3 Chip Authentication Mapping

The encoding of the mapping data is identical to the Generic Mapping (cf. Section 3.5.3.2.1)

3.5.3.3 Public Keys

The public keys SHALL be encoded as described in Section 5.9.4.

3.5.3.4 Authentication Token

The authentication token (cf. Section 3.4.3) SHALL be encoded as octet string.

3.5.3.5 Encrypted Chip Authentication Data

The Chip Authentication Data SHALL be encoded as octet string using the function FE2OS() specified in [6] before encryption. Note that FE2OS() requires the encoding with the same number of octets as the prime order of the group, i.e. possibly including leading 0x00's. The Encrypted Chip Authentication Data SHALL be encoded as octet string.

4 Chip Authentication

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

The static Chip Authentication Key Pair(s) MUST be stored on the MRTD chip.

- The private key SHALL be stored securely in the MRTD chip's memory.
- The public key SHALL be provided as `SubjectPublicKeyInfo` in the `ChipAuthenticationPublicKeyInfo` structure (see section 5.3.4).

The protocol provides implicit authentication of both the MRTD chip itself and the stored data by performing Secure Messaging using the new session keys.

As the ePassport Application is selected as result of the Access eMRTD-Procedure, Chip Authentication is performed in the ePassport Application.

4.1 Protocol Specification

The following steps are performed by the terminal and the MRTD chip, a simplified version is also shown in Figure 3.

1. The MRTD chip sends its static Diffie-Hellman public key PK_{PICC}, and the domain parameters D_{PICC} to the terminal.
2. The terminal generates an ephemeral Diffie-Hellman key pair $(SK_{DH,PCD}, PK_{DH,PCD}, D_{PICC})$, and sends the ephemeral public key $PK_{DH,PCD}$ to the MRTD chip.
3. Both the MRTD chip and the terminal compute the following:
   1. The shared secret $K = \mathbf{KA}(SK_{PICC}, PK_{DH,PCD}, D_{PICC}) = \mathbf{KA}(SK_{DH,PCD}, PK_{PICC}, D_{PICC})$
   2. The session keys $KS_{MAC} = \mathbf{KDF_{MAC}}(K)$ and $KS_{Enc} = \mathbf{KDF_{Enc}}(K)$ derived from K for Secure Messaging.

To verify the authenticity of the PK_{PICC} the terminal SHALL perform Passive Authentication.

4.2 Security Status

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

Note: *Passive Authentication MUST be performed in combination with Chip Authentication. Only after a successful validation of the respective Security Object the MRTD chip may be considered genuine.*

MRTD Chip (PICC)		***Terminal (PCD)***
static key pair $(SK_{PICC}, PK_{PICC}, D_{PICC})$		
	$\frac{PK_{PICC}}{D_{PICC}}\rangle$	
		choose random ephemeral key pair $(SK_{DH,PCD}, PK_{DH,PCD}, D_{PICC})$
	$\langle \frac{PK_{DH,PCD}}{}$	
$K = \mathbf{KA}(SK_{PICC}, PK_{DH,PCD}, D_{PICC})$		$K = \mathbf{KA}(SK_{DH,PCD}, PK_{PICC}, D_{PICC})$

Figure 3: Chip Authentication

4.3 Cryptographic Specifications

Particular algorithms are selected by the issuer of the MRTD. The inspection system MUST support all combinations described in the following. The MRTD chip MAY support more than one combination of algorithms.

4.3.1 Chip Authentication with DH

For Chip Authentication with DH the respective algorithms and formats from Table 6 and Table 3 MUST be used. For Public Keys PKCS#3 [24] MUST be used instead of X9.42 [1].

OID	*Sym. Cipher*	*Key Length*	*Secure Messaging*
id-CA-DH-3DES-CBC-CBC	3DES	112	CBC / CBC
id-CA-DH-AES-CBC-CMAC-128	AES	128	CBC / CMAC
id-CA-DH-AES-CBC-CMAC-192	AES	192	CBC / CMAC
id-CA-DH-AES-CBC-CMAC-256	AES	256	CBC / CMAC

Table 3: Object Identifiers for Chip Authentication with DH

4.3.2 Chip Authentication with ECDH

For Chip Authentication with ECDH the respective algorithms and formats from Table 6 and Table 4 MUST be used.

OID	*Sym. Cipher*	*Key Length*	*Secure Messaging*
id-CA-ECDH-3DES-CBC-CBC	3DES	112	CBC / CBC
id-CA-ECDH-AES-CBC-CMAC-128	AES	128	CBC / CMAC
id-CA-ECDH-AES-CBC-CMAC-192	AES	192	CBC / CMAC
id-CA-ECDH-AES-CBC-CMAC-256	AES	256	CBC / CMAC

Table 4: Object Identifiers for Chip Authentication with ECDH

4.4 Applications Protocol Data Units

Depending on the symmetric algorithm to be used two implementations of Chip Authentication are available.

