INTERNATIONAL STANDARD

ISO/IEC 18013-3

First edition
2009-03-01

Information technology — Personal identification — ISO-compliant driving licence

Part 3: Access control, authentication and integrity validation

Technologies de l'information — Identification des personnes — Permis de conduire conforme à l'ISO

Partie 3: Contrôle d'accès, authentification et validation d'intégrité

Reference number
ISO/IEC 18013-3:2009(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 18013-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

ISO/IEC 18013 consists of the following parts, under the general title *Information technology — Personal identification — ISO-compliant driving licence*:

- *Part 1: Physical characteristics and basic data set.* Part 1 defines the basic terms for ISO/IEC 18013, including physical characteristics, basic data element set, visual layout, and physical security features.

- *Part 2: Machine-readable technologies.* Part 2 defines the technologies that may be used for ISO/IEC 18013, including the logical data structure and data mapping for each technology.

- *Part 3: Access control, authentication and integrity validation.* Part 3 defines the electronic security features that may be incorporated under ISO/IEC 18013, including mechanisms for controlling access to data, verifying the origin of an ISO-compliant driving licence, and confirming data integrity.

Introduction

This part of ISO/IEC 18013 prescribes requirements for the implementation of mechanisms to control access to data recorded in the machine-readable technology on an ISO-compliant driving licence (IDL), verifying the origin of an IDL, and confirming data integrity.

One of the functions of an IDL is to facilitate international interchange. Whilst storing data in machine-readable form on the IDL supports this function by speeding up data input and eliminating transcription errors, certain machine-readable technologies are vulnerable to being read without the knowledge of the card holder and to other means of unauthorized access by unintended persons, that is other than driving licence or law enforcement authorities. Controlling access to IDL data stored in machine-readable form protects the data on the card from being read remotely by electronic means without the knowledge of the card holder.

Identifying falsified driving licences, or an alteration to the human-readable data on authentic driving licences present a major problem for driving licence and law enforcement authorities, both domestically and in the context of international interchange. Verifying the authenticity of an IDL and confirming the integrity of the data recorded on an IDL provide driving licence and law enforcement authorities with a means to identify an authentic IDL from a falsified or altered one in the interests of traffic law enforcement and other traffic safety processes.

Information technology — Personal identification — ISO-compliant driving licence —

Part 3: Access control, authentication and integrity validation

1 Scope

ISO/IEC 18013 establishes guidelines for the design format and data content of an ISO-compliant driving licence (IDL) with regard to human-readable features (ISO/IEC 18013-1), machine-readable technologies (ISO/IEC 18013-2), and access control, authentication and integrity validation (ISO/IEC 18013-3). It creates a common basis for international use and mutual recognition of the IDL without impeding individual countries/states to apply their privacy rules and national/community/regional motor vehicle authorities in taking care of their specific needs.

This part of ISO/IEC 18013

a) is based on the machine-readable data content specified in ISO/IEC 18013-2;

b) specifies mechanisms and rules available to issuing authorities (IAs) for

   1) access control (i.e. limiting access to the machine-readable data recorded on the IDL),

   2) document authentication (i.e. confirming that the document was issued by the claimed IA),

   3) data integrity validation (i.e. confirming that the data has not been changed since issuing).

This part of ISO/IEC 18013 does not address issues related to the subsequent use of data obtained from the IDL, e.g. privacy issues.

2 Conformance

A driving licence is in conformance with this part of ISO/IEC 18013 if it meets all mandatory requirements specified directly or by reference herein. Compliance with ISO/IEC 18013-2 is required for compliance with this part of ISO/IEC 18013.

Compliance with ISO/IEC 18013-1 is not required for compliance with this part of ISO/IEC 18013. Conversely, the incorporation of a machine-readable technology which is not compliant with this part of ISO/IEC 18013 does not render the IDL non-compliant with ISO/IEC 18013-1.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 1831:1980, *Printing specifications for optical character recognition*

ISO/IEC 7816-4:2005, *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*

ISO/IEC 7816-8:2004, *Identification cards — Integrated circuit cards — Part 8: Commands for security operations*

ISO/IEC 8825-1:2002, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 8859-1:1998, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

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

ISO/IEC 10118-3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

ISO/IEC 11770-2:1996, *Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques*

ISO/IEC 11770-2:1996/Cor.1:2005, *Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques — Corrigendum 1*

ISO/IEC 11770-3, *Information technology — Security techniques — Key management — Part 3: Mechanisms using asymmetric techniques*

ISO/IEC 18013-1, *Information technology — Personal identification — ISO-compliant driving licence — Part 1: Physical characteristics and basic data set*

ISO/IEC 18013-2, *Information technology — Personal identification — ISO-compliant driving licence — Part 2: Machine-readable technologies*

ISO/IEC 18033-3:2005, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

ISO/IEC 18033-3:2005/Cor.1:2006, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers — Corrigendum 1*

ISO/IEC 18033-3:2005/Cor.2:2007, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers — Corrigendum 2*

ANSI X9.62:2005, *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*

FIPS 186-2 (including Change Notice), *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication, National Institute of Standards and Technology, 27 January 2000

NIST SP 800-38B, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, May 2005

RFC 2631, E. Rescorla, *Diffie-Hellman Key Agreement Method*, June 1999, http://www.ietf.org/rfc.html

RFC 3279, W. Polk et al., *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, April 2002, http://www.ietf.org/rfc.html

RFC 3280, R. Housley et al., *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, April 2002, http://www.ietf.org/rfc.html

RFC 3369, R. Hously, *Cryptographic Message Syntax*, August 2002, http://www.ietf.org/rfc.html

RFC 4055, J. Schaad, B. Kaliski, R. Housley, *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, June 2005, http://www.ietf.org/rfc.html

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18013-1, ISO/IEC 18013-2 and the following apply.

4.1
active authentication
mechanism that uses information stored in a secure area of a secure integrated circuit (SIC) to confirm that the SIC and the other machine-readable data were issued together

NOTE See 8.2.

4.2
basic access protection
BAP
mechanism to confirm that an inspection system (IS) has physical access to a proximity integrated circuit card (PICC) before the IS is allowed access to the data stored on the PICC and to ensure that communication between the IS and the PICC (once access is authorized) is protected

NOTE See 8.5 and Annex B.

4.3
chip authentication
ephemeral-static key agreement protocol that provides authentication of the secure integrated circuit and strong secure messaging

NOTE See 8.6 and C.3.

4.4
clone
unauthorized exact copy of a document that has the same security characteristics as the original document and that cannot be distinguished from the legitimate one

4.5
eavesdropping
unauthorized interception and interpretation of information-bearing emanations

NOTE Adapted from ISO/IEC 2382-8:1998, 08.05.25.

4.6
extended access protection
EAP
protocol for limiting access to select optional data groups to reading authorities

NOTE See 8.6.

4.7
input string
string of characters printed on an ISO-compliant driving licence [as human-readable text, optionally (or by specification) accompanied by or consisting of a machine-readable rendering thereof] used as input (either manually or automatically through the use of suitable equipment) for the non-match alert and basic access protection mechanisms

4.8
issuing authority
IA
licensing authority, or issuing country if separate licensing authorities have not been authorized, which applies a digital signature to an ISO-compliant driving licence and is responsible for the associated key management

NOTE Adapted from ISO/IEC 18013-1.

4.9
non-match alert
mechanism to detect any differences between the machine-readable information and (some of) the human-readable information on an ISO-compliant driving licence

NOTE See 8.4.

4.10
passive authentication
mechanism to confirm that machine-readable data on an ISO-compliant driving licence (IDL) has not been changed since the IDL was issued

NOTE See 8.1.

4.11
pseudo issuing authority
PIA
authority that does not issue ISO-compliant driving licences [but that is similar to an issuing authority (IA) in all other respects] and which does not issue document keys, but which does have a root key pair with which it can sign documents of other IAs or PIAs that it trusts

4.12
public key infrastructure
PKI
technologies and products using public key (asymmetric) cryptography

NOTE Both passive authentication and extended access protection use this technology.

4.13
reading authority
RA
authorized entity reading the machine-readable data on an ISO-compliant driving licence (IDL)

NOTE Driving licence authorities other than the authority that issued the IDL and law enforcement authorities are examples of reading authorities.

4.14
reference string
string of characters used as a reference against which to compare the input string when using the non-match alert mechanism, and used for session key calculation purposes by the secure integrated circuit during execution of the basic access protection mechanism

4.15
scanning area identifier
SAI
one or more graphical elements that demarcate an input string

4.16
secure integrated circuit
SIC
integrated circuit that includes both a security feature (or security features), and memory and/or a central processing unit

NOTE 1 An integrated circuit card with contacts and a proximity integrated circuit card (PICC) are examples of a SIC.

NOTE 2 A SIC can be embedded in different solutions, for example in ID-1 sized cards (as used for the ISO-compliant driving licence) and in a booklet (as found in passports).

4.17
secure memory
integrated circuit (IC) memory of which the content (once populated by an issuing authority during the personalization process) is accessible only by the IC operating system for internal use, and cannot be made available by the operating system to any reading device

4.18
skimming
reading data from a proximity integrated circuit card (PICC) without the card holder's awareness

4.19
trust chain
sequential set of trust points that a verifying authority references to verify a specific issuing authority's public root key

4.20
trust model
description of the functional and logical aspects of a traditional public key infrastructure, specifically excluding technical implementation details

4.21
trust network
component of a trust model that describes the trust relationships and chains between issuing authorities

4.22
trust point
issuing authority or pseudo issuing authority that publishes a trust list (and the related public root keys) that verifying entities can reference

4.23
twinning
copying the data and/or integrated circuit of a physically and/or biometrically similar driver to the attacker’s integrated circuit or ISO-compliant driving licence

4.24
unpacked BCD
binary coding of a sequence of integers using 4 bits for each integer (where the bit weights are 8421) and encoding one integer in the least significant bits of each byte

NOTE Only unsigned BCD is used in this part of ISO/IEC 18013.

4.25
verifying authority
VA
verifying entity that is part of a trust network, i.e. that also is an issuing authority or a pseudo issuing authority

NOTE 1 Not all verifying entities are VAs: A car rental company can be a verifying entity, but is not a VA as it is not part of the trust network.

NOTE 2 VAs can be divided into immediate VAs and non-immediate VAs.

4.25.1
immediate VA
VA that acquired the public root key of the issuing authority via out-of-band means

4.25.2
non-immediate VA
VA that acquired the public root key of the issuing authority from another VA

4.26
verifying entity
entity that tries to determine if a digital signature is valid (i.e. if the data to which a certificate has been applied has not been changed, and if the signature was generated by the issuing authority the verifying entity expects)

5 Abbreviated terms

APDU	application protocol data unit
BAC	basic access control
BAP	basic access protection
BCD	binary coded decimal
BER-TLV	basic encoding rules – tag-length-value (see ISO/IEC 8825-1:2002)
CA	certification authority
CBC	cipher block chaining
DER	distinguished encoding rules (see ISO/IEC 8825-1:2002)
DF	dedicated file
DG	data group
DO	data object
DST	control reference template for digital signature (see ISO/IEC 7816-4:2005)
EAP	extended access protection
EF	elementary file
IA	issuing authority
IC	integrated circuit
ICC	integrated circuit card
IDL	ISO-compliant driving licence
IS	inspection system
IV	initialization vector
KAT	control reference template for key agreement (see ISO/IEC 7816-4:2005)
LDS	logical data structure
MAC	message authentication code
MSE	manage security environment (see ISO/IEC 7816-4:2005)
OCR	optical character recognition
OID	object identifier
PIA	pseudo issuing authority
PIC	proximity integrated circuit

PICC	proximity integrated circuit card
PKI	public key infrastructure
PSO	perform security operation (see ISO/IEC 7816-4:2005)
RA	reading authority
SAI	scanning area identifier
SIC	secure integrated circuit
SM	secure messaging
SOD	document security object
SSC	send sequence counter
TRCA	trust root certificate authority
UTC	coordinated universal time
VA	verifying authority
2D	two-dimensional

6 Functional requirements

6.1 Access control

Access control can be broken down into the following functional requirements:

a) Prevent skimming of machine-readable data on a PICC by ensuring that physical access to the IDL is acquired prior to reading.

b) Prevent unnoticed alteration of communication between a reader and a SIC.

c) Prevent eavesdropping between a reader and a SIC.

d) Selectively restrict access to specific optional machine-readable data groups for specific reading authorities.

6.2 Document authentication

Document authentication can functionally be established by allowing for verification of the origin of an IDL.

