MACHINE READABLE TRAVEL DOCUMENTS

TECHNICAL REPORT

Supplemental Access Control for Machine Readable Travel Documents

Version - 1.01

Date – November 11, 2010

Published by authority of the Secretary General

ISO/IEC JTC1 SC17 WG3/TF5

FOR THE

INTERNATIONAL CIVIL AVIATION ORGANIZATION

File	: TR-PACE-101-final2.odt
Author	: ISO/IEC JTC1 SC17 WG3/TF5

Release Control

Release	*Date*	*Description*
1.00	2009-03-23	Initial public version with temporary restrictions on the usage of the elliptic curve integrated mapping.
1.01	2010-11-11	Changed title to avoid ambiguities in the naming. Removed restrictions on usage of the elliptic curve integrated mapping.

Table of Contents

1. Introduction

This Technical Report specifies an access control mechanism that is supplementary to Basic Access Control. It is based on Password Authenticated Connection Establishment (PACE) as specified in [5] and complementary contributions [7], [10], [4], [2] and [23]. This framework based on PACE v2 allows for various implementation options (cf. Section 2.3, 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 [23] may only be available for implementations in the context of ISO 7501.*

1.1 Background

Doc 9303 [8], [9] 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 it's 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.

1.2 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.

1.3 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.

1.4 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 [8], [9] 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.

1.5.2 Keys and Passwords

The following keys 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
Session Keys	KS_{ENC}, KS_{MAC}	Session keys are ephemeral keys established by an access control mechanism (BAC or PACE) for Secure Messaging.
PACE Key	K_{π},	PACE keys are derived from a password (CAN or MRZ).
Key Agreement	**KA**( )	(Elliptic curve) Diffie Hellman key agreement to generate a shared secret.
Key Derivation Function	**KDF**( )	Key derivation function to generate a symmetric key from a shared secret.
Hash	**H**( )	Cryptographic hash function.
Encryption and Decryption	**E**( ), **D**( )	Symmetric key encryption / decryption.
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 Machine Readable Zone as defined by Doc 9303.

2. Overview

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

2.1 General outline

PACE establishes Secure Messaging between an MRTD chip and an inspection system based on weak (short) passwords. It 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 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.

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.*

2.2 Inspection procedure

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 Contol 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:

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

1. Read CardAccess

The inspection system SHALL read the file CardAccess (cf. Section 3.1.5) 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 SHOULD 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 2.3 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 [8], [9], 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.

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 $\mathbf{Map}()$
$\tilde{D} = \mathbf{Map}(D_{PICC}, s)$		$\tilde{D} = \mathbf{Map}(D_{PICC}, s)$
choose random ephemeral key pair $(\widetilde{SK_{PICC}}, \widetilde{PK_{PICC}}, \tilde{D})$		choose random ephemeral key pair $(\widetilde{SK_{PCD}}, \widetilde{PK_{PCD}}, \tilde{D})$
check that $\widetilde{PK_{PCD}} \neq \widetilde{PK_{PICC}}$	$\langle \frac{\widetilde{PK_{PCD}}}{\widetilde{PK_{PICC}}} \rangle$	check that $\widetilde{PK_{PICC}} \neq \widetilde{PK_{PCD}}$
$K = \mathbf{KA}(\widetilde{SK_{PICC}}, \widetilde{PK_{PCD}}, \tilde{D})$		$K = \mathbf{KA}(\widetilde{SK_{PCD}}, \widetilde{PK_{PICC}}, \tilde{D})$
$T_{PICC} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK_{PCD}})$	$\langle \frac{T_{PCD}}{T_{PICC}} \rangle$	$T_{PCD} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK_{PICC}})$
verify T_{PCD}		verify T_{PICC}

Figure 2.1: PACE

2.3 Password Authenticated Connection Establishment (PACE)

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).

2.3.1 Protocol Setup

The inspection system reads the parameters for PACE supported by the MRTD chip from the file CardAccess (cf. Section 3.1.5) and selects the parameters to be used.

The following commands SHALL be used:

- Read Binary as specified in [8], [9].
- MSE:Set AT as specified in Section 3.2.1.

2.3.2 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 3.2.2. A simplified version of the protocol is also shown in Figure 2.1.