- The following command SHALL be used to implement Chip Authentication with 3DES Secure Messaging:
    1. MSE:Set KAT
- The following sequence of commands SHALL be used to implement Chip Authentication with AES Secure Messaging and MAY be used to implement Chip Authentication with 3DES Secure Messaging:
    1. MSE:Set AT
    2. General Authenticate

4.4.1 Implementation using MSE:Set KAT

Note: *MSE:Set KAT may only be used for* `id-CA-DH-3DES-CBC-CBC` *and* `id-CA-ECDH-3DES-CBC-CBC`*, i.e. Secure Messaging is restricted to 3DES.*

Command			
CLA		Context specific	
INS	0x22	Manage Security Environment	
P1/P2	0x41A6	Set Key Agreement Template for computation.	
Data	0x91	*Ephemeral Public Key* Ephemeral public key $PK_{DH,PCD}$ (cf. Section 5.9.4) 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. Section 4 and 5.2.2).	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.
	other	*Operating system dependent error* The previously established session keys remain valid.

4.4.2 Implementation using MSE:Set AT and General Authenticate

4.4.2.1 MSE:Set AT

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

Command			
CLA		Context specific	
INS	0x22	Manage Security Environment	
P1/P2	0x41A4	*Chip Authentication:* Set Authentication Template for internal authentication.	
Data	0x80	*Cryptographic mechanism reference* Object Identifier of the protocol to select (value only, Tag 0x06 is omitted).	REQUIRED
	0x84	*Reference of a private key* This data object is REQUIRED to indicate the identifier of the private key to be used if the private key is ambiguous, i.e. more than one private key is available for Chip Authentication.	CONDITIONAL
Response			
Data	–	Absent	
Status Bytes	0x9000	*Normal operation* The protocol has been selected and initialized.	
	0x6A80	*Incorrect parameters in the command data field* Algorithm not supported or initialization failed.	
	0x6A88	*Referenced data not found* The referenced data (i.e. private key) is not available.	
	other	*Operating system dependent error* The initialization of the protocol failed.	

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

4.4.2.2 General Authenticate

The command General Authenticate is used to perform the Chip Authentication.

Command				
CLA		Context specific		
INS	0x86	General Authenticate		
P1/P2	0x0000	Keys and protocol implicitly known.		
Data	0x7C	*Dynamic Authentication Data* Protocol specific data objects.		REQUIRED
		0x80	Ephemeral Public Key	

Response			
Data	0x7C	*Dynamic Authentication Data* Protocol specific data objects	REQUIRED
Status Bytes	0x9000	*Normal operation* The protocol (step) was successful.	
	0x6300	*Authentication failed* The protocol (step) failed.	
	0x6A80	*Incorrect parameters in data field* Provided data is invalid.	
	0x6A88	*Referenced data not found* The referenced data (i.e. private key) is not available.	
	other	*Operating system dependent error* The protocol (step) failed.	

Note: *The public keys for Chip Authentication supported by the chip are made available in the Security Object (see section 5.4). If more than one public key is supported, the terminal MUST select the corresponding private key of the chip to be used within* MSE:Set AT.

4.4.3 Ephemeral Public Key

The ephemeral public keys (cf. Section 5.9.4) SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

5 Technical specifications

This section describes the technical specifications required to implement the protocols.

5.1 Object Identifier

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

id-PACE OBJECT IDENTIFIER ::= {
  bsi-de protocols(2) smartcard(2) 4
}

id-PACE-DH-GM                    OBJECT IDENTIFIER ::= {id-PACE 1}
id-PACE-DH-GM-3DES-CBC-CBC       OBJECT IDENTIFIER ::= {id-PACE-DH-GM 1}
id-PACE-DH-GM-AES-CBC-CMAC-128   OBJECT IDENTIFIER ::= {id-PACE-DH-GM 2}
id-PACE-DH-GM-AES-CBC-CMAC-192   OBJECT IDENTIFIER ::= {id-PACE-DH-GM 3}
id-PACE-DH-GM-AES-CBC-CMAC-256   OBJECT IDENTIFIER ::= {id-PACE-DH-GM 4}

id-PACE-ECDH-GM                  OBJECT IDENTIFIER ::= {id-PACE 2}
id-PACE-ECDH-GM-3DES-CBC-CBC     OBJECT IDENTIFIER ::= {id-PACE-ECDH-GM 1}
id-PACE-ECDH-GM-AES-CBC-CMAC-128 OBJECT IDENTIFIER ::= {id-PACE-ECDH-GM 2}
id-PACE-ECDH-GM-AES-CBC-CMAC-192 OBJECT IDENTIFIER ::= {id-PACE-ECDH-GM 3}
id-PACE-ECDH-GM-AES-CBC-CMAC-256 OBJECT IDENTIFIER ::= {id-PACE-ECDH-GM 4}

id-PACE-DH-IM                    OBJECT IDENTIFIER ::= {id-PACE 3}
```

```
id-PACE-DH-IM-3DES-CBC-CBC        OBJECT IDENTIFIER ::= {id-PACE-DH-IM 1}
id-PACE-DH-IM-AES-CBC-CMAC-128    OBJECT IDENTIFIER ::= {id-PACE-DH-IM 2}
id-PACE-DH-IM-AES-CBC-CMAC-192    OBJECT IDENTIFIER ::= {id-PACE-DH-IM 3}
id-PACE-DH-IM-AES-CBC-CMAC-256    OBJECT IDENTIFIER ::= {id-PACE-DH-IM 4}