6.3 Data integrity validation

Data integrity validation can be broken down into the following functional requirements:

a) Verify that the IDL (including the machine-readable data) is not a clone of another IDL. A cloning attempt can schematically be illustrated as shown in Figure 1.

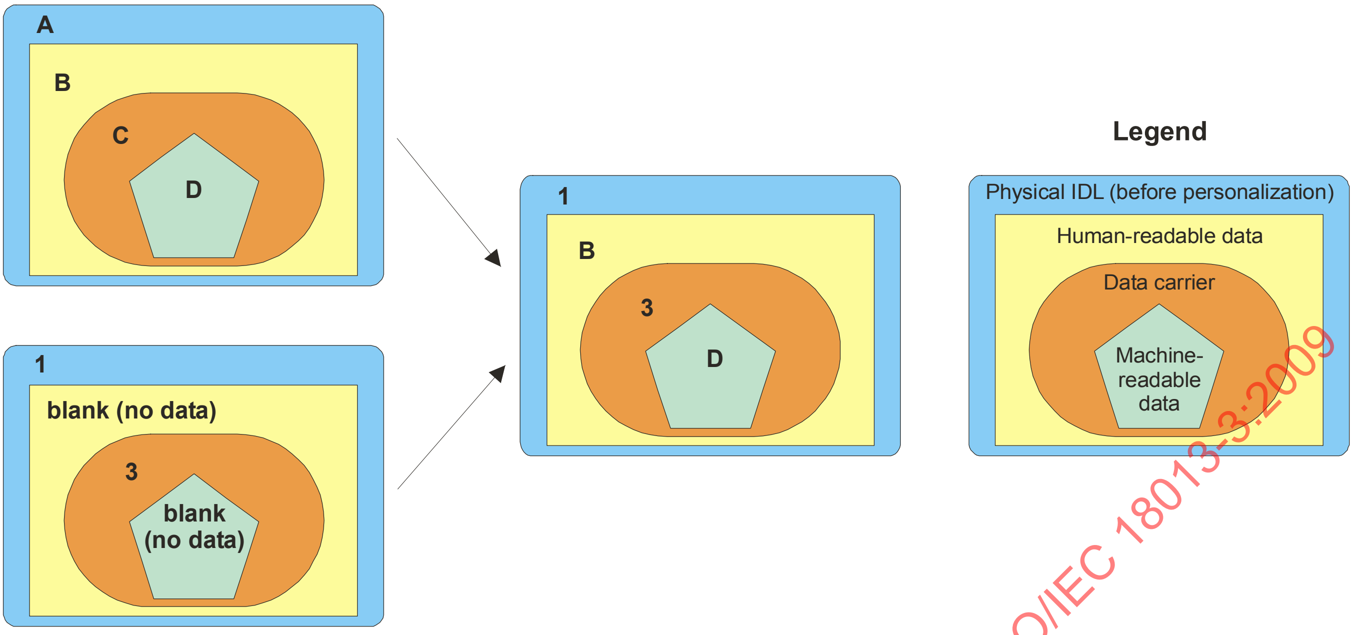


Figure 1 — Data integrity validation: IDL cloning

b) Protect against the exchange of machine-readable data carriers between otherwise authentic IDLs[1]. This type of attack can schematically be illustrated as shown in Figure 2.

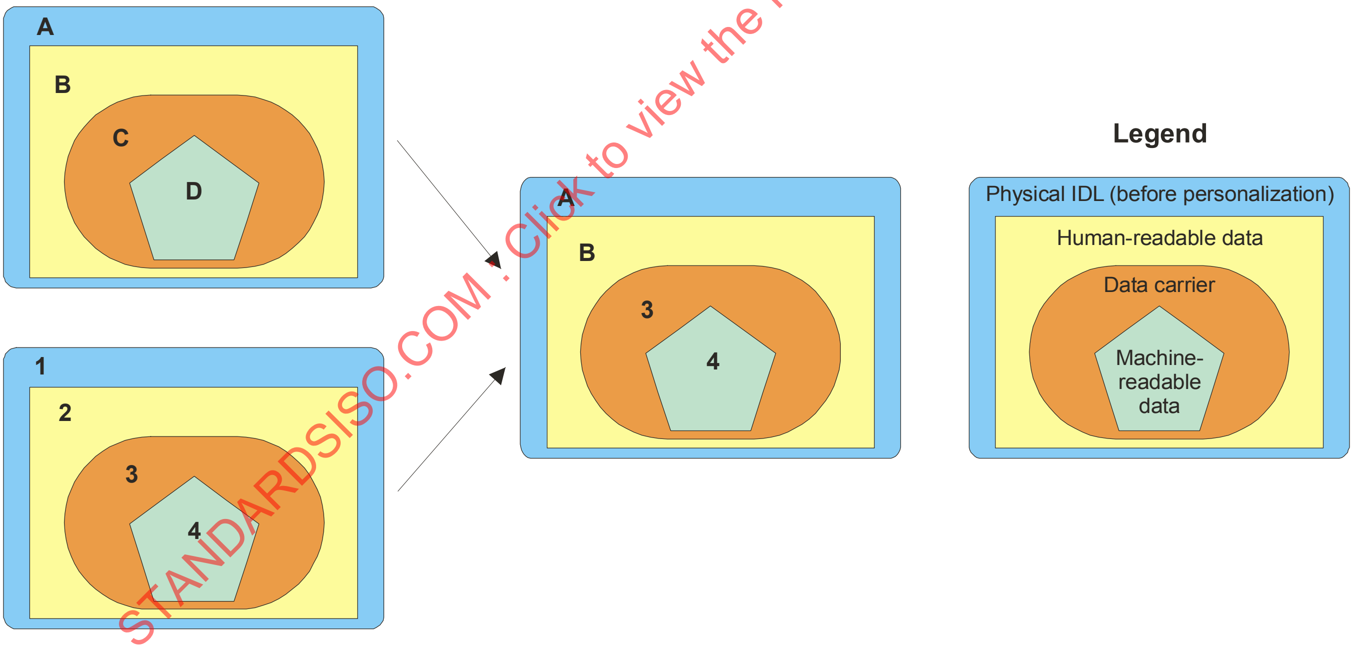


Figure 2 — Data integrity validation: Data carrier exchange or twinning

[1] This guards (amongst others) against an IC "twinning" attack. This type of attack is of particular concern in inspection environments where machine-readable data and human-readable data is not compared (or only cursorily compared by an operator using e.g. a portrait image). Finding a biometrically similar driver is possible by skimming the data of a few thousand IDL PICs.

c) Verify that the physical IDL and the machine-readable data thereon were issued (belong) together. This type of attack can schematically be illustrated as shown in Figure 3.

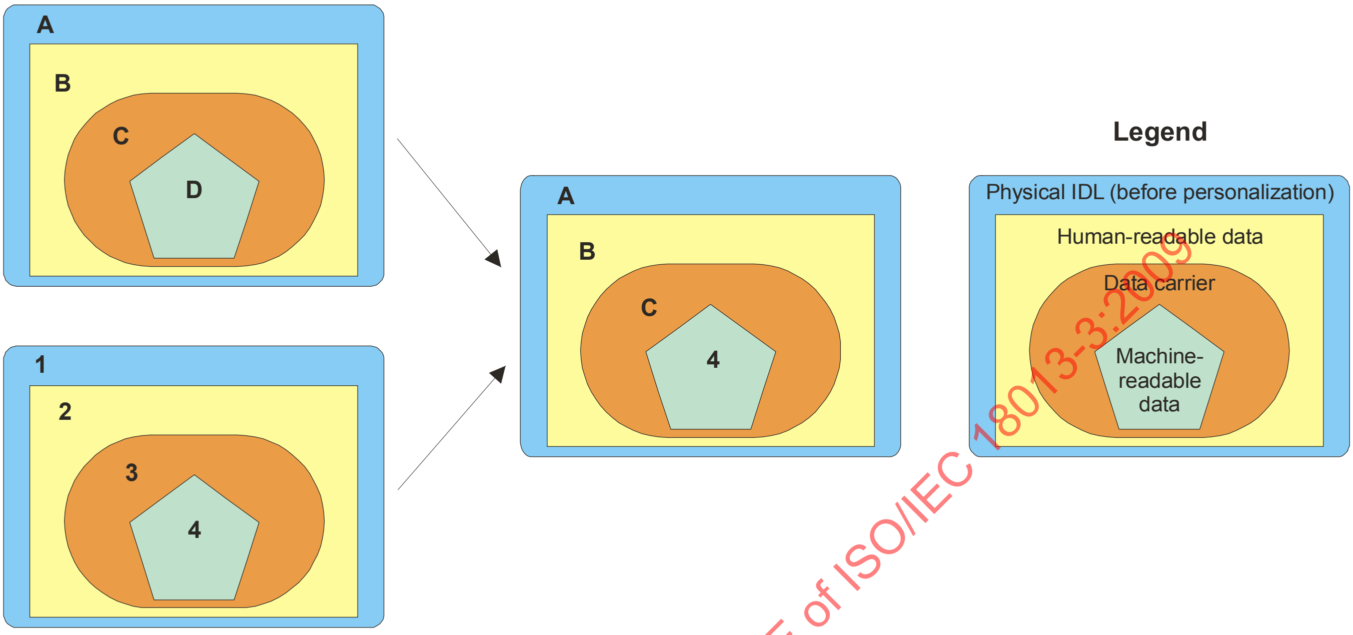


Figure 3 — Data integrity validation: Machine-readable data exchange

d) Validate the integrity of the human readable data (i.e. confirm that the human-readable data has not changed since issuing). This type of attack can schematically be illustrated as shown in Figure 4.

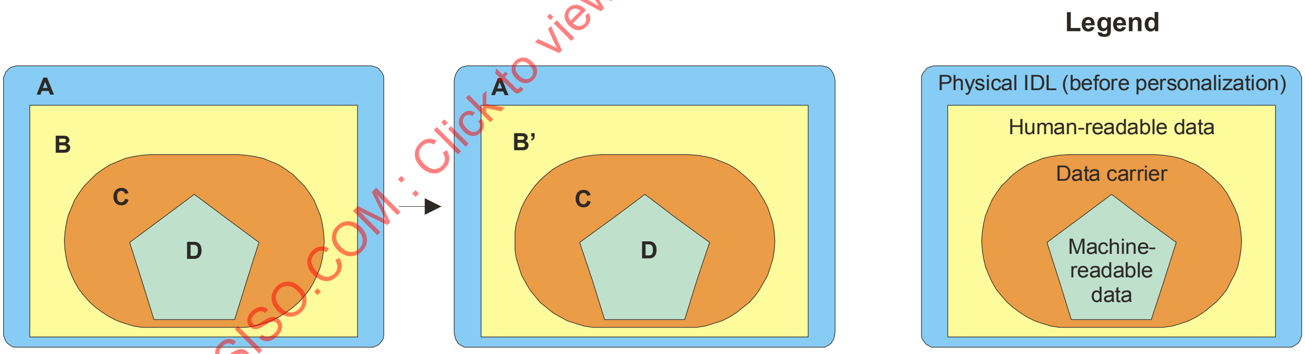


Figure 4 — Data integrity validation: Human-readable data alteration

e) Validate the integrity of the machine-readable data (i.e. confirm that the machine-readable data has not changed since issuing). This can schematically be illustrated as shown in Figure 5.

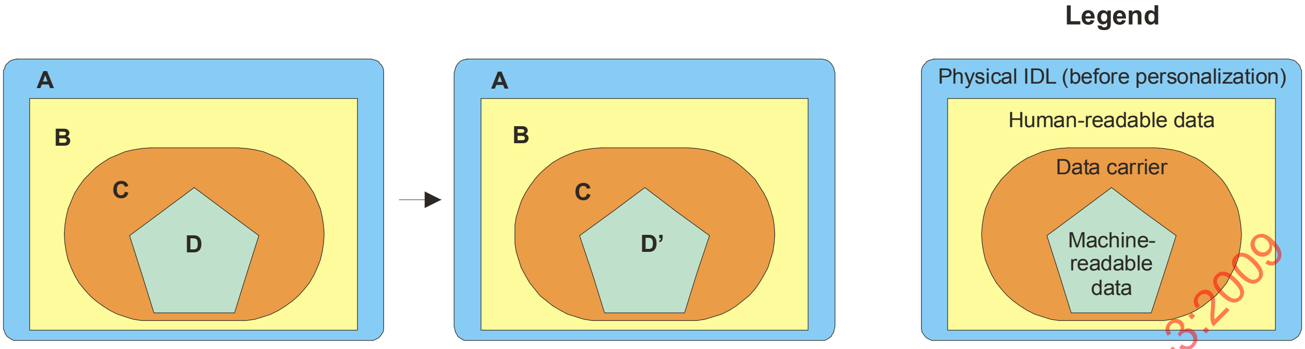


Figure 5 — Data integrity validation: Machine-readable data alteration

7 Mapping of mechanisms to requirements and technologies

Mechanisms that, when used individually, can address specific functional requirements are specified in Clause 8. Mechanisms can also be used together to address a broader range of functional requirements. Not all machine-readable technologies have the same functional requirements, and some mechanisms are applicable only to specific machine-readable technologies. Table 1 specifies the relationships between the mechanisms, machine-readable technologies, and functional requirements.