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:
   a) They exchange additional data required for the mapping of the nonce (cf. Section 3.3.2):
      - For the generic mapping the MRTD chip and the inspection system exchange ephemeral key public keys.
      - For the integrated mapping the inspection system sends an additional nonce to the MRTD chip.
   b) They compute the ephemeral domain parameters $\tilde{D} = \mathbf{Map}(D_{PICC}, s, ...)$ as described in Section 4.3.
   c) They perform an anonymous Diffie-Hellman key agreement (cf. Section 4.1) based on the ephemeral domain parameters and generate the shared secret
   $$K = \mathbf{KA}(\widetilde{SK_{PICC}}, \widetilde{PK_{PCD}}, \tilde{D}) = \mathbf{KA}(\widetilde{SK_{PCD}}, \widetilde{PK_{PICC}}, \tilde{D}).$$
   d) During Diffie-Hellman key agreement, each party SHOULD check that the two public keys $\widetilde{PK_{PICC}}$ and $\widetilde{PK_{PCD}}$ differ.
   e) They derive session keys $KS_{MAC} = \mathbf{KDF_{MAC}}(K)$ and $KS_{Enc} = \mathbf{KDF_{Enc}}(K)$ as described in Section 4.2.
   f) They exchange and verify the authentication token $T_{PCD} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK_{PICC}})$ and $T_{PICC} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK_{PCD}})$ as described in Section 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. Technical specifications

This section describes the technical specification required to implement Supplemental Access Control.

3.1 Logical Data Structure

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
}
```

3.1.1 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.

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 proprietary domain parameters a `PACEDomainParameterInfo` MUST be present.

Security Infos for other Protocols

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

3.1.2 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, 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. Table 6) or provides multiple proprietary domain parameters for PACE.

```
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),
  version     INTEGER, -- MUST be 2
  parameterId INTEGER OPTIONAL
}
```

3.1.3 PACEDomainParameterInfo

This data structure is REQUIRED if the MRTD chip provides proprietary 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 proprietary 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),
  domainParameter AlgorithmIdentifier,
  parameterId     INTEGER OPTIONAL
}
```

The domain parameters for PACE MUST be provided as `AlgorithmIdentifier`. The data structure `AlgorithmIdentifier` is defined as follows:

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

The object identifier `algorithm` SHALL be `dhpublicnumber` or `ecPublicKey` for DH or ECDH, respectively.

```
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
}
```

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

***Note:** The MRTD chip MAY support more than one set of proprietary 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 6. Inspection systems MUST NOT use proprietary 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 4.5.3.

3.1.4 PACE Object Identifier

The following Object Identifier SHALL be used:

```
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}
```

3.1.5 Storage on the Chip

The MRTD chip SHALL provide `SecurityInfos` in a transparent elementary file *CardAccess* contained in the master file (cf. Table 1):

File Name	***EF.CardAccess***
File ID	0x011C
Short File ID	0x1C
Read Access	ALWAYS
Write Access	NEVER
Size	variable
Content	DER encoded `SecurityInfos`

Table 1: Elementary File CardAccess

- The file CardAccess SHALL contain the relevant `SecurityInfos` that are required for PACE:
  - `PACEInfo`
  - `PACEDomainParameterInfo`
- The full set of `SecurityInfos` SHALL additionally be stored in DG14 of the ePassport Application.

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

3.2 Application Protocol Data Units

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

1. MSE:Set AT
2. General Authenticate

3.2.1 MSE:Set AT

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

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

3.2.2 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 3.3.	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.	

3.3 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

3.3.1 Encrypted Nonce

The encrypted nonce (cf. Section 4.3) SHALL be encoded as octet string.

3.3.2 Mapping Data

The exchanged data is specific to the used mapping:

Generic Mapping

The ephemeral public keys (cf. Section 4.3 and Section 4.5.3) SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

Integrated Mapping

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

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

3.3.3 Public Keys

The public keys SHALL be encoded as described in Section 4.5.3.

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

3.3.4 Authentication Token

The authentication token (cf. Section 4.4) SHALL be encoded as octet string.

3.4 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 [12].

3.4.1 Errors

If the MRTD chip expects the end of the chain, but receives a command that is not marked as the last command, the MRTD chip SHALL indicate that the last command in a chain was expected with status bytes 0x6883.

4. Cryptographic Specifications

This section contains the cryptographic details of the specification.