id-PACE-ECDH-IM                   OBJECT IDENTIFIER ::= {id-PACE 4}
id-PACE-ECDH-IM-3DES-CBC-CBC      OBJECT IDENTIFIER ::= {id-PACE-ECDH-IM 1}
id-PACE-ECDH-IM-AES-CBC-CMAC-128  OBJECT IDENTIFIER ::= {id-PACE-ECDH-IM 2}
id-PACE-ECDH-IM-AES-CBC-CMAC-192  OBJECT IDENTIFIER ::= {id-PACE-ECDH-IM 3}
id-PACE-ECDH-IM-AES-CBC-CMAC-256  OBJECT IDENTIFIER ::= {id-PACE-ECDH-IM 4}

id-PACE-ECDH-CAM                  OBJECT IDENTIFIER ::= {id-PACE 6}

id-PACE-ECDH-CAM-AES-CBC-CMAC-128 OBJECT IDENTIFIER ::= {id-PACE-ECDH-CAM 2}
id-PACE-ECDH-CAM-AES-CBC-CMAC-192 OBJECT IDENTIFIER ::= {id-PACE-ECDH-CAM 3}
id-PACE-ECDH-CAM-AES-CBC-CMAC-256 OBJECT IDENTIFIER ::= {id-PACE-ECDH-CAM 4}

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

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

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

id-CA-DH                    OBJECT IDENTIFIER ::= {id-CA 1}
id-CA-DH-3DES-CBC-CBC       OBJECT IDENTIFIER ::= {id-CA-DH 1}
id-CA-DH-AES-CBC-CMAC-128   OBJECT IDENTIFIER ::= {id-CA-DH 2}
id-CA-DH-AES-CBC-CMAC-192   OBJECT IDENTIFIER ::= {id-CA-DH 3}
id-CA-DH-AES-CBC-CMAC-256   OBJECT IDENTIFIER ::= {id-CA-DH 4}

id-CA-ECDH                  OBJECT IDENTIFIER ::= {id-CA 2}
id-CA-ECDH-3DES-CBC-CBC     OBJECT IDENTIFIER ::= {id-CA-ECDH 1}
id-CA-ECDH-AES-CBC-CMAC-128 OBJECT IDENTIFIER ::= {id-CA-ECDH 2}
id-CA-ECDH-AES-CBC-CMAC-192 OBJECT IDENTIFIER ::= {id-CA-ECDH 3}
id-CA-ECDH-AES-CBC-CMAC-256 OBJECT IDENTIFIER ::= {id-CA-ECDH 4}
```

5.2 Information on Supported Protocols

The ASN.1 data structure `SecurityInfos` SHALL be provided by the MRTD chip 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.

5.2.1 Security Infos for PACE

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

- At least one `PACEInfo` using a standardized domain parameter MUST be present.
- For each supported set of explicit domain parameters a `PACEDomainParameterInfo` MUST be present.

5.2.2 Security Infos for Chip Authentication

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

- At least one `ChipAuthenticationInfo` and the corresponding `ChipAuthenticationPublicKeyInfo` using explicit domain parameters MUST be present.

5.2.3 Security Infos for other Protocols

`SecurityInfos` may contain additional entries indicating support for other protocols. The inspection system may discard any unknown entry.

5.3 ASN.1 Structures

The data structures `SubjectPublicKeyInfo` and `AlgorithmIdentifier` are defined as follows; more details can be found in [7]:

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

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

Details on the `parameters` can be found in [1] and [6].

5.3.1 PACEInfo

This data structure provides detailed information on an implementation of PACE.

- The object identifier `protocol` SHALL identify the algorithms to be used (i.e. key agreement, mapping, symmetric cipher and MAC).
- The integer `version` SHALL identify the version of the protocol. Only version 2 is supported by this specification.

- The integer `parameterId` is used to indicate the domain parameter identifier. It MUST be used if the MRTD chip uses standardized domain parameters (cf. 7), provides multiple explicit domain parameters for PACE or protocol is one of the *-`CAM`-* OIDs. In case of PACE with Chip Authentication Mapping, the `parameterID` also denotes the ID of the Chip Authentication key used, i.e. the chip MUST provide a `ChipAuthenticationPublicKeyInfo` with `keyID` equal to `parameterID` from this data structure.