Table 1 — Applicable mechanisms

Functional requirement	Mechanisms		
	2D bar code, optical memory, magnetic stripe	ICC with contacts	PICC
Prevent skimming	Physical access to the IDL is required to read the machine-readable data	Physical access to the IDL is required to read the machine-readable data	BAP
Prevent unnoticed alteration of communication between a reader and an IDL	N/A	Chip authentication (EAP) in combination with BAP secure messaging	BAP Chip authentication (EAP) in combination with BAP secure messaging
Prevent eavesdropping	N/A	Chip authentication (EAP) in combination with BAP secure messaging	BAP [a] Chip authentication (EAP) in combination with BAP secure messaging
Selectively restrict access to specific optional machine-readable data groups for specific reading authorities	N/A	Terminal authentication (EAP)	Terminal authentication (EAP)
Allow for verification of the origin of an IDL	Step 1: Verify trustworthiness of / obtain a trustworthy asymmetric/public key (via a PKI and a protected distribution communication channel, or via alternative trust building methods) Step 2: Use trusted key to perform passive authentication, followed by use of the non-match alert mechanism, or followed by visual comparison of the human-readable data printed on the document with the authenticated machine-readable data	Step 1: Verify trustworthiness of / obtain a trustworthy asymmetric/public key (via a PKI and a protected distribution communication channel, or via alternative trust building methods) Step 2: Use trusted key to perform passive authentication; if the passive authentication was preceded by BAP the IDL is authentic, if BAP is not implemented passive authentication can be followed by use of the non-match alert mechanism, or followed by visual comparison of the human-readable data printed on the document with the authenticated machine-readable data	Step 1: Verify trustworthiness of / obtain a trustworthy asymmetric/public key (via a PKI and a protected distribution communication channel, or via alternative trust building methods) Step 2: Use trusted key to perform passive authentication; if the passive authentication was preceded by BAP the IDL is authentic, if BAP is not implemented passive authentication can be followed by use of the non-match alert mechanism, or followed by visual comparison of the human-readable data printed on the document with the authenticated machine-readable data
Verify that the IDL (including the machine-readable data) is not a clone of another IDL	2D bar code and magnetic stripe: Cannot be verified through available international standards Optical memory: Include unique card serial number in data protected by passive authentication	Active authentication via challenge response Chip authentication (EAP)	Active authentication via challenge response Chip authentication (EAP)

Functional requirement	Mechanisms		
	2D bar code, optical memory, magnetic stripe	ICC with contacts	PICC
Protect against the exchange of machine readable data carriers between otherwise authentic IDLs	Non-match alert Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data
Verify that the physical IDL and the machine-readable data thereon were issued (belong) together (i.e. that the machine-readable data has not been copied from another IDL)	Non-match alert Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data Active authentication via challenge response[a]	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data Active authentication via challenge response[a]
Validate the integrity of the human readable data (i.e. confirm that the human-readable data has not changed since issuing)	Non-match alert Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data Visual inspection of the security features incorporated into the IDL (see ISO/IEC 18013-1)	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data Visual inspection of the security features incorporated into the IDL (see ISO/IEC 18013-1)	Non-match alert BAP Passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data Visual inspection of the security features incorporated into the IDL (see ISO/IEC 18013-1)
Validate the integrity of the machine-readable data (i.e. confirm that the machine-readable data has not changed since issuing)	Passive authentication	Passive authentication	Passive authentication

NOTE Visual comparison of human-readable information printed on an IDL with machine-readable information obtained from the same IDL is not formally specified as a mechanism in this part of ISO/IEC 18013; it is assumed that such comparison can be implemented by any RA in a manner that meets local needs.

[a] The entropy of the reference string used has to be commensurate with the IA's assessment of the threat.

IA's may selectively implement the mechanisms identified above, subject to any dependencies noted in Clause 8.

8 Mechanisms

8.1 Passive authentication

8.1.1 Purpose

The purpose of passive authentication is to confirm that machine-readable data has not been changed since the IDL was issued.

8.1.2 Applicability

Passive authentication is applicable to all machine-readable technologies.

8.1.3 Description

Passive authentication is implemented by way of a digital signature over specified machine-readable data on the IDL, using a public-private (asymmetric) key pair.

In the case of standard encoding, a separate message digest is calculated for each data group, and included in the machine-readable data. The collection of message digests is then digitally signed (using a private key that is kept secret by the IA), and the digital signature is added to the machine-readable data.

In the case of compact encoding, no message digests are calculated separately. The contents of the data groups present is directly signed (using a private key that is kept secret by the IA), and the digital signature is added to the machine-readable data.

NOTE A message digest has the following properties:

a) It is very small in size compared to the IDL data.

b) The probability of finding any two (different) IDL data sets that lead to the same message digest is negligible. This has the following implications:

i. The probability of finding an IDL data set A that produces the same message digest as a given IDL data set B is negligible.

ii. The probability that a message digest (for the data on an IDL) remains the same upon a change in the data is negligible.

When the IDL is presented to a RA, the RA uses the IA's public key to verify the digital signature. The RA also computes the message digests of each of the data groups that it is interested in, and compares them to the corresponding message digests stored in the machine-readable data. If:

a) the digital signature verifies,

b) the calculated message digests are the same as the message digests stored in the machine-readable data, and

c) the RA is confident that the public key used to verify the digital signature belongs to the claimed IA,

the RA can consider the data groups that it is interested in to be authentic. If the digital signature does not verify, either an incorrect public key was used, or the data on the IDL has been changed. Depending on the digital signature method used, it may be possible to further narrow down the cause of the non-verification.

This part of ISO/IEC 18013 does not prescribe methods to obtain and/or to establish trust in public keys. It is the responsibility of each RA to obtain and/or to establish trust in the public keys used to verify a digital signature on an IDL. Informative methods and approaches to establish such trust are however provided in

Annex A, which describes the principles for a PKI that may be used for public key distribution in the absence of one global certification authority.

This part of ISO/IEC 18013 does not prescribe methods for the generation, administration and safekeeping of key pairs. It is the responsibility of each issuing jurisdiction to ensure that keys are generated, administered and protected as necessary.

8.1.4 Hash function

8.1.4.1 Standard encoding

For standard encoding, IA's shall choose the SHA-1, SHA-224, SHA-256, SHA-384 or the SHA-512 hash function.

NOTE SHA-256 is recommended. SHA-1 remains for compatibility with ICAO Doc 9303-1.

A message digest is calculated separately for each data group present, and stored in the machine-readable data (see 8.1.5.1). The same hash function is used for all data groups. A message digest for a data group is calculated on the concatenation of those data elements present in the data group in the order specified in ISO/IEC 18013-2.

NOTE This approach allows reading authorities to read only those data groups it is interested in.

8.1.4.2 Compact encoding

For compact encoding, IA's shall not calculate separate message digests for each data group. Therefore, no hash function is specified for compact encoding (except as required as part of any digital signature mechanism – see 8.1.5.3).

8.1.5 Signing method

8.1.5.1 Standard encoding

An IDL digital signature is generated over the concatenation of the message digests of the data groups present.

IA's may use either ECDSA or RSA as digital signature methods for standard encoding.

IA's using RSA shall use RFC 4055. RFC 4055 specifies two signature mechanisms, RSASSA-PSS and RSASSA-PKCS1-v1_5. It is recommended to generate signatures according to RSASSA-PSS, but reading authorities shall also be prepared to verify signatures according to RSASSA-PKCS1-v1_5. The minimum size of the modulus, *n*, shall be 1024 bits.

IA's implementing ECDSA shall use ANSI X9.62. The elliptic curve domain parameters used to generate the ECDSA key pair shall be described explicitly in the parameters of the public key, i.e. parameters shall be of type ECParameters (no named curves, no implicit parameters) and shall include the optional cofactor. ECPoints shall be in uncompressed format. The minimum size for the base point order shall be 160 bits.

In addition to EF.COM and the data groups specified in ISO/IEC 18013-2, IA's shall add the SOD to accommodate the hashes of the individual data groups (see 8.1.4.1) and the digital signature of the data on the IDL. The SOD is implemented as a `SignedData` Type, as specified in RFC 3369 (including processing rules). The SOD shall be produced in DER format.

Table 2 — SignedData Type

Value	Type[a]	Comments
SignedData	m	
Version	m	
digestAlgorithms	m	
encapContentInfo	m	
eContentType	m	id-icao-ldsSecurityObject
eContent	m	The encoded contents of an ldsSecurityObject
certificates	o	
Crls	x	
signerInfos	m	
SignerInfo	m	
version	m	
Sid	m	
issuerandSerialNumber	c	It is recommended that IA's support this field over subjectKeyIdentifier.
subjectKeyIdentifier	c	
digestAlgorithm	m	The algorithm identifier of the algorithm used to produce the hash value over encapsulatedContent and SignedAttrs.
signedAttrs	m	IA's may wish to include additional attributes for inclusion in the signature, however these do not have to be processed by a RA except to verify the signature value.
signatureAlgorithm	m	The algorithm identifier of the algorithm used to produce the signature value and any associated parameters.
signature	m	The result of the signature generation process.
unsignedAttrs	o	IA's may wish to use this field, but it is not recommended and reading authorities may choose to ignore them.

[a] m = mandatory (the field shall be present); x = do not use (the field shall not be populated); o = optional (the field may be present); c = choice (the field contents is a choice from alternatives)

ASN.1 sequence

```
LDSSecurityObject { joint-iso-itu-t(2) international-organizations(23) icao(136) mrtd(1)
security(1) ldsSecurityObject(1)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

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

-- Constants

ub-DataGroups INTEGER ::= 16
```

```
-- Object Identifiers
id-icao OBJECT IDENTIFIER ::= {2.23.136}
id-icao-mrtd OBJECT IDENTIFIER ::= {id-icao 1}
id-icao-mrtd-security OBJECT IDENTIFIER ::= {id-icao-mrtd 1}
id-icao-ldsSecurityObject OBJECT IDENTIFIER ::= {id-icao-mrtd-security 1}

-- LDS Security Object

LDSSecurityObjectVersion ::= INTEGER {V0(0)}

DigestAlgorithmIdentifier ::= AlgorithmIdentifier

LDSSecurityObject ::= SEQUENCE {
   version LDSSecurityObjectVersion,
   hashAlgorithm DigestAlgorithmIdentifier,
   dataGroupHashValues SEQUENCE SIZE (2..ub-DataGroups) OF DataGroupHash }

DataGroupHash ::= SEQUENCE {
dataGroupNumber DataGroupNumber,
dataGroupHashValue OCTET STRING }

DataGroupNumber ::= INTEGER {
dataGroup1 (1),
dataGroup2 (2),
dataGroup3 (3),
dataGroup4 (4),
dataGroup5 (5),
dataGroup6 (6),
dataGroup7 (7),
dataGroup8 (8),
dataGroup9 (9),
dataGroup10 (10),
dataGroup11 (11),
dataGroup12 (12),
dataGroup13 (13),
dataGroup14 (14),
dataGroup15 (15),
dataGroup16 (16)}
END
```

NOTE 1 The field dataGroupHashValue contains the calculated hash over the complete contents of the Data Group EF, specified by dataGroupNumber.

NOTE 2 Data Groups 15 and 16 may be defined in future.

8.1.5.2 Standard encoding for optical memory

The DER encoded SOD defined in 8.1.5.1 shall be stored as one data object in tag 15140.

In order to allow identification of clone attempts, IA's may optionally include the card serial number (found at Track 4, Sector 14, in 43-byte sector format; see ISO/IEC 11694-4) in DG3 using tag 1500. With reference to Table D.11 in ISO/IEC 18013-2, tag 1500 shall follow tag 1438 (Optional Data Discriminator) in sequence.

8.1.5.3 Compact encoding

An IDL digital signature is generated over the full data stream from the start of DG1 (including the data group delimiter between DG1 and the header) to the end of DG12 (including the data group delimiter that terminates DG12, but excluding DG.SOD, if present). The digital signature is stored in DG.SOD.

NOTE The header is not included in the information to be signed as the length information in the header pertains to DG11 as well.