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

4.1 Key Agreement Algorithms

This specification supports Diffie-Hellman and Elliptic Curve Diffie-Hellman key agreement as summarized in Table 2.

Algorithm / Format	***DH***	***ECDH***
Key Agreement Algorithm	PKCS#3 [22]	ECKA [6]
X.509 Public Key Format	X9.42 [1]	ECC [6]
TLV Public Key Format	TLV, cf. Section 4.5.1	TLV, cf. Section 4.5.2
Ephemeral Public Key Validation	RFC 2631 [21]	ECC [6]

Table 2: Algorithms and Formats for Key Agreement

4.1.1 DH

For PACE with DH the respective algorithms and formats from Table 2 and Table 3 MUST be used.

OID	***Mapping***	***Sym. Cipher***	***Key len 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	Generic	AES	128	CBC / CMAC	CMAC
id-PACE-DH-GM-AES-CBC-CMAC-192	Generic	AES	192	CBC / CMAC	CMAC
id-PACE-DH-GM-AES-CBC-CMAC-256	Generic	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	Integrated	AES	128	CBC / CMAC	CMAC
id-PACE-DH-IM-AES-CBC-CMAC-192	Integrated	AES	192	CBC / CMAC	CMAC
id-PACE-DH-IM-AES-CBC-CMAC-256	Integrated	AES	256	CBC / CMAC	CMAC

Table 3: Object Identifiers for PACE with DH

4.1.2 ECDH

For PACE with ECDH the respective algorithms and formats from Table 2 and Table 4 MUST be used. Only prime curves with uncompressed points SHALL be used.

OID	***Mapping***	***Sym. Cipher***	***Key len*** 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	Generic	AES	128	CBC / CMAC	CMAC
id-PACE-ECDH-GM-AES-CBC-CMAC-192	Generic	AES	192	CBC / CMAC	CMAC
id-PACE-ECDH-GM-AES-CBC-CMAC-256	Generic	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	Integrated	AES	128	CBC / CMAC	CMAC
id-PACE-ECDH-IM-AES-CBC-CMAC-192	Integrated	AES	192	CBC / CMAC	CMAC
id-PACE-ECDH-IM-AES-CBC-CMAC-256	Integrated	AES	256	CBC / CMAC	CMAC

Table 4: Object Identifiers for PACE with ECDH

4.1.3 Standardized Domain Parameters

The standardized domain parameters described in Table 6 SHOULD be used. Proprietary domain parameters provided by `PACEDomainParameterInfo` MUST NOT use those IDs reserved for standardized domain parameters.

4.2 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 5.

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

Table 5: Encoding of Passwords

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

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

* This curve cannot be used with the integrated mapping.

Table 6: Standardized Domain Parameters

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. $\text{keydata} = \mathbf{H}(K \| c)$
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.*

4.2.1 3DES

To derive 128-bit (112-bit excluding parity bits) 3DES [19] keys the hash function SHA-1 [17] 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).

4.2.2 AES

To derive 128-bit AES [18] keys the hash function SHA-1 [17] 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 [18] keys SHA-256 [17] 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.

4.3 Encrypting and Mapping Nonces

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

- The nonce s SHALL be encrypted in CBC mode according to ISO 10116 [11] 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 random generator using an algorithm-specific mapping function **Map**.
- The nonce $t \in_R \{0 \ldots 2^k - 1\}$ required in the Integrated Mapping SHALL be sent in clear.

To map the nonce s or the nonces s, t into the cryptographic group either the generic mapping or the integrated mapping, respectively, SHALL be used.

4.3.1 ECDH Mapping

Let G and $\tilde{G}$ be the static and an ephemeral base point on the elliptic curve.

Generic Mapping

The function $\mathbf{Map}: G \mapsto \tilde{G}$ is defined as $\tilde{G} = s \cdot G + H$, 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].