```
PACEInfo ::= SEQUENCE {
  protocol    OBJECT IDENTIFIER(
              id-PACE-DH-GM-3DES-CBC-CBC |
              id-PACE-DH-GM-AES-CBC-CMAC-128 |
              id-PACE-DH-GM-AES-CBC-CMAC-192 |
              id-PACE-DH-GM-AES-CBC-CMAC-256 |
              id-PACE-ECDH-GM-3DES-CBC-CBC |
              id-PACE-ECDH-GM-AES-CBC-CMAC-128 |
              id-PACE-ECDH-GM-AES-CBC-CMAC-192 |
              id-PACE-ECDH-GM-AES-CBC-CMAC-256 |
              id-PACE-DH-IM-3DES-CBC-CBC |
              id-PACE-DH-IM-AES-CBC-CMAC-128 |
              id-PACE-DH-IM-AES-CBC-CMAC-192 |
              id-PACE-DH-IM-AES-CBC-CMAC-256 |
              id-PACE-ECDH-IM-3DES-CBC-CBC |
              id-PACE-ECDH-IM-AES-CBC-CMAC-128 |
              id-PACE-ECDH-IM-AES-CBC-CMAC-192 |
              id-PACE-ECDH-IM-AES-CBC-CMAC-256 |
              id-PACE-ECDH-CAM-AES-CBC-CMAC-128 |
              id-PACE-ECDH-CAM-AES-CBC-CMAC-192 |
              id-PACE-ECDH-CAM-AES-CBC-CMAC-256),
  version     INTEGER, -- MUST be 2
  parameterId INTEGER OPTIONAL
}
```

5.3.2 PACEDomainParameterInfo

This data structure is REQUIRED if the MRTD chip provides explicit domain parameters for PACE of the MRTD chip and MUST be omitted otherwise.

- The object identifier `protocol` SHALL identify the type of the domain parameters (i.e. DH or ECDH).
- The sequence `domainParameter` SHALL contain the domain parameters.
- The integer `parameterId` MAY be used to indicate the local domain parameter identifier. It MUST be used if the MRTD chip provides multiple explicit domain parameters for PACE.

```
PACEDomainParameterInfo ::= SEQUENCE {
  protocol        OBJECT IDENTIFIER(
                  id-PACE-DH-GM |
                  id-PACE-ECDH-GM |
                  id-PACE-DH-IM |
                  id-PACE-ECDH-IM
```

```
                  id-PACE-DH-CAM |
                  id-PACE-ECDH-CAM),
  domainParameter AlgorithmIdentifier,
  parameterId     INTEGER OPTIONAL
}
```

Note: *The MRTD chip MAY support more than one set of explicit domain parameters (i.e. the chip may support different algorithms and/or key lengths). In this case the identifier MUST be disclosed in the corresponding* `PACEDomainParameterInfo`*.*

Domain parameters contained in `PACEDomainParameterInfo` are unprotected and may be insecure. Using insecure domain parameters for PACE will leak the used password. MRTD chips MUST support at least one set of standardized domain parameters as specified in Table 7. Inspection systems MUST NOT use explicit domain parameters provided by the MRTD chip unless those domain parameters are explicitly known to be secure by the inspection systems.

Ephemeral public keys MUST be exchanged as plain public key values. More information on the encoding can be found in Section 5.9.4.

5.3.3 ChipAuthenticationInfo

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

- The object identifier `protocol` SHALL identify the algorithms to be used (i.e. key agreement, symmetric cipher and MAC).
- The integer `version` SHALL identify the version of the protocol. Currently, only version 1 is supported by this specification.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the MRTD chip provides multiple public keys for Chip Authentication.

```
ChipAuthenticationInfo ::= SEQUENCE {
  protocol OBJECT IDENTIFIER(
           id-CA-DH-3DES-CBC-CBC |
           id-CA-DH-AES-CBC-CMAC-128 |
           id-CA-DH-AES-CBC-CMAC-192 |
           id-CA-DH-AES-CBC-CMAC-256 |
           id-CA-ECDH-3DES-CBC-CBC |
           id-CA-ECDH-AES-CBC-CMAC-128 |
           id-CA-ECDH-AES-CBC-CMAC-192 |
           id-CA-ECDH-AES-CBC-CMAC-256),
  version  INTEGER, -- MUST be 1
  keyId    INTEGER OPTIONAL
}
```

5.3.4 ChipAuthenticationPublicKeyInfo

This data structure provides a public key for Chip Authentication or PACE with Chip Authentication Mapping of the MRTD chip.

- The object identifier `protocol` SHALL identify the type of the public key (i.e. DH or ECDH).

- The sequence `chipAuthenticationPublicKey` SHALL contain the public key in encoded form.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the MRTD chip provides multiple public keys for Chip Authentication or if this key is used for PACE with Chip Authentication Mapping.