IA's shall use ECDSA (as defined in ANSI X9.62) as a signing method for compact encoding, and in order to reduce the storage requirements for the domain parameters, shall use one of the curves specified in FIPS 186-2, Appendix 6.

DG.SOD shall consist of a concatenation of two Type 2 data groups and one Type 1 data group, storing the following information:

a) DG.SOD.1: Digital signature, shall be the DER encoded ASN.1 sequence of two integers, *r* and *s*:

```
SEQUENCE ::= { r INTEGER, s INTEGER }
```

b) DG.SOD.2: Public key; octet string representation of the public point in uncompressed form according to X9.62

c) DG.SOD.3: Curve identifier

DG.SOD shall be added after DG12, as follows:

[header] × [Data Group 1] × [Data Group 2] × [Data Group 3] × [Data Group 4] ×[Data Group 7] × [Data Group 11] × [Data Group 12] × [DG.SOD.1 length] [digital signature] × [DG.SOD.2 length] [public key] × [DG.SOD.3: named curve] ¶

The inclusion of DG.SOD.2 and DG.SOD.3 (as a pair) is optional. Length information is encoded using ASN.1 rules.

NOTE Reading authorities should be able to verify (or obtain) a public key (and domain parameters) from the IA using the data in other data groups (specifically the ISO issuer ID number and document discriminator in DG3 and the licence number and date of issue in DG1).

The curve identifier shall consist of one byte, as shown in Table 3.

Table 3 — Curve identifiers

Curve identifier	Curve name in FIPS 186-2	Curve identifier	Curve name in Annex F
'01'	P-192	'11'	brainpoolP160r1
'02'	P-224	'12'	brainpoolP160t1
'03'	P-256	'13'	brainpoolP192r1
'04'	P-384	'14'	brainpoolP192t1
'05'	P-521	'15'	brainpoolP224r1
'06'	Reserve for future use	'16'	brainpoolP224t1
'07'	Reserve for future use	'17'	brainpoolP256r1
'08'	Reserve for future use	'18'	brainpoolP256t1
		'19'	brainpoolP320r1
		'1A'	brainpoolP320t1
		'1B'	brainpoolP384r1
		'1C'	brainpoolP384t1
		'1D'	brainpoolP512r1
		'1E'	brainpoolP512t1

EXAMPLE Suppose that a compact encoded data string contains the following data groups: DG1, DG2, DG7 and DG.SOD.1. A digital signature is included, but the public key and curve identifier are not included. The sequence of data groups and data group delimiters will be as follows:

[header] × [DG1] × [DG2] × × × [DG7] × × × [DG.SOD.1] ¶

8.2 Active authentication

8.2.1 Purpose

Active authentication uses information stored in a secure area of an SIC to confirm that the SIC and the other machine-readable data were issued together.

8.2.2 Applicability

This mechanism is limited to SICs.

8.2.3 Description

A challenge-response protocol matches the private and public keys of an SIC-individual key pair. The private key is stored in the SIC's secure memory and cannot be copied. The public key is stored as part of the signed data on the SIC. (The challenge-response protocol thus has to be preceded by passive authentication.) The SIC (or more specifically the SIC operating system) can prove knowledge of the SIC-individual private key in a challenge-response protocol. In this protocol the SIC digitally signs a challenge randomly chosen by the IS. The IS is convinced that the SIC is genuine if and only if the returned signature is correct.

NOTE Active authentication using a challenge-response protocol is open to a "traceability" attack. The challenge-response protocol does not place any limitation on the format or content of the challenge. If the challenge includes time, date, location, and a secret key, a log of challenge responses can be used to track IDLs and furthermore to prove presence (due to the inclusion of the secret key). The counterargument is that basic access protection would safeguard against unknown read attempts, and that a driver's location is known in any case in those instances where an IDL is handed over for inspection.

8.2.4 Mechanism

8.2.4.1 General

The SIC-individual public key shall be stored in DG13, an additional data group specifically intended for use with the active authentication mechanism. The format of the structure (`SubjectPublicKeyInfo`) is specified in RFC 3280. `SubjectPublicKeyInfo` shall be produced in DER format as follows:

```
ActiveAuthenticationPublicKeyInfo ::= SubjectPublicKeyInfo

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

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

Allowed algorithm object identifiers and parameters are specified in RFC 3279.

Active authentication shall be performed using the ISO7816 INTERAL AUTHENTICATE command as defined by ISO/IEC 7816-4:2005. The input shall be a terminal-generated nonce (RND.IFD) that shall be 8 bytes in length. The SIC computes a signature which shall be sent back to the IS. The IS shall check the response and verify the signature.

8.2.4.2 Signature generation using RSA

The signature shall be computed according to ISO 9796-2 Digital Signature Scheme 1. M shall consist of M1 and M2, where M1 is an SIC-generated nonce of c-4 bits and M2 is RND.IFD. The SHA-1 hashing algorithm and trailer option 1 shall be used. The result of the signature generation shall be the signature Σ without the recoverable message part M2.

8.2.4.3 Signature generation using ECC

The signature shall be computed according to ANSI X9.62 ECDSA. SHA-1 shall be used as hashing algorithm. The result of the signature generation shall be the DER encoded ASN.1 sequence of two integers, *r* and *s*: `SEQUENCE ::= { r INTEGER, s INTEGER }`.

8.2.4.4 Security mechanism indicator

SICs which implement active authentication may optionally indicate this in the security mechanism indicator (see 9). The object identifier shall be `id-sm-AA`.

```
id-sm-AA OBJECT IDENTIFIER ::= {
   iso(1) standard(0) driving-licence(18013) part-3(3)
   security-mechanisms(2) 3}
```

The parameters are mandatory and shall be of type `param-AA`.