Note: *The key agreement algorithm ECKA prevents small subgroup attacks by using compatible cofactor multiplication.*

Integrated Mapping

The function $\mathbf{Map}: G \mapsto \tilde{G}$ is defined as $\tilde{G} = f_G(\mathbf{R}_p(s, t))$, 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

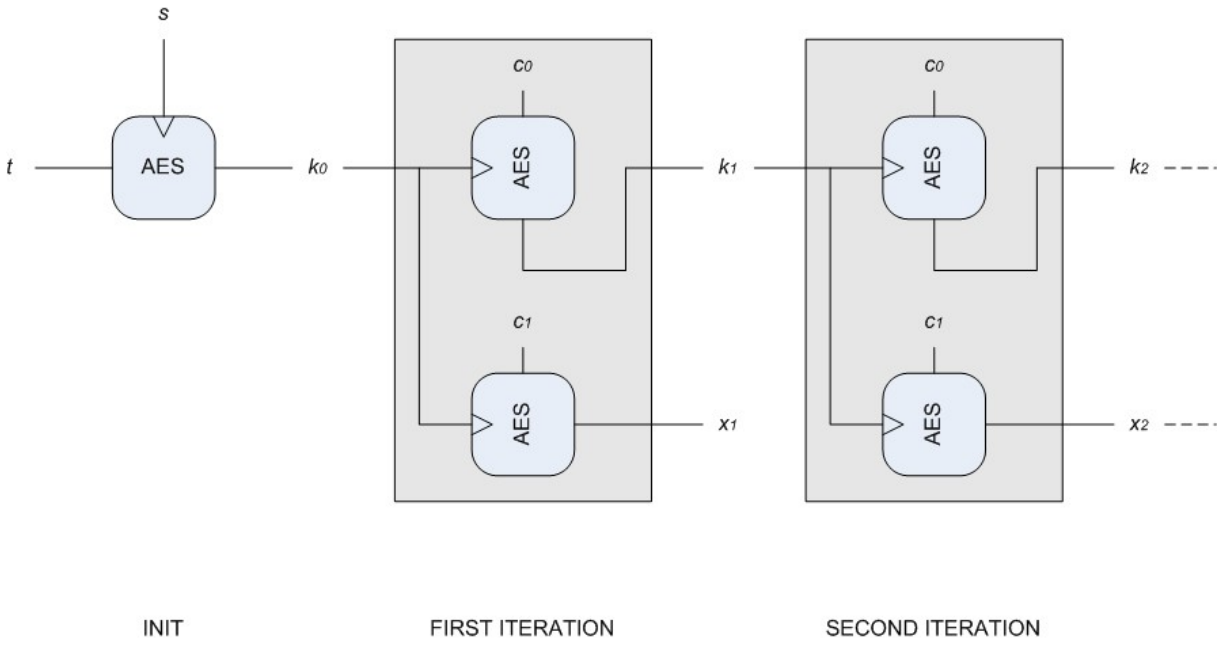

Figure 4.1: The function R(s,t) using AES as block cipher

chip. The pseudo-random function $\mathbf{R}_p()$ is described in Section 4.3.3. The function $f_G()$ is defined in [4] and [23].

4.3.2 DH Mapping

Let g and $\tilde{g}$ be the static and an ephemeral generator.

Generic Mapping

The function $\mathbf{Map}: g \mapsto \tilde{g}$ is defined as $\tilde{g} = g^s \cdot h$, 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.

Note: *The public key validation method described in RFC 2631 [21] MUST be used to prevent small subgroup attacks.*

Integrated Mapping

The function $\mathbf{Map}: g \mapsto \tilde{g}$ is defined as $\tilde{g} = f_g(\mathbf{R}_p(s,t))$, 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 4.3.3. 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 $\tilde{g} \neq 1$.

4.3.3 Pseudorandom Number Mapping

The function $\mathbf{R}_p(s,t)$ is a function that maps octet strings s and t each 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 4.1. The construction is

based on the respective block cipher $\mathbf{E}()$ in CBC mode according to ISO 10116 [11] with $IV = 0$, where k is the key size (in bits) of $\mathbf{E}()$. Where required, the outputs x_i and k_i MUST be truncated to bit size k. The value n SHALL be selected as smallest number, such that $n \cdot k \geq \log_2 p + 64$.