```
ChipAuthenticationPublicKeyInfo ::= SEQUENCE {
  protocol                    OBJECT IDENTIFIER(id-PK-DH | id-PK-ECDH),
  chipAuthenticationPublicKey SubjectPublicKeyInfo,
  keyId                       INTEGER OPTIONAL
}
```

Note: *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 `ChipAuthenticationInfo` and `ChipAuthenticationPublicKeyInfo`*

5.4 Storage on the Chip

To indicate support for the protocols and supported parameters, the MRTD chip SHALL provide `SecurityInfos` in transparent elementary files (cf. Table 5):

- The file *CardAccess* contained in the master file is REQUIRED if PACE is supported by the MRTD chip and SHALL contain the following `SecurityInfos` that are required for PACE:
  - `PACEInfo`
  - `PACEDomainParameterInfo`.
- The file *CardSecurity* contained in the master file is REQUIRED if PACE with Chip Authentication Mapping is supported by the MRTD chip and SHALL contain the following `SecurityInfos`:
  - `ChipAuthenticationPublicKeyInfo` as required for PACE-CAM
  - The `SecurityInfos` contained in CardAccess.

File Name	***MF/EF.CardAccess***	***MF/EF.CardSecurity***	***DF.ICAO/DG14***
File ID	0x011C	0x011D	0x010E
Short File ID	0x1C	0x1D	0x0E
Read Access	ALWAYS	PACE	BAC or PACE
Write Access	NEVER	NEVER	NEVER
Size	variable	variable	variable
Content	DER encoded `SecurityInfos`	DER encoded `SignedData`	DER encoded `SecurityInfos`

Table 5: Elementary Files CardAccess, CardSecurity, DG14

- The file *DG14* contained in the ePassport Application (cf. [9], [10]) is REQUIRED if Chip Authentication or PACE-GM/-IM is supported by the MRTD chip and SHALL contain the following `SecurityInfos`:
  - `ChipAuthenticationInfo` as required for Chip Authentication
  - `ChipAuthenticationPublicKeyInfo` as required for Chip Authentication
  - The `SecurityInfos` contained in CardAccess.

The files MAY contain additional `SecurityInfos` out of scope of this specification.

Note: *While the authenticity of `SecurityInfos` stored in DG14 and CardSecurity is protected by Passive Authentication, the file CardAccess is unprotected.*

5.5 APDUs

5.5.1 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 [13].

5.5.1.1 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/INFO as specified in [13].

5.5.1.2 Terminals

For terminals support of extended length is REQUIRED. A terminal SHOULD examine whether or not support for extended length is indicated in the MRTD chip's ATR/ATS or in EF.ATR/INFO before using this option. The terminal 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/INFO.

- MSE:Set KAT
- General Authenticate

5.5.2 Command Chaining

Command chaining MUST be used for the General Authenticate command to link the sequence of commands to the execution of the protocol. Command chaining MUST NOT be used for other purposes unless clearly indicated by the chip. For details on command chaining see [13].

5.6 Key Agreement Algorithms

Algorithm / Format	*DH*	*ECDH*
Key Agreement Algorithm	PKCS#3 [23]	ECKA [6]
X.509 Public Key Format	X9.42 [1]	ECC [6]
TLV Public Key Format	TLV, cf. Section 5.9.2	TLV, cf. Section 5.9.3
Ephemeral Public Key Validation	RFC 2631 [22]	ECC [6]

Table 6: Algorithms and Formats for Key Agreement

This specification supports Diffie-Hellman and Elliptic Curve Diffie-Hellman key agreement as summarized in Table 6.

5.7 Domain Parameters

With the exception of domain parameters contained in `PACEInfo`, all domain parameters SHALL be provided as `AlgorithmIdentifier` (cf. Section 5.3).

Within `PACEInfo`, the ID of standardized domain parameters described in Table 7 SHALL be referenced directly. Explicit domain parameters provided by `PACEDomainParameterInfo` MUST NOT use those IDs reserved for standardized domain parameters.

5.7.1 Standardized Domain Parameters

The standardized domain parameters described in Table 7 SHOULD be used. Explicit domain parameters MUST NOT use those IDs reserved for standardized domain parameters.

The following object identifier SHOULD be used to reference standardized domain parameters in an `AlgorithmIdentifier` (cf. Section 5.3):

```
standardizedDomainParameters OBJECT IDENTIFIER ::= {
  bsi-de algorithms(1) 2
}
```

Within an `AlgorithmIdentifier` this object identifier SHALL reference the ID of the standardized domain parameter as contained in Table 7 as `INTEGER`, contained as `parameters` in the `AlgorithmIdentifier`.

ID	*Name*	*Size*	*Type*	*Reference*
0	1024-bit MODP Group with 160-bit Prime Order Subgroup	1024/160	GFP	[15]
1	2048-bit MODP Group with 224-bit Prime Order Subgroup	2048/224	GFP	[15]
2	2048-bit MODP Group with 256-bit Prime Order Subgroup	2048/256	GFP	[15]
3 - 7	RFU			
8	NIST P-192 (secp192r1)	192	ECP	[17], [15]
9	BrainpoolP192r1	192	ECP	[16]
10	NIST P-224 (secp224r1)*	224	ECP	[17], [15]
11	BrainpoolP224r1	224	ECP	[16]
12	NIST P-256 (secp256r1)	256	ECP	[17], [15]
13	BrainpoolP256r1	256	ECP	[16]
14	BrainpoolP320r1	320	ECP	[16]
15	NIST P-384 (secp384r1)	384	ECP	[17], [15]
16	BrainpoolP384r1	384	ECP	[16]
17	BrainpoolP512r1	512	ECP	[16]
18	NIST P-521 (secp521r1)	521	ECP	[17], [15]
19-31	RFU			

* This curve cannot be used with the integrated mapping.

Table 7: Standardized Domain Parameters

5.7.2 Explicit Domain Parameters

The object identifier `dhpublicnumber` or `ecPublicKey` for DH or ECDH, respectively, SHALL be used to reference explicit domain parameters in an `AlgorithmIdentifier` (cf. Section 5.3):