```
param-AA ::= SEQUENCE {
   version INTEGER,
   publicKeyDG INTEGER}
```

The version shall be set to v1(0) for this version of active authentication. The `publicKeyDG` field shall be set to the number of the data group of the active authentication public key, i.e. 13.

8.3 Scanning area identifier

8.3.1 Applicability

This mechanism is used with the non-match alert or BAP mechanisms, which collectively are applicable to SICs, 2D barcode, magnetic stripe and optical memory (see 8.4 and 8.5).

8.3.2 Description

8.3.2.1 General

The primary purpose of this mechanism is to facilitate machine assisted automatic or interactive verification procedures that match machine-readable data to human-readable data. This mechanism is not a solution on its own, but is intended for use as a component of either of the following mechanisms:

a) Non-match alert (see 8.4), which checks if the machine-readable and human-readable data belong together.

b) BAP (see 8.5), which ensures that a PICC cannot be read without physical access to the IDL.

NOTE: The BAP mechanism also performs the function of the non-match alert mechanism, but potentially at a different level of security.

The SAI demarcates the input string. The input string can be entered manually by an operator, or can be read automatically using machine-readable technologies.

The SAI and any machine-readable information demarcated by it shall be adapted for reading in the B900 infrared band defined in ISO 1831:1980. Under such illumination the printing background and any overlays within the SAI will backscatter light in a homogeneous way (i.e. the SAI shall contain no optically variable device, local deviations in properties, security features, or surface gloss exceeding natural transparent overlay gloss, that will interfere with readability). In addition, machine-readable printed information (e.g. bar codes and OCR characters) shall comply with the associated standards. The brightness of the printing background (including the effects of overlays) will not be less than 40% compared to an ideal white surface under the same illumination. Human-readable information inside the SAI shall be rendered such that it can be read without difficulty.

The contrast ratio between character lines and the background shall be a minimum of 4:1. Any reflective layer covering the SAI area shall not further deteriorate this contrast value when illuminated under angles of incidence up to 45 degrees and looked at under angles more than 10 degrees outside of the nominal glance angle.

The presence of the SAI is identified as follows:

a) If PICC access is required and the access conditions are unsatisfied (i.e. BAP is in place), the reader searches for a SAI on the IDL.

b) If access to DG12 is available, BAP is not applicable and the non-match alert mechanism is present. DG12 may indicate the location of the SAI, alternatively, the reader searches for a SAI on the IDL.

Not more than one SAI shall be present on a single IDL.

The reference string (and the input string, when stored in a barcode) shall be encoded in accordance with ISO/IEC 8859-1:1998.

The content of the SAI can follow one of three standards, i.e. based on an existing text field, containing a dedicated text field, or containing a barcode.

8.3.2.2 SAI content based on existing text field

The SAI may be constructed around an existing field on an IDL. In this case the SAI shall consist of a double lined rectangle as illustrated by the example in Figure 6.

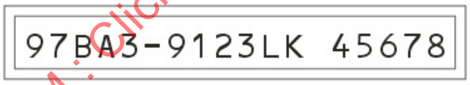


Figure 6 — SAI around an existing field (not to scale)

Each line shall have a thickness of 0.2mm ± 0.1mm, and the distance between the centre of each line shall be 0.6mm ± 0.1mm. The lines shall be black in colour. The clear distance between the inside line and the extremities of the input string shall be at least 1mm.

The input string shall be printed using OCR-B size 1 or a character set with the same symbols and character shapes but using a reduced size of either 25% or 50% reduction of all linear dimensions compared to OCR-B size 1.

To prevent confusion about the input sequence of the characters, text within the rectangle shall be limited to one line. The data element name (if reflected on the IDL) and/or the data field reference code shall not be included within the rectangle.

In order to facilitate manual entry of the input string when needed, the input string may contain only Latin capital letters (hexadecimal range 41 – 5A of ISO/IEC 8859-1:1998[2]), Latin small letters (hexadecimal range 61 – 7A of ISO/IEC 8859-1:1998[2]), N characters, and the S characters shown in Table 4.

NOTE Latin capital letters (hexadecimal range 41 – 5A of ISO/IEC 8859-1:1998) are recommended.

[2] References to ISO/IEC 8859-1:1998 are only for purposes of identifying the characters. Encoding methods are specified elsewhere.

Table 4 — S characters allowed for SAI content based on existing field

S character	Hexadecimal number in ISO/IEC 8859-1:1998[a]
<space>	20
–	2D
<	3C

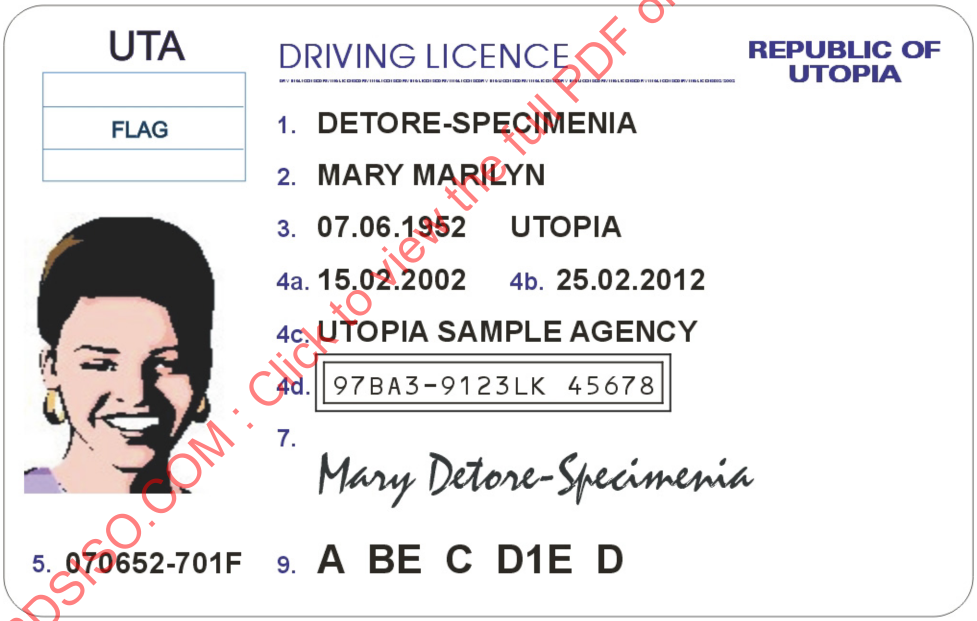
[a] Reference to ISO/IEC 8859-1:1998 is only for the purpose of identifying the characters. Encoding methods are specified separately.

The reference string shall not contain any space characters (i.e. hexadecimal character '20' of ISO/IEC 8859-1:1998). The IS thus shall delete all space characters (hexadecimal code 20 of ISO/IEC 8859-1:1998) from the input string before further processing.

NOTE 1 IA's should carefully consider the level of entropy of existing fields before use within the SAI. Any rules followed in the construction of the field will decrease the entropy.

NOTE 2 IA's considering the use of an SAI based on an existing field should keep in mind that a check digit is not provided for in this case.

EXAMPLE Figure 7



UTA

FLAG

DRIVING LICENCE

REPUBLIC OF UTOPIA

1. DETORE-SPECIMENIA

2. MARY MARILYN

3. 07.06.1952 UTOPIA

4a. 15.02.2002 4b. 25.02.2012

4c. UTOPIA SAMPLE AGENCY

4d. 97BA3-9123LK 45678

7. Mary Detore-Specimenia

5. 070652-701F 9. A BE C D1E D

Figure 7 — Existing data field on portrait side of IDL designated as input string (not to scale)

8.3.2.3 SAI content consisting of a dedicated text field

A dedicated field (i.e. a field designed specifically for this purpose) may be used within a SAI. In this case, the SAI shall consist of a rectangle constructed by a black single line with a thickness of 0.65 mm ± 0.1mm. The clear distance between the line and the extremities of the printed input string shall be at least 1mm.

EXAMPLE Figure 8

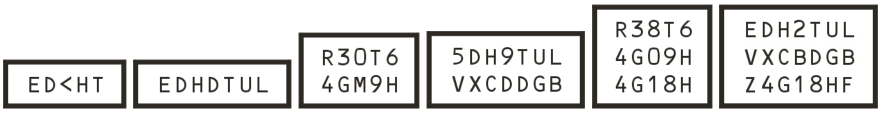


Figure 8 — SAI around a dedicated field (not to scale)

The content of the SAI shall comply with the following requirements:

a) The text shall be limited to the set of ICAO MRZ characters, i.e.: 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z <

b) An odd number of characters per line shall be used.

c) If characters are printed in more than one line, each line shall have the same number of characters.

d) A line shall not have more than seven characters.

e) No more than three lines shall be used.

f) The text shall be printed using OCR-B font at 100% of OCR-B size 1.

g) The middle character of each line shall be reserved for a check digit. Before adding the check digit(s), the input string may have the following lengths:

   i. For one line only, string lengths of 4 or 6 characters are permitted.

   ii. For two lines, string lengths of 8 or 12 characters are permitted.

   iii. For three lines 12 or 18 characters are permitted.

To construct an input string (with check digits) consisting of *i* lines, the input string (before check digits are added) is parsed into *i* parts of equal length. The leftmost part is used to form line 1, the following parts each to form an additional line. Each part is split in the middle into a left and a right subsection. The left subsection forms the leftmost part of the corresponding line, the right subsection forms the rightmost part of the line. An additional character position is inserted in the middle of each line (for the check digit). The value of the check digit of each line will be calculated as explained further below.

For the purpose of check digit calculation each character in the input string (before the check digits are added) represents a numerical value V_C defined by the formula $V_C = E_C$, where E_C is a unique number allotted to each character value, as defined in Table 5.

Table 5 — Character numerical equivalents

Char	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I
E_C	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Char	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	<	
E_C	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	

If line 1 is the only line then the unique check digit will be calculated as follows:

a) A check sum S is calculated as $S = \Sigma V_{Ci}$, where i runs from 0 to n-1 and V_{Ci} indicates the respective value V_C for the character in the i-th position in the input string of length n, counted from right to left, starting with position 0 for the rightmost character.

b) From *S* the corresponding check digit *c* is calculated as *c* = *S* modulo 37; (integer remainder when dividing *S* by 37).

If there are two or three lines, then the check digits c_1 for line one and c_2 for line two will be calculated as follows:

a) A check sum S_V is calculated over the entire input string (before it is split into lines, and before adding the check digits) as $S_V = \Sigma V_{Ci}$, where *i* runs from 0 to *n*-1 and indicates the character in the *i*-th position in the input string of length *n*, counted from right to left, starting with position 0 for the rightmost character.

b) A check sum S_P is calculated over the entire input string (before it is split into lines, and before adding the check digits) as $S_P = \Sigma(V_{Ci}*i)$, where *i* runs from 0 to *n*-1 and *i* indicates the *i*-th position number in the input string of length *n*, counted from right to left, starting with position 0 for the rightmost character, and V_{Ci} indicates the respective value for the character in the *i*-th position.

c) Check digits c_1 and c_2 are calculated as c_1 = (S_V modulo 37) and c_2 = (S_P modulo 37) respectively.

In the case of exactly three lines the inserted character c_3 in the centre of line three will represent the version number and will be set to “1” for this version of the standard. Version numbers are not supported for less than three lines.

When printed on an IDL, lines are numbered from top to bottom, as shown in Figure 9.

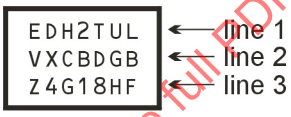


Figure 9 — Line configuration (not to scale)

EXAMPLE Input string for Figure 9 is EDHTULVXCDGBZ4G8HF

Input string	E	D	H	T	U	L	V	X	C	D	G	B	Z	4	G	8	H	F
E_C	14	13	17	29	30	21	31	33	12	13	16	11	35	4	16	8	17	15
i	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$V_{Ci}*i$	238	208	255	406	390	252	341	330	108	104	112	66	175	16	48	16	17	0

S_V = 335

S_P = 3082

Check digit	Numeric value	Code value
c1	2	2
c2	11	B

c3 = 1

The input string submitted as input to the non-match alert or BAP mechanisms is the input string inclusive of the check digit, i.e. the concatenation s1 + s2 + s3 where si is the string in line i of the corresponding input string, including the respective check digit and read from the left to the right, with i ranging from 1 to a maximum of 3.

NOTE Mathematically, the check digit scheme allows identifying any single digit reading error in the case of a one-line input string and to correct any single digit reading error as well as identifying any double error in the case of a two- or three-line input string. Less than 3% of any other statistical reading error may pass inadvertently. This is only true if adequate measures are taken on the side of the reading equipment to make use of the information offered by the check digit scheme.

It is recommended that the minimum entropy of the input string be 40 bits or more (before addition of the check digits). The use of random data is recommended.

NOTE IA's may consider imposing restrictions on allowable content of input strings to meet local requirements, e.g. to address the "nasty word" problem, or to limit the value range of practically existing check digits (e.g. numerical characters only). Methods exist that allow enforcement of such restrictions without violation of the rules set out above. The use of such methods is left to the discretion of the IA's. Attention is drawn to the fact that such restrictions will reduce the entropy of the input string.

EXAMPLE Figure 10

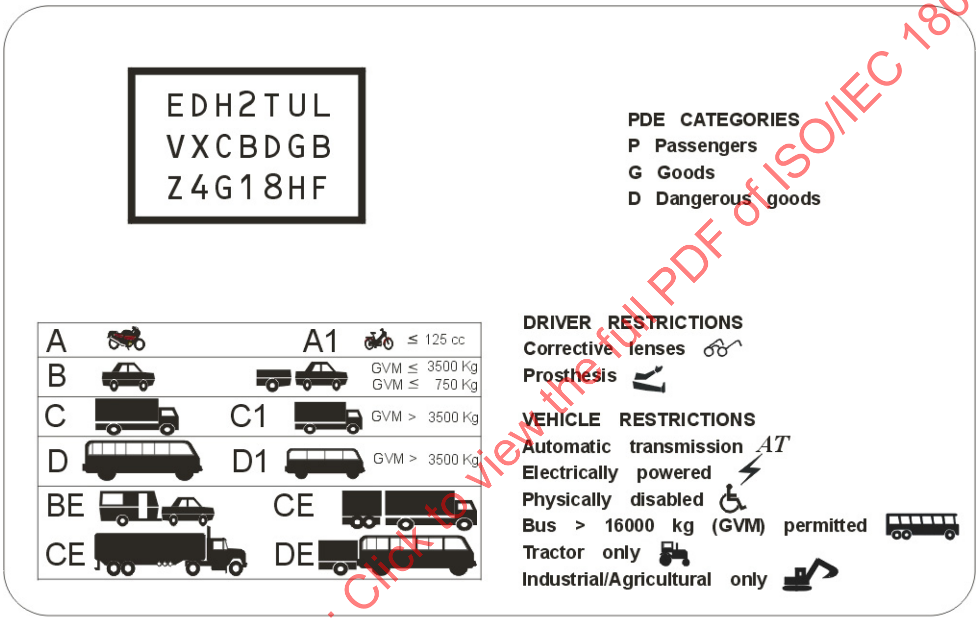


Figure 10 – Dedicated data field on non-portrait side of IDL designated as input string (not to scale)

8.3.2.4 SAI content consisting of a barcode

A barcode may be used within a SAI as the primary means of input. The SAI's corners shall be clearly shown. Each corner indicator shall be constructed of two perpendicular lines joined at the end points, with each line having a thickness of 0.5mm ±0.3mm, and a length of 4mm ±2mm.

EXAMPLE Figure 11

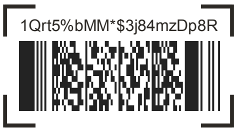


Figure 11 — SAI around a dedicated field (not to scale)

The input string may contain any A, N or S characters, and is not limited in length. However, it is recommended that the input string be limited to the characters specified in 8.3.2.3 to facilitate manual entry when required. The barcode may be accompanied by a human-readable version of the input string (in accordance with ISO/IEC 8859-1:1998), printed in a font not smaller than 6pt.

EXAMPLE Figure 12

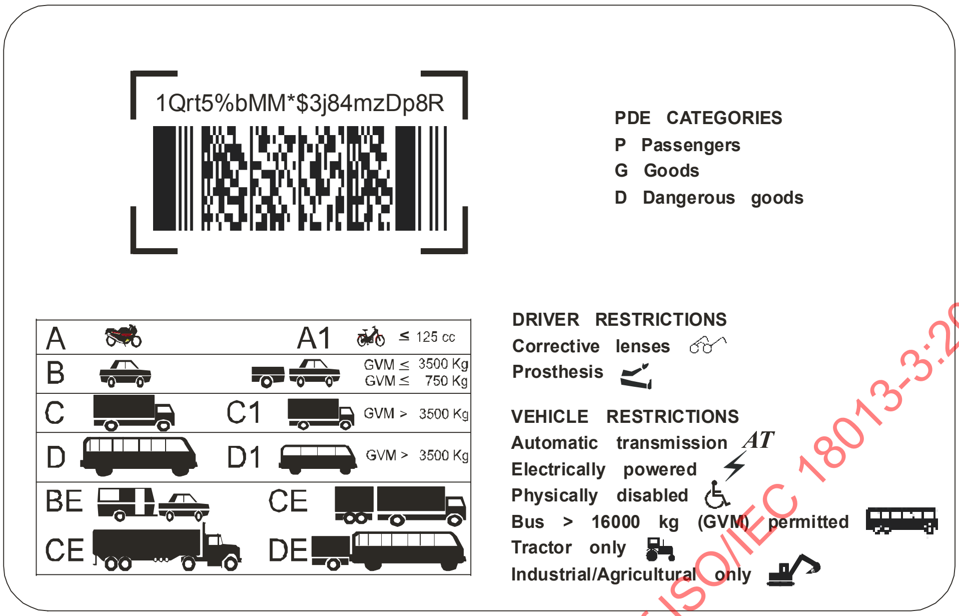


Figure 12 – Barcode on non-portrait side of IDL designated as input string (not to scale)

8.4 Non-match alert

8.4.1 Purpose

This mechanism allows detection of any differences between the machine-readable information and (some of) the human-readable information (printed on an IDL). This mechanism may be of value in the presence of machine-readable technologies with data content not accessible to visual inspection and on data carriers or physical support that may be transferred completely from one IDL to another without strong visual evidence of alteration.

8.4.2 Applicability

This mechanism is applicable to SICs, 2D barcode, magnetic stripe and optical memory.

8.4.3 Description

Conceptually, the reference string is verified via a digital signature, after which the reference string is compared to the input string. If the comparison is not successful, it means that the machine-readable data and the (human-readable) printed data are not consistent.

NOTE 1 It is assumed that the RA trusts the public key used to verify the digital signature.

NOTE 2 If the comparison is successful, the RA may conclude that the machine-readable data and the physical document are consistent if it can be assumed that alteration to any part of the printed data (i.e. not just the area that contains the input string) would be visually apparent.

The reference string is not separately verified with a dedicated digital signature, but is verified during passive authentication (the digital signature used for passive authentication already covers the reference string).

NOTE The function provided by the non-match alert mechanism can also be achieved by performing passive authentication followed by a visual comparison of the human-readable data printed on the document with the authenticated machine-readable data.

8.4.4 Mechanism

8.4.4.1 General

The manner in which the non-match alert mechanism is implemented is specified in DG12, using the parameters and values as shown in Table 6.

Table 6 — Non-match alert parameters

Name, Fixed (F) or Variable (V), Mandatory (M) or Optional (O)	Field format/ length/type	Example
SAI_referencestring , V, M	Byte 1: '00' if the input string follows '01' if a reference to where the input string can be obtained follows Subsequent bytes: If byte 1 = '00', the input string follows from byte 2 (inclusive). The input string is encoded in accordance with ISO/IEC 8859-1:1998. If byte 1 = '01', the reference to the field that contains the input string is constructed as aabb where aa is the data group and bb is the sequence number of the referenced data element, with aabb encoded as unsigned BCD	An input string of ABC4DEF contained in the SAI_referencestring field will be coded as '00 41 42 43 34 44 45 46', where '41 42 43 34 44 45 46' is the encoded form of ABC4DEF. If the licence number is used as the input string, this will be coded as '01 01 08'. If the 16th data element of Data Group 12 is used as the input parameter, the input string will be coded as '01 12 16'
SAI_inputmethod, V, O	Byte 1: SAI standard and input method. The four most significant bits (upper nibble) of byte 1 can take on any of the following values: '0x' if the input string is based on an existing field '1x' if the input string is based on a dedicated field '2x' if the input string is stored in a barcode The four least significant bits (lower nibble) of byte 1 denotes the input method, and can take on any of the following values: 'x0' if the input string is intended for manual input 'x1' if the input string is intended for OCR interpretation 'x2' if the input string is stored as a barcode. Byte 2: Barcode standard. If the first byte is of the form 'x2', byte 2 is mandatory, taking on any of the following values: '00' for PDF417 '01' for Code 39 (ISO/IEC 16388) '02' for Code 128 (ISO/IEC 15417) '03' for data matrix (ISO/IEC 16022)	If the licence number is used as the input string (i.e. a SAI is constructed around the existing licence number field on the IDL), the value of SAI_inputmethod will be '00'. If, in addition, the input string is printed in OCR-B font, the input string will be '00 01', or alternatively '00 01 00'. If, in addition, the SAI is located on the portrait side of the IDL, with the top left corner of the SAI at 29mm from the left edge of the card and 24mm from the bottom edge of the card, and the right bottom corner of the SAI at 59mm from the left edge of the card and 14mm from the bottom edge of the card, the input string will be '00 01 00 00 29 24 59 14'

Name, Fixed (F) or Variable (V), Mandatory (M) or Optional (O)	Field format/ length/type	Example
	'FE' for other barcode standards not provided for above 'FF' no barcode Byte 2 is also mandatory if Bytes 3 to 7 are present. Bytes 3 to 7: Position of the SAI, expressed as 'aa bb cc dd ee', where 'aa' is the side of the card on which the SAI appears ('00' for portrait side, and '01' for non-portrait side), 'bb cc' is the top left corner of the SAI (where 'bb' is the distance from the left edge of the IDL and 'cc' is the distance from the bottom edge of the card), and 'dd ee' is the bottom right corner of the SAI (where 'dd' is the distance from the left edge of the IDL and 'ee' is the distance from the bottom edge of the card), with all distances measured in millimetres, and encoded as BCD. The bytes are progressively mandatory. I.e., SAI_inputmethod can consist only of byte 1, or only of bytes 1 and 2, or of bytes 1 to 7.	

8.4.4.2 Compact encoding

When used with compact encoding, DG12 shall be constructed as a Type 1 data group as follows:

… × [SAI_referencestring] ÷ [SAI_inputmethod] × …

8.4.4.3 Standard encoding

When used with standard encoding, the parameters shall be stored in DG12 as shown in Table 7.

Table 7 — Non-match alert parameters for standard encoding

Tag	Length	Value
‘82’	X	SAI_referencestring as defined in Table 6.
‘81’	X	SAI_inputmethod as defined in Table 6.

IA's may optionally indicate the presence of the non-match alert mechanism in the EF.COM file using the security mechanism indicator as specified in 9.

The object identifier for non-match alert shall be `id-sm-NMA`.