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

The constants c_0 and c_1 are defined as follows:

- For 3DES and AES-128 ($k = 128$):
  - c_0=0x`a668892a7c41e3ca739f40b057d85904`
  - c_1=0x`a4e136ac725f738b01c1f60217c188ad`
- For AES-192 ($k = 192$):
  - c_0=0x`6db6525e849de0b546d2707146de4441ece0428618fd3f9c`
  - c_1=0x`2177e4552ab798a6c14678715b4b340f9fc895df5b133a33`
- For AES-256 ($k = 256$):
  - c_0=0x`d463d65234124ef7897054986dca0a174e28df758cbaa03f240616414d5a1676`
  - c_1=0x`54bd7255f0aaf831bec3423fcf39d69b6cbf066677d0faae5aadd99df8e53517`

4.4 Authentication token

The authentication token SHALL be computed over a public key data object (cf. Section 4.5) containing the object identifier as indicated in MSE:Set AT (cf. Section 3.2.1), and the received ephemeral public key (i.e. excluding the domain parameters, cf. Section 4.5.3) using an authentication code and the key KS_{MAC} derived from the key agreement.

Note: *Padding is performed internally by the message authentication code.*

4.4.1 3DES

3DES [19] SHALL be used in Retail-mode according to ISO/IEC 9797-1 [13] MAC algorithm 3 / padding method 2 with block cipher DES and $IV = 0$.

4.4.2 AES

AES [18] SHALL be used in CMAC-mode [20] with a MAC length of 8 bytes.

4.5 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.

- 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.

4.5.1 Diffie Hellman Public Keys

The data objects contained in a DH public key are shown in Table 7. 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 7: DH Public Key

4.5.2 Elliptic Curve Public Keys

The data objects contained in an EC public key are shown in Table 8. 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 8: EC Public Keys

4.5.3 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.

4.6 Secure Messaging

Secure Messaging is based on either 3DES [19] or AES [18] 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 Appendix 4.2.

Note: *Padding is always performed by the secure messaging layer, s.th. 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.

4.6.1 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 stored session keys and reset the terminal's access rights.

4.6.2 3DES

3DES is specified in [19].

3DES Encryption

For message encryption two key 3DES SHALL be used in CBC-mode according to ISO 10116 [11] with key K_{Enc} and $IV = 0$.

3DES Authentication

For message authentication 3DES SHALL be used in Retail-mode according to ISO/IEC 9797-1 [13] 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.

4.6.3 AES

AES is specified in [18].

AES Encryption

For message encryption AES SHALL be used in CBC-mode according to ISO 10116 [11] with key K_{Enc} and $IV = \mathbf{E}(K_{Enc}, SSC)$.

AES Authentication

For message authentication AES SHALL be used in CMAC-mode [20] with K_{MAC} with a MAC length of 8 bytes. The datagram to be authenticated SHALL be prepended by the Send Sequence Counter.

5. Point Encoding for the Integrated Mapping

5.1 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 5.2.1, outputs the elliptic curve point in affine coordinates (x, y);
2. An alternate implementation, presented in Section 5.3.1, 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* $p \equiv 3 \ mod\ 4$.

5.2 Implementation for affine coordinates

5.2.1 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 $(x, y) = [f]\,(x, y)$.

5.2.2 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.*

5.3 Implementation for Jacobian coordinates

5.3.1 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 $(X, Y, Z) = [f]\,(X, Y, Z)$.

5.3.2 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.*

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 1.11, 2009

[7] 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

[8] 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

[9] 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

[10] Icart, Thomas, How to Hash onto Elliptic Curves, Advances in Cryptology – CRYPTO 2009, Springer-Verlag, 2009

[11] ISO/IEC 10116:2006, Information technology − Security techniques − Modes of operation for an n-bit block cipher, 2006

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

[13] ISO/IEC 9797-1:1999, Information technology − Security techniques − Message Authentication Codes (MACs) − Part 1: Mechanisms using a block cipher, 1999

[14] Lepinski, Matt; Kent, Stephen, RFC 5114: dditional Diffie-Hellman Groups for Use with IETF Standards, 2008

[15] Lochter, Manfred; Merkle, Johannes, RFC 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, 2010

[16] NIST FIPS 186-3, Digital Signature Standard (DSS), 2009

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

[18] NIST FIPS PUB 197, Specification for the Advanced Encryption Standard (AES), 2001

[19] NIST FIPS PUB 46-3, Data Encryption Standard (DES), 1999

[20] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005

[21] Rescorla, Eric, RFC 2631: Diffie-Hellman key agreement method, 1999

[22] RSA Laboratories, PKCS#3: Diffie-Hellman key-agreement standard, 1993

[23] Sagem, MorphoMapping Patents FR09-54043 and FR09-54053, 2009