```
dhpublicnumber OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1
}

ecPublicKey OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) ansi-x962(10045) keyType(2) 1
}
```

In the case of elliptic curves domain parameters MUST be described explicitly in the `ECParameters` structure, contained as `parameters` in the `AlgorithmIdentifier`, i.e. named curves and implicit domain parameters MUST NOT be used.

5.8 Key Derivation Function

Let $\mathbf{KDF}_{\mathbf{Enc}}(K)=\mathbf{KDF}(K,1)$, $\mathbf{KDF}_{\mathbf{MAC}}(K)=\mathbf{KDF}(K,2)$, be key derivation functions to derive encryption and authentication keys, respectively, from a shared secret K. Let $\mathbf{KDF}_{\pi}(\pi)=\mathbf{KDF}(f(\pi),3)$, be a key derivation function to derive encryption keys from a password π. The encoding of passwords, i.e. $K=f(\pi)$ is specified in Table 8.

Password	***Encoding***
MRZ	SHA-1(Serial Number || Date of Birth || Date of Expiry)
CAN	ISO/IEC8859 encoded character string

Table 8: Encoding of Passwords

The key derivation function $\mathbf{KDF}(K,c)$, is defined as follows:

Input: The following inputs are required:

- The shared secret value K **(REQUIRED)**
- A 32-bit, big-endian integer counter c **(REQUIRED)**

Output: An octet string keydata.

Actions: The following actions are performed:

1. $\text{keydata}=\mathbf{H}(K\|r\|c)$
2. Output octet string keydata

The key derivation function $\mathbf{KDF}(K,c)$ requires a suitable hash function denoted by $\mathbf{H}()$, i.e the bit-length of the hash function SHALL be greater or equal to the bit-length of the derived key. The hash value SHALL be interpreted as big-endian byte output.

Note: *The shared secret K is defined as an octet string. If the shared secret is generated with ECKA [6], the x-coordinate of the generated point SHALL be used.*

5.8.1 3DES

To derive 128-bit (112-bit excluding parity bits) 3DES [20] keys the hash function SHA-1 [18] SHALL be used and the following additional steps MUST be performed:

- Use octets 1 to 8 of keydata to form keydataA and octets 9 to 16 of keydata to form keydataB; additional octets are not used.
- Adjust the parity bits of keydataA and keydataB to form correct DES keys (OPTIONAL).

5.8.2 AES

To derive 128-bit AES [19] keys the hash function SHA-1 [18] SHALL be used and the following additional step MUST be performed:

- Use octets 1 to 16 of keydata; additional octets are not used.

To derive 192-bit and 256-bit AES [19] keys SHA-256 [18] SHALL be used. For 192-bit AES keys the following additional step MUST be performed:

- Use octets 1 to 24 of keydata; additional octets are not used.

5.9 Public Key Data Objects

A public key data object is a DER encoded TLV sequence consisting of an object identifier and several context specific data objects nested within the template 0x7F49:

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

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

5.9.1 Data Object Encoding

An unsigned integer SHALL be converted to an octet string using the binary representation of the integer in big-endian format. The minimum number of octets SHALL be used, i.e. leading octets of value 0x00 MUST NOT be used.

To encode elliptic curve points, uncompressed encoding according to [6] SHALL be used.

5.9.2 Diffie Hellman Public Keys

The data objects contained in a DH public key are shown in Table 9. The order of the data objects is fixed.

Data Object	***Abbrev.***	***Tag***	***Type***
Object Identifier		0x06	Object Identifier
Prime modulus	*p*	0x81	Unsigned Integer
Order of the subgroup	*q*	0x82	Unsigned Integer
Generator	*g*	0x83	Unsigned Integer
Public value	*y*	0x84	Unsigned Integer

Table 9: DH Public Key

Note: *The encoding of key components as unsigned integer implies that each of them is encoded over the least number of bytes possible, i.e. without preceding bytes set to 0x00. In particular, DH public key may be encoded over a number of bytes smaller than the number of bytes of the prime*

5.9.3 Elliptic Curve Public Keys

The data objects contained in an EC public key are shown in Table 10. The order of the data objects is fixed, CONDITIONAL domain parameters MUST be either all present, except the cofactor, or all absent as follows:

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

Table 10: EC Public Keys

5.9.4 Ephemeral Public Keys

For ephemeral public keys the format and the domain parameters are already known. Therefore, only the plain public key value, i.e. the public value y for Diffie-Hellman public keys and the public point Y for Elliptic Curve Public Keys, is used to convey the ephemeral public key in a context specific data object.

Note: *According to Section 1.5.2.3 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.*

5.10 Secure Messaging

5.10.1 Session Initiation

A *session* is started when secure messaging is established. Within a session the secure messaging keys (i.e. established by Basic Access Control, PACE or Chip Authentication) may be changed.

Secure Messaging is based on either 3DES [20] or AES [19] in encrypt-then-authenticate mode, i.e. data is padded, encrypted and afterwards the formatted encrypted data is input to the authentication calculation. The session keys SHALL be derived using the key derivation function described in Section 5.8.