```
id-sm-NMA OBJECT IDENTIFIER ::= {
  iso(1) standard(0) driving-licence(18013) part-3(3)
security-mechanisms(2) 4
}
```

The parameters are optional and shall be of type `param-NMA`.

```
param-NMA ::= SEQUENCE {
   version INTEGER, SAI_inputmethod OCTET STRING
}
```

The `version` field shall be set to v1(0) for this version of the non-match alert mechanism.

The `SAI_inputmethod` field shall be set to the corresponding value in DG12. If the `SAI_inputmethod` field is not present in DG12, the field shall also not be included in EF.COM.

8.4.4.4 Standard encoding for optical memory

When used with standard encoding for optical memory, the parameters shall be stored in DG12 as shown in Table 8.

Table 8 — Non-match alert parameters for standard encoding

Tag	Value
15200	SAI_referencestring as defined in Table 6.
15201	SAI_inputmethod as defined in Table 6.

8.5 Basic access protection

8.5.1 Purpose

BAP confirms that an IS has physical access to a PICC before the IS is allowed access to the data stored on the PICC. In addition, BAP ensures that communication between the IS and the PICC (once access is authorized) is protected.

NOTE Although BAP was designed for PICCs, it can also be used to satisfy a variety of functional requirements for ICs with contacts as listed in Table 1.

8.5.2 Applicability

This mechanism is limited to PICCs and ICs with contacts.

8.5.3 Description

For a SIC that is accessed via a contactless interface and protected by the BAP mechanism shall deny access to its content by the contactless interface unless the IS can prove that it is authorized to access the SIC. This proof shall be given in a challenge-response protocol, where the IS proves knowledge of the PICC-individual document basic access keys (K_{ENC} and K_{MAC}) that are derived from the input string.

After the IS has been authenticated successfully, the SIC shall enforce encryption and message authentication of the communication channel between the IS and the SIC by Secure Messaging techniques.

8.5.4 Mechanism

The document basic access keys are stored in the internal elementary file (see ISO/IEC 7816-4:2005).

A SIC that supports BAP shall respond to unauthenticated read attempts (including selection of files in the logical data structure) with ‘Security status not satisfied’ (0x6982). The presence of BAP thus is determined by an IS from the response to read attempts (see below) and the subsequent success in locating the SAI.

BAP is specified in Annex B. The following shall be used in the application of Annex B:

a) The reference string shall be used as the document keying material (K_{doc}).

b) The IS can automatically read the input string, or can allow an operator to manually enter the input string (if present) into the IS.

c) The first byte of the input string shall identify the BAP configuration used.

NOTE BAP is intended mainly as an anti-skimming mechanism. However, provided that the entropy of the reference string is commensurate with the IA's assessment of the threat, this mechanism can also serve as protection against eavesdropping.

IA's may optionally add the BAP logo to an IDL to indicate that the SAI contains a BAP input string. Figure 13 shows the BAP logo. The minimum dimensions of the A-dimension of the logo shall be 5mm, and the B-dimension shall be scaled proportionally in a ratio of 5:3. It is recommended that the BAP logo be placed in close proximity to the associated SAI.

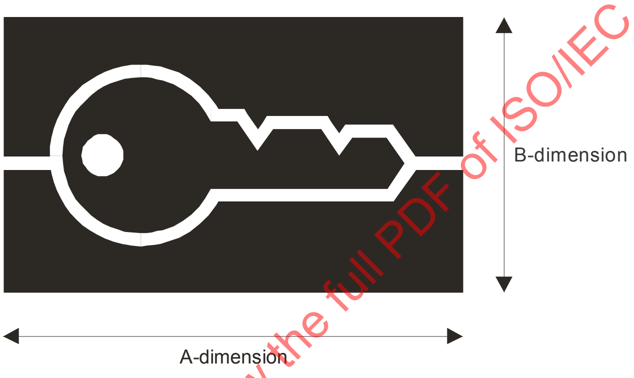


Figure 13 — Basic access protection logo (not to scale)

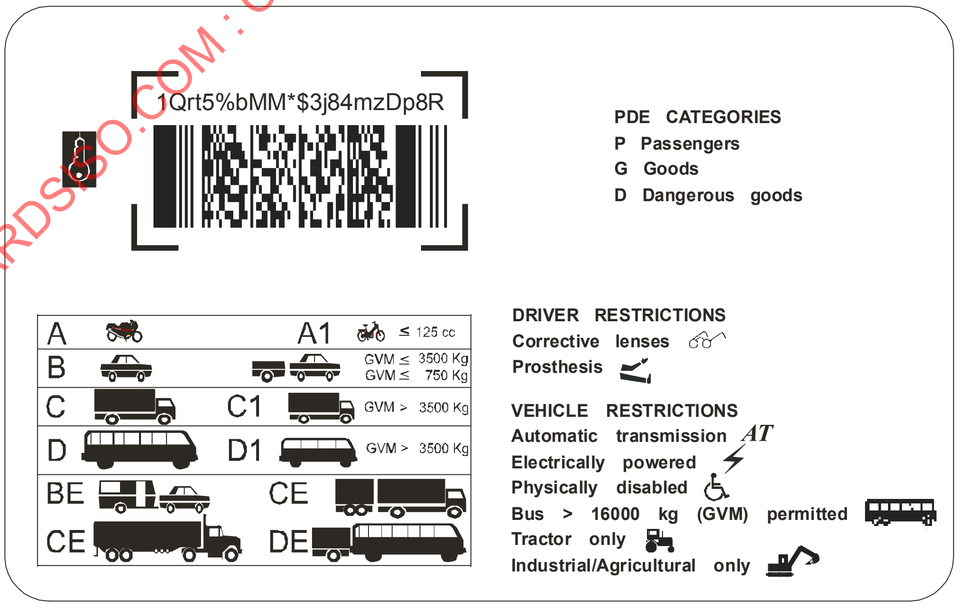


Figure 14 — Basic access protection logo on IDL (not to scale)

EXAMPLE Suppose the following barcode is printed on the IDL:

This barcode encodes the string “1462483345434115654434034118361284817041”, whose hexadecimal rendering is ‘31 34 36 32 34 38 33 33 34 35 34 33 34 31 31 35 36 35 34 34 33 34 30 33 34 31 31 38 33 36 31 32 38 34 38 31 37 30 34 31’

The first byte (‘31’) of the input string indicates BAP configuration 1. The entire input string is used as K_{doc}. The resulting derived document access keys are:

K_{enc}:	‘E4 80 0E 99 54 FF 39 EB F6 09 15 FD 46 C4 3F DD’
K_{mac}:	‘E7 EA 15 4F ED 39 38 77 A1 12 9D CB 7A 54 93 50’

The barcode in this example contains 40 numeric digits. As the first byte conveys predictable information, only the following 39 bytes may contribute entropy. Assuming they were generated randomly, the resulting K_{doc} contains $\log_2(10^{39}) \approx 129$ bits of entropy.

8.6 Extended access protection

8.6.1 Purpose

EAP consists of:

a) Chip authentication, which provides for authentication of the SIC and strong secure messaging.

b) Terminal authentication, which provides for conditional authenticated access to data groups.

8.6.2 Applicability

This mechanism is applicable only to SICs.

8.6.3 Description and mechanism

Extended access protection is specified in Annex C. The following shall be used in the application of Annex C:

a) The SIC's key agreement public key shall be stored in DG14, formatted in accordance with C.3.4.

b) The driving licence number, as it appears in DG1, shall be used as the SIC identifier.

c) Strong secure messaging (established using chip authentication as described in C.3) shall be active before terminal authentication can take place.

d) DG14 shall be accessible without terminal authentication.

NOTE Both BAP and EAP use the card commands GET CHALLENGE (INS = '84') and MUTUAL/EXTERNAL AUTHENTICATE (INS = '82'). Since a driving licence may implement both protocols, the two cases need to be distinguished in such a case. As per c) above, strong secure messaging that has been established through chip authentication shall be active before EAP Terminal Authentication can take place. Thus these two commands are used for EAP only if secure messaging is active (CLA = 0x0C), otherwise they are used for BAP.

9 Security mechanism indicator

The security mechanism(s) deployed on an IDL with an SIC can be identified[3] to reading authorities by the addition of tag '86' to EF.COM. Tag '86' identifies the data groups subject to each mechanism used. Tag '86' shall have the following DER-encoded ASN.1 TLV structure:

```
SecurityMechanisms ::= SET OF SecurityMechanism

SecurityMechanism ::= SEQUENCE {
  mechanism MechanismIdentifier,
  datagroups SET OF INTEGER}

MechanismIdentifier ::= SEQUENCE {
  mechanism OBJECT IDENTIFIER,
  parameters ANY DEFINED BY mechanism OPTIONAL}
```

EXAMPLE Assume an implementation using LDS version 1.0 having the following data content – mandatory data (DG 1) and optional licence holder data (DG 2). The optional licence holder data is protected using EAP (see Annex C) with the current trust root set to the example in C.A.1, no alternate trust root, and BAP configuration 1 (see Annex B) as secure messaging configuration. EF.COM would be encoded as follows (in order to improve clarity, tags are printed in *RED ITALICS*; lengths are printed in UNDERLINED BLUE and values are printed in BLACK):

```
    '60' '57'
        '5F 01' '02'
            '01 00'
        '5C' '02'
            '61 6B'
        '86' '4C'
            '31 4A 30 48 30 41 06 07 28 81 8C 5D 03 02 02 30
            36 02 01 00 02 01 0E 06 08 28 81 8C 5D 03 02 01
            01 04 11 10 C0 0D 1D 69 61 15 35 70 67 0B 2C 3A
            EE 64 FF 36 04 11 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 31 03 02 01 02'
```

where '31 4A 30 46 … 01 02' is the following DER-encoded ASN.1 structure:

```
    SET
        SEQUENCE
            SEQUENCE
                OBJECT IDENTIFIER: 1.0.18013.3.2.2
                SEQUENCE
                    INTEGER: 00
                    INTEGER: 0E
                    OBJECT IDENTIFIER: 1.0.18013.3.2.1.1
                    OCTET STRING: 10C00D1D6961153570670B2C3AEE64FF36
                    OCTET STRING: 00000000000000000000000000000000
            SET
                INTEGER: 02
```

10 SIC LDS

In addition to the data groups identified in Figure C.1 in ISO/IEC 18013-2, this part of ISO/IEC 18013 defines the data groups shown in Table 9.

[3] Use of the security mechanism indicator is optional unless a mechanism mandates the security mechanism to be used.

Table 9 — Assignment of file identifiers and Data Group Tags

Elementary file	Name	Short EF Identifier	EFID	Tag
EF.SOD	Document security object	'1D'	'001D'	'77'
EF.DG12	Non-match alert	'0C'	'000C'	'71'
EF.DG13	Active authentication	'0D'	'000D'	'6F'
EF.DG14	Extended access protection	'0E'	000E	'6E'

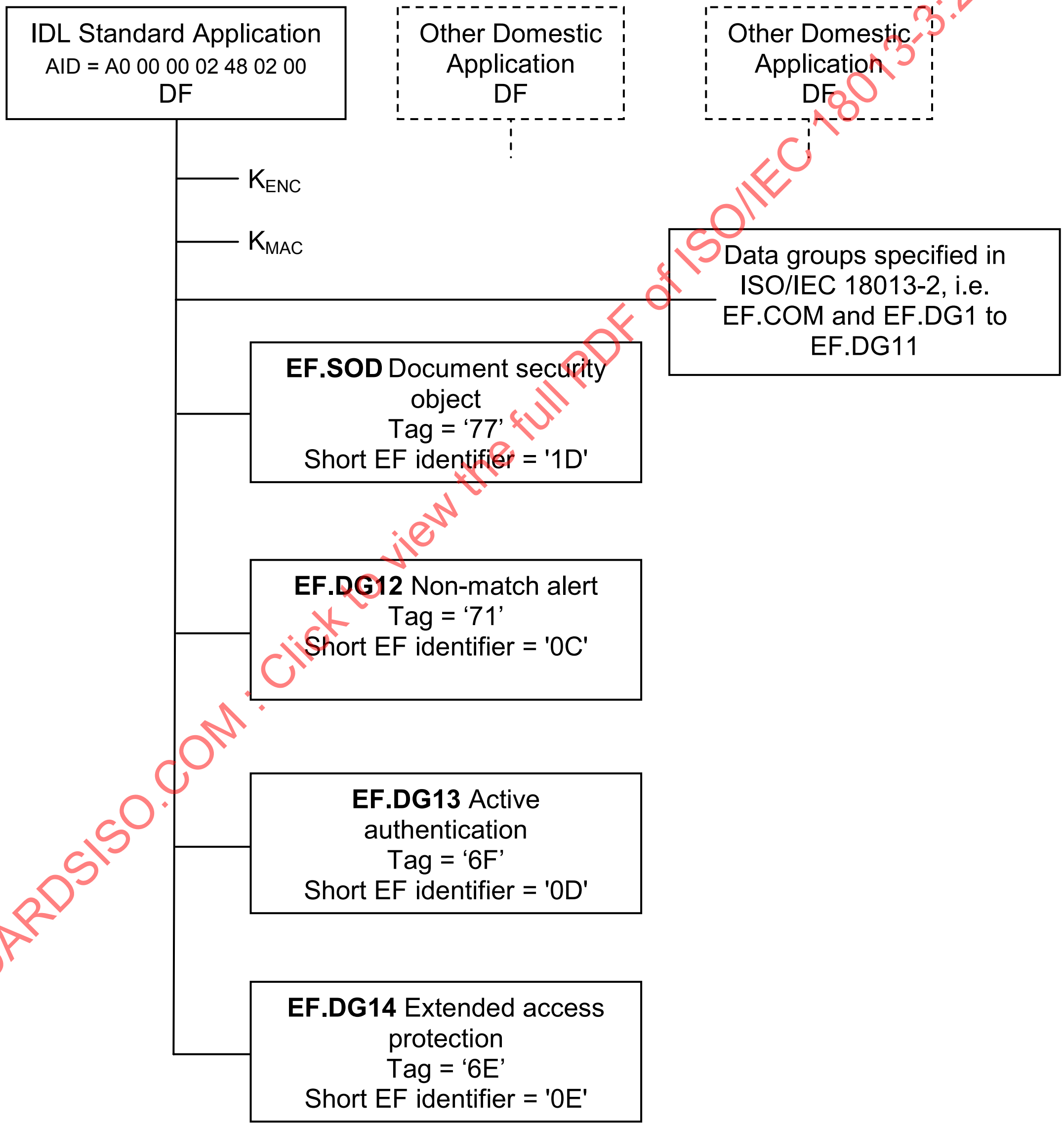

IDL Standard Application
AID = A0 00 00 02 48 02 00
DF
Other Domestic Application DF
Other Domestic Application DF
K_{ENC}
K_{MAC}
Data groups specified in ISO/IEC 18013-2, i.e. EF.COM and EF.DG1 to EF.DG11
EF.SOD Document security object
Tag = '77'
Short EF identifier = '1D'
EF.DG12 Non-match alert
Tag = '71'
Short EF identifier = '0C'
EF.DG13 Active authentication
Tag = '6F'
Short EF identifier = '0D'
EF.DG14 Extended access protection
Tag = '6E'
Short EF identifier = '0E'

Figure 15 — Data groups

10.1 EF.SOD – Document security object (short EF identifier = ‘1D’, Tag = '77')

EF.SOD is defined in 8.1.5.1.

10.2 EF.DG12 Non-match alert (short EF identifier= '0C', Tag = ‘71’)

DG12 is defined in 8.4.4.

10.3 EF.DG13 Active authentication (short EF identifier = '0D', Tag = ‘6F’)

DG13 is defined in 8.2.4.1.

10.4 EF.DG14 Extended access protection (short EF identifier = '0E', Tag = ‘6E’)

DG14 is defined in C.3.4.

Annex A
(informative)

Public key infrastructure (PKI)

A.1 Introduction

This annex suggests a structure for a public key infrastructure specifically in respect of IDLs that are used internationally. Such a use environment poses unique challenges, primarily the infeasibility of any one issuing jurisdiction to conclude, implement and maintain key sharing agreements and infrastructures with all other issuing jurisdictions (given the assumption that a "super certification authority" that controls and issues keys globally is not available).

The concepts in this annex were developed from first principles, and explore the functional and logical aspects of a PKI. Standardisation of technical aspects relating to implementation (e.g. certificate formats) are outside the scope of this annex, although some issues that would need standardisation are pointed out. Consequently, this annex describes the mechanisms to establish a "trust model" more than it describes a traditional PKI.

The trust model described in this annex is based on a "trust by proxy" principle, i.e. A trusts C because A trusts B and B trusts C. Consequently, the ultimate responsibility to establish trust in a public key remains with the RA (also see Annex B dealing with trust establishment). The trust model should not be seen as infallible confirmation of the origin and integrity of a pubic key.

A.2 PKI Design principles

In addition to the principle stated in A.1, the trust model was designed to comply with the following principles:

a) No central CA is available.

b) The trust model should exploit existing trust relationships.

c) The rules for setting up and maintaining the trust network should not require any one entity to manage the trust network, but should allow for the trust network to essentially manage itself.

d) Use a two-level key approach[4], consisting of a root key-pair, and a document key pair. The document key pair is used to sign[5] and verify an IDL, and the root key pair is used to sign and verify the document key and other information as described below. Public root keys are to be exchanged by out-of-band means.

NOTE 1 Given the autonomy of each country, and the political sensitivities that exist between some, a trust model that uses one global certification authority is considered infeasible.

NOTE 2 Existing trust relationships are an ideal starting point to build a trust network. For example, the American Association of Motor Vehicle Administrators (AAMVA) already has trust relationships established with most IA's in North America. In Southern Africa, the Southern Africa Development Community (SADC) has relationships set up with all the countries in Southern Africa, with a similar function performed by the Economic Community of West African States (ECOWAS) in West Africa, and the Common Market for Eastern and Southern Africa (COMESA) in Eastern and Southern Africa. In Europe, the European Commission has relationships with numerous IA's.

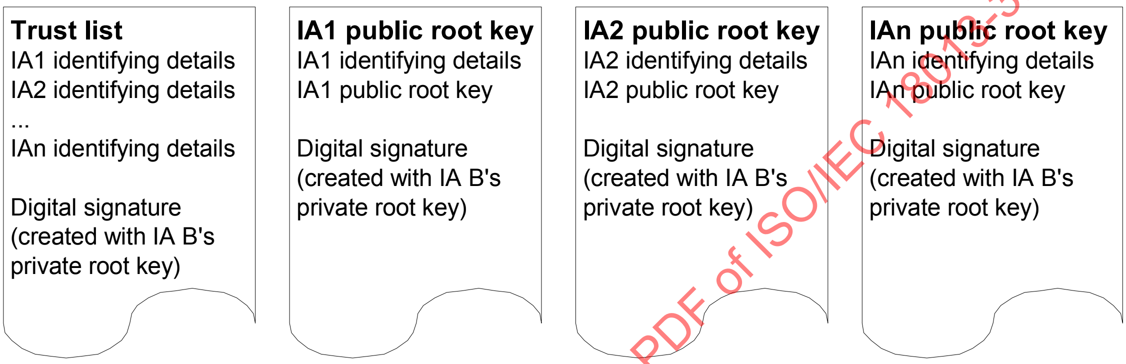
[4] A one-level key approach is possible, but less appropriate given that keys used to sign IDLs are replaced periodically.

[5] In this context, "sign" means that a private key is used to create a digital signature for a certificate that contains a public key and/or other information.

A.3 General description[6]

Each IA generates two sets of key pairs, a root key pair[7], and a document key pair[8]. The IA signs its own public root key using its private root key[9]. The public root key is distributed out-of-band only[10]. The public document key is also signed with the private root key. For standard encoding, the public document key may optionally be distributed with the IDL.

As noted above, it is infeasible to expect each IA's public root key to be distributed (out of band) to every other IA. Consequently, if IA A trusts IA B, then IA A would also like to know who else IA B trusts. IA B thus needs to publish[11] a list (signed with IA B's private root key) of the IAs (including IA A, if applicable) that it trusts. IA B also signs the public root key of each IA in IA B's trust list[12], and publishes all the signed public root keys. B thus publishes the documents (or certificates)[13] shown in Figure A.1.



Trust list
IA1 identifying details
IA2 identifying details
...
IAn identifying details

Digital signature
(created with IA B's
private root key)

IA1 public root key
IA1 identifying details
IA1 public root key

Digital signature
(created with IA B's
private root key)

IA2 public root key
IA2 identifying details
IA2 public root key

Digital signature
(created with IA B's
private root key)

IAn public root key
IAn identifying details
IAn public root key

Digital signature
(created with IA B's
private root key)

Figure A.1 — Certificates published by an IA

If every IA publishes the same items that IA B publishes (i.e. a signed trust list, and signed public root keys of each authority on the trust list), it enables IA A to construct a diagram of the complete trust network. For example, if IA B trusts IA C, IA A uses IA C's public root key (signed by IA B), to verify IA C's trust list. If IA C trusts IA D, IA A uses IA C's public root key to verify IA D's public root key, and then verifies IA D's trust list, and so on. IA A thus compiles a trust network diagram[14] which can be used to verify any IDL presented.

The above allows IA A to verify the public root key of each IA (in the trust network). The public root key then is used to verify the public document keys published by each IA.

[6] The trust model is somewhat similar to the user-centric trust model employed by PGP, and as discussed by C Adams and S Lloyd (see bibliography).

[7] The root key as used in this document is similar to the Country Signing CA Key in the ICAO PKI Technical Report (see bibliography).

[8] The document key as used in this document is similar to the Document Signer Key in the ICAO PKI Technical Report (see bibliography).

[9] This means that the IA certifies that the public root key is associated with the IA. In a more traditional environment, a CA would sign e.g. IA A's public root key, thus certifying the association between the public root key and IA A. If IA B trusts the CA, IA B would then also trust the association between IA A and its public root key. If IA A self-signs its public root key, and IA B trusts IA A, a CA becomes superfluous (at least for the purpose of associating IA A with its public root key).

[10] ICAO's PKI Technical Report (see bibliography) mentions the out-of-band *confirmation* of a root public key, thus inferring that the actual distribution of such keys may take place separately from the out-of-band confirmation, possibly using "in-band" communication.

[11] See A.4.1 for a discussion of the term "publish".

[12] IA B would presumably (but not necessarily) only directly trust those IAs for which it has received the public root key in an out-of-band fashion.

[13] In addition, IA B would also publish information on IA B's public keys used to sign IDLs. This however does not directly pertain to the trust network, and is only discussed later.