Note: *Padding is always performed by the secure messaging layer, therefore the underlying message authentication code need not perform any internal padding.*

An unsigned integer SHALL be used as Send Sequence Counter (SSC). The bitsize of the SSC SHALL be equal to the blocksize of the block cipher used for Secure Messaging, i.e. 64 bit for 3DES and 128 bit for AES.

The SSC SHALL be increased every time before a command or response APDU is generated, i.e. if the starting value is x, in the next command the value of the SSC is $x+1$. The value of SSC for the first response is $x+2$.

If Secure Messaging is restarted, the SSC is used as follows:

- The commands used for key agreement are protected with the old session keys and old SSC. This applies in particular for the response of the last command used for session key agreement.
- The Send Sequence Counter is set to its new start value, i.e. within this specification the SSC is set to 0.
- The new session keys and the new SSC are used to protect subsequent commands/responses.

5.10.2 Session Termination

The MRTD chip MUST abort Secure Messaging if and only if a Secure Messaging error occurs or a plain APDU is received.

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

Note: *The MRTD chip MAY implicitly select the Master File when a session is terminated.*

5.10.3 3DES

3DES is specified in [20].

5.10.3.13DES Encryption

For message encryption two key 3DES SHALL be used in CBC-mode according to ISO/IEC 10116 [12] with key K_{Enc} and $IV = 0$.

5.10.3.23DES Authentication

For message authentication 3DES SHALL be used in Retail-mode according to ISO/IEC 9797-1 [14] MAC algorithm 3 with block cipher DES, key K_{MAC} and $IV = 0$. The datagram to be authenticated SHALL be prepended by the Send Sequence Counter.

5.10.4 AES

AES is specified in [19].

5.10.4.1AES Encryption

For message encryption AES SHALL be used in CBC-mode according to ISO/IEC 10116 [12] with key K_{Enc} and $IV = \mathbf{E}(K_{Enc}, SSC)$.

5.10.4.2AES Authentication

For message authentication AES SHALL be used in CMAC-mode [21] with K_{MAC} with a MAC length of 8 bytes. The datagram to be authenticated SHALL be prepended by the Send Sequence Counter.

A. Point Encoding for the ECDH-Integrated Mapping

A.1. High-level description of the point encoding method

The algorithm takes as inputs the curve parameters *(a, b, p, f)* where *(a, b)* are the curve coefficients, *p* is the characteristic of the prime field over which the curve

$$E : y^2 \equiv x^3 + ax + b \ (mod\ p)$$

is defined. The order of *E* is always of the form *fq* for some prime *q* and *f* is called the co-factor. PACE v2 requires the generation of a point that belongs to the *q*-subgroup of *E* that we denote by *E[q]*. The point encoding also takes as input a number *t* such that

$$0 < t < p$$

and returns, in constant time, a point that belongs to *E[q]*. As described in [4], point encoding comes in two flavors, depending on the coordinate system preferred by the implementation:

1. A first implementation, described in Section A.2, outputs the elliptic curve point in affine coordinates *(x, y)*;
2. An alternate implementation, presented in Section A.3, outputs the same point in Jacobian coordinates *(X, Y, Z)*.

Irrespective of which option is taken, the generated point is identical in the sense that

$$x = XZ^{-2} \ mod\ p \ and \ y = YZ^{-3} \ mod\ p$$

and the implementation of the subsequent phase of PACE v2 (the elliptic curve Diffie-Hellman key exchange phase) can therefore take advantage of using the option that best fits the interface of the cryptographic API that performs elliptic-curve operations.

As noted hereafter, point encoding for affine coordinates roughly requires two modular exponentiations modulo *p* whereas point encoding for Jacobian coordinates only requires a single one.

Note: *Note that for the two available implementations, point encoding explicitly requires that* $p \equiv 3 \ mod\ 4$.

A.2. Implementation for affine coordinates

A.2.1. Implementation for affine coordinates

The algorithm is implemented as follows:

Inputs: curve parameters *(a, b, p, f)* and *t* such that $0 < t < p$

Output: a point *(x, y)* in the prime-order subgroup *E[q]* of *E*

1. Compute $\alpha = -t^2 \ mod\ p$
2. Compute $X_2 = -ba^{-1}(1+(\alpha+\alpha^2)^{-1}) \ mod\ p$
3. Compute $X_3 = \alpha X_2 \ mod\ p$
4. Compute $h_2 = (X_2)^3 + a X_2 + b \ mod\ p$
5. Compute $h_3 = (X_3)^3 + a X_3 + b \ mod\ p$

6. Compute $U = t^3 h_2 \bmod p$
7. Compute $A = (h_2)^{p-1-(p+1)/4} \bmod p$
8. If $A^2 h_2 = 1 \bmod p$ define $(x, y) = (X_2, A\, h_2 \bmod p)$
9. Otherwise define $(x, y) = (X_3, A\, U \bmod p)$
10. Output $(x, y) = [f]\ (x, y)$.

A.2.2. Implementation Notes

Neglecting modular multiplications and additions, the execution time of the above implementation is dominated by two modular exponentiations:

- Step 2 can be rewritten

$$X_2 = -ba^{-1}(1+(\alpha+\alpha^2)^{-1}) = -b(1+\alpha+\alpha^2)\,(a(\alpha+\alpha^2))^{p-2} \bmod p$$

  which essentially amounts to a modular exponentiation with exponent $p-2$;
- Step 7 is a modular exponentiation with exponent $p-1-(p+1)/4$.

Note: *Step 10 requires a scalar multiplication by the co-factor f. For many curves, the co-factor is equal to 1 so that this scalar multiplication can be avoided.*

A.3. Implementation for Jacobian coordinates

A.3.1. Implementation for Jacobian coordinates

The algorithm is implemented as follows:

Inputs: curve parameters (a, b, p, f) and t such that $0 < t < p$

Output: a point (X, Y, Z) in the prime-order subgroup $E[q]$ of E

1. Compute $\alpha = -t^2 \bmod p$
2. Compute $Z = a(\alpha+\alpha^2) \bmod p$
3. Compute $X_2 = -bZ\,(1+\alpha+\alpha^2) \bmod p$
4. Compute $X_3 = \alpha\, X_2 \bmod p$
5. Compute $h_2 = (X_2)^3 + a\, X_2 Z^4 + b\, Z^6 \bmod p$
6. Compute $h_3 = (X_3)^3 + a\, X_3 Z^4 + b\, Z^6 \bmod p$
7. Compute $U = -\alpha\, t\, h_2 \bmod p$
8. Compute $A = (h_2)^{p-1-(p+1)/4} \bmod p$
9. If $A^2 h_2 = 1 \bmod p$ define $(X, Y, Z) = (X_2, A\, h_2 \bmod p, Z)$
10. Otherwise define (X, Y, Z) = (X_3, A U mod p, Z)
11. Output $(X, Y, Z) = [f]\,(X, Y, Z)$.

A.3.2. Implementation Notes

Neglecting modular multiplications and additions, the execution time of the above implementation is dominated by the single modular exponentiation of Step 7. Therefore, it is expected to be roughly twice faster than the implementation for affine coordinates.

Note: *The scalar multiplication in Step 10 can be completely avoided when the co-factor f is equal to 1.*

B. Challenge Semantics (Informative)

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

1. The terminal 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 terminal can as well generate this bit string in an unpredictable but (publicly) verifiable way, e.g. let SK_{PCD} be the terminal'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 terminal has indeed generated this challenge. Due to the transferability of the terminal's signature, any third party having trust in the terminal 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 terminals 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 a terminal 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.

References

[1] ANSI X9.42-2000, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, 1999
[2] Bender, Jens; Fischlin, Marc; and Kügler, Dennis, Security Analysis of the PACE Key-Agreement Protocol, Information Security – Proceedings of ISC 09, Springer-Verlag, 2009
[3] Bradner, Scott, RFC 2119: Key words for use in RFCs to indicate requirement levels, 1997
[4] Brier, Eric; Coron, Jean-Sébastien; Icart, Thomas; Madore, David; Randriam, Hugues; and Tibouch, Mehdi, Efficient Indifferentiable Hashing into Ordinary Elliptic Curves, Advances in Cryptology – CRYPTO 2010, Springer-Verlag, 2010
[5] BSI TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents Version 2.02, 2009
[6] BSI TR-03111, Elliptic Curve Cryptography (ECC) Version 2.0, 2012
[7] Cooper, David; Santesson, Stefan; Farrell, Stephen; Boeyen, Sharon; Housley, Russell and Polk, Tim RFC 5280, Internet X.509 public key infrastructure - certificate and certificate revocation list (CRL) profile, 2002
[8] Coron, Jean-Sébastien; Gouget, Aline; and Paillier, Pascal, Password Authenticated Secure Channel v5 (PASC5), 2009, available at http://www2.afnor.org/espace_normalisation/structure.aspx?commid=49956&lang=french
[9] ICAO Doc 9303, Machine Readable Travel Documents - Part 1: Machine Readable Passport, Volume 2: Specifications for electronically enabled passports with biometric identification capabilities, 6th Edition, 2006
[10] ICAO Doc 9303, Machine Readable Travel Documents - Part 3: Machine Readable Official Travel Documents, Volume 2: Specifications for electronically enabled official travel documents with biometric identification capabilities, 3rd Edition, 2008
[11] Icart, Thomas, How to Hash onto Elliptic Curves, Advances in Cryptology – CRYPTO 2009, Springer-Verlag, 2009
[12] ISO/IEC 10116:2006, Information technology − Security techniques − Modes of operation for an n-bit block cipher, 2006
[13] ISO/IEC 7816-4:2013, Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange, 2013
[14] ISO/IEC 9797-1:1999, Information technology − Security techniques − Message Authentication Codes (MACs) − Part 1: Mechanisms using a block cipher, 1999
[15] Lepinski, Matt; Kent, Stephen, RFC 5114: dditional Diffie-Hellman Groups for Use with IETF Standards, 2008
[16] Lochter, Manfred; Merkle, Johannes, RFC 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, 2010
[17] NIST FIPS 186-3, Digital Signature Standard (DSS), 2009
[18] NIST FIPS PUB 180-2, Secure hash standard (and Change Notice to include SHA-224), 2002
[19] NIST FIPS PUB 197, Specification for the Advanced Encryption Standard (AES), 2001
[20] NIST FIPS PUB 46-3, Data Encryption Standard (DES), 1999
[21] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005
[22] Rescorla, Eric, RFC 2631: Diffie-Hellman key agreement method, 1999
[23] RSA Laboratories, PKCS#3: Diffie-Hellman key-agreement standard, 1993

[24] RSA Laboratories RSA Laboratories Technical Note, PKCS#3: Diffie-Hellman key-agreement standard, 1993
[25] Sagem, MorphoMapping Patents FR09-54043 and FR09-54053, 2009