[14] The trust network diagram includes all the public keys, lists and other information required to verify an IDL in an off-line manner. Typically, the trust network diagram is updated periodically (e.g. daily), and used as reference for all internal verification actions.

Public root keys distributed between IAs in an out-of-band fashion essentially become trust anchors, and should be stored, protected and used in an appropriately secure manner.[15]

It is anticipated that the global trust network will eventually consist of a collection of "local" trust networks, with the different local trust networks connected by a limited number of trust "links". For example, in the Southern Africa Development Community (SADC), South Africa may act as an aggregator for the SADC countries. That is, each SADC country exchanges public root keys (out-of-band) only with South Africa, and vice versa. South Africa then signs each public root key, and publishes same along with the list of all SADC countries. In the European Union (EU), the European Parliament may decide to act as a central trust broker, or PIA. That is, the EU obtains the public root key for each IA in the EU, signs it with the EU private root key, and publishes same. The American Association of Motor Vehicle Administrators (AAMVA) may fulfil a similar function in North America. Figure A.2 illustrates an example of a trust network.

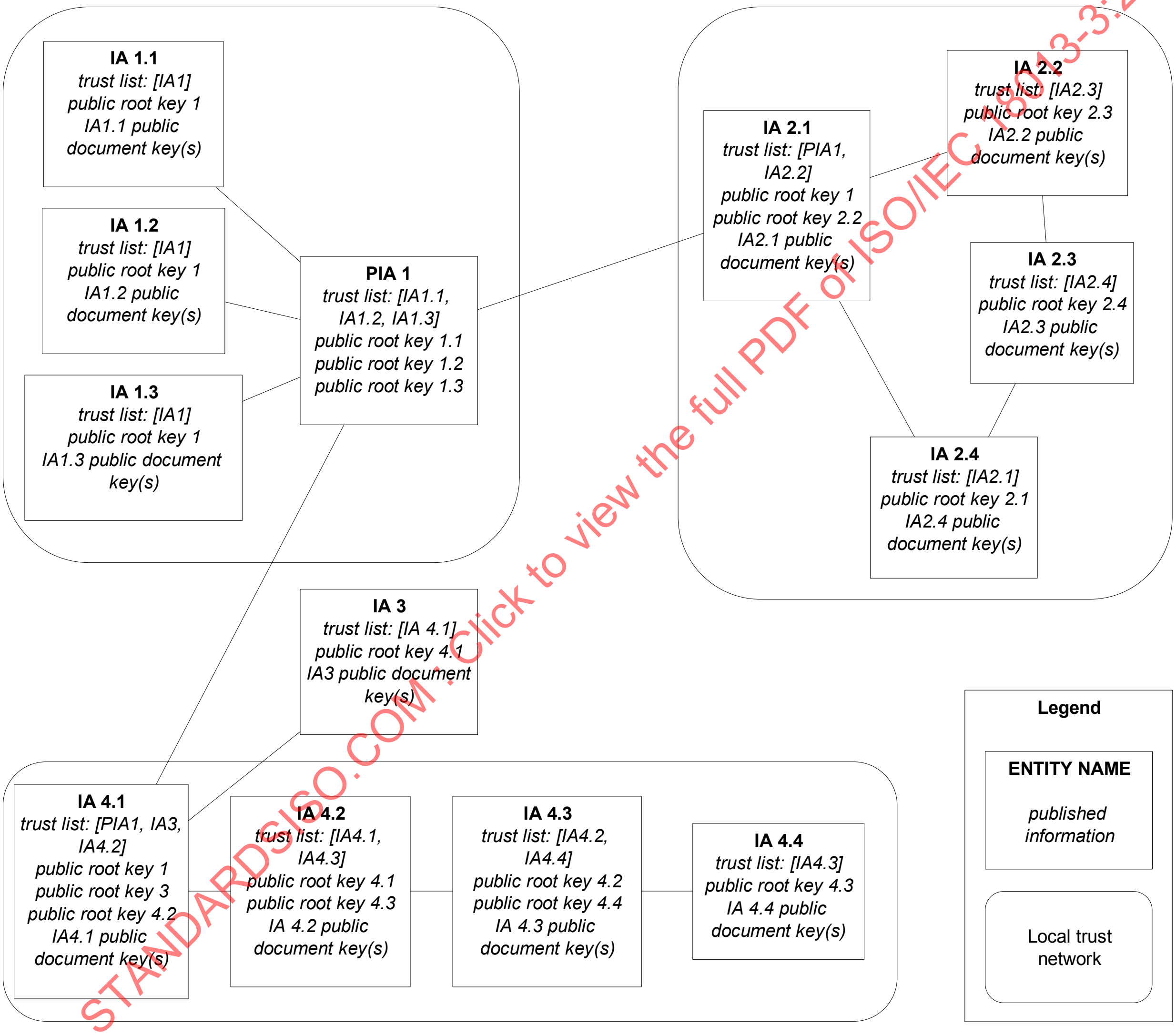


Figure A.2 — Trust network example

[15] The proposed trust model requires the unrestricted availability of the public root key. The ICAO PKI Technical Report (see bibliography) however appears to discourage such availability of ICAO's equivalent of the public root key.

A.4 Implementation

A.4.1 Publication mechanism

It is suggested that the information to be published, be published (as signed certificates) in a location that is easily accessible by all verifying entities. Consequently, each IA's identifying details should include directions on how and where published information can be accessed.

NOTE The Internet would be a location that complies with the above accessibility requirements. However, security concerns have been expressed about using the Internet. (If the Internet is used, a requirement to use server side authenticated SSL (similar to the ICAO requirement) for all communication can be added to enhance security. This implies that each IA will have to obtain a single server key from a commercial party.) The disadvantage of alternative publication locations in general is that they may not be readily accessible to all verifying entities (with the potential for consequential difficulties for the IA's card holders). In the end, it remains the responsibility of the IA to select a suitable publication location.

The security of the publication location is the responsibility of each IA. However, the authenticity of all published information can be verified using the IA's public root key, thus adding an additional layer of security to published information.

It follows that in cases where an IA may not have the necessary infrastructure or expertise or may not want to publish and/or maintain published information in-house, such publication may be performed (under agreement) by other IAs or trusted third parties.

The availability of published information is crucial. If published information is not available, a verifying entity cannot update its trust network diagram, and thus will be unaware of any changes in the trust network diagram such as new document keys issued, document keys revoked, or IAs removed from the trust network. This approach thus is vulnerable to a denial of service attack (see A.7.3).

A.4.2 Information published

As discussed in A.3, each IA (in its role as VA) publishes a trust list, and the public root key of each IA on the trust list. Each of these documents is signed with the publishing VA's private root key.

Each VA may optionally also publish the network trust diagram that it has constructed. Such a published network trust diagram can potentially be referenced by local industry (e.g. airlines and car hire companies) as a primary source to verify foreign driving licences.[16]

The VA (if it is also an IA) also publishes its own public document keys.

A standard format for each of the above documents (or certificates) has to be specified.

A.5 Certificate content

In considering certificate content, cognizance was taken of industry experts' recommendations that X.509 certificates should only be used if absolutely necessary[17]. Consequently, and in keeping with the intent of this annex (see A.1), the certificate content proposed below is based solely on the functional requirements pertaining to the IDL environment and the proposed trust model. The actual certificate profile eventually specified may or may not be X.509 compliant.

NOTE X.509 compliancy has advantages and disadvantages, which need to be carefully considered by an IA prior to implementation. The disadvantages are discussed in the noted references, and in essence involve unneeded complexity

[16] In a sense, a published trust network diagram is similar to the public key directory defined in the ICAO PKI Technical Report (see bibliography), with the difference that it is intended for local use only. However, nothing prevents a company in e.g. Australia to use (because it trusts) a network trust diagram published by e.g. Germany.

[17] See [3], [4] and [5] in the bibliography.

and questionable "fitness for purpose". The disadvantage to using a non-X.509 certificate is that IAs' existing environments may already be set up to accommodate only X.509 certificates.

A.5.1 Verification processes

As mentioned above, the functional processes drive the certificate contents. This clause discusses the processes involved.

When a public document key certificate needs to be verified, the following steps are executed:

a) Identify the IA that signed the certificate. Primary identification is based on the ISO Issuer Number in Data Group (DG) 3 (this field thus becomes mandatory if a digital signature is used). Additional identification information includes directions on how and where information published by the IA can be accessed.

b) Using a trust network diagram[18], identify the IA's public root key(s). If the IA has published more than one root key, identify the public root key used associated with the public document key certificate. The following options exist to identify this particular public root key:

    1) Include a public root key identifier in the public document key certificate, and include the same identifier in the public root key certificate.

    2) Use the date that the public document key certificate was signed to identify the public root key that was used for signing certificates during this period. This requires that the public root key certificate contain "valid for signing from" and "valid for signing until" dates, and limits the IA to the use of only one key at any given time.

c) Use the IA's public root key to check the public document key certificate's digital signature. This requires knowledge of the digital signature algorithm (and accompanying parameters) used. The algorithm and parameters are specified in the public document key certificate.

d) Verify that the public document key is still valid, that is, that the document key has not been revoked yet. This can be set up in either of the following ways:

    1) Use certificate revocation lists.

    2) Use a "Revoked Y/N" field in the public document key certificate[19]. This is unusual in that it implies that a certificate for the same key can be issued twice. However, it also negates the need for certificate revocation lists (refer to A.6.2 for more information), thus reducing the complexity of the overall solution.

When an IDL needs to be verified, the following actions are involved:

a) Identify the IA. The same identification process as for the public document key certificate verification is followed.

b) Obtain the correct public document key. Regardless of whether or not the public document key is included on the IDL, an attempt should be made to obtain the public document key from the appropriate public document key certificate on the IA's publication area (or alternatively as included in the VA's most recent trust network diagram).

[18] If the IA does not appear on the VA's trust network diagram, the public document key certificate cannot be verified. The IA first has to be added to the VA's trust network diagram via the out-of-band exchange of the IA's public root key with any of the existing IAs on the VA's trust network diagram.

[19] This field would always be "N" for a public key certificate included on the IDL. Only public keys published by the IA can have a "Y" value for this field.

For compact encoding, the appropriate public document key is identified by comparing the issue date of the IDL with the "valid for signing from" and "valid for signing until" dates of the available public document keys of the IA. For standard encoding, the appropriate public document key can be identified using a key identifier.

c) Verify the public document key certificate, as described above.

d) Check the digital signature on the IDL using the public document key. This requires knowledge of the digital signature algorithm and parameters used to sign the IDL. This information is stored in DG.SOD. (for both compact encoding and standard encoding).

A.5.2 Document key

Based on the discussion in A.5.1, a public document key certificate should contain the following information:

a) IA identifying details.

b) Public document key.

c) Public document key identifier (if dates are not used to identify the public document key).

d) Identifier of the public root key used to sign the document public key certificate (if dates are not used to identify the public root key).

e) Beginning (and ending) issuing dates for which the key is valid (at the time of signing the certificate) (if dates are used to identify the public document key).

f) Digital signature algorithm and parameter information.

g) Revocation indicator (Yes/No).

h) Revocation date (mandatory if revocation indicator is Yes) (if dates are used to identify the public document key).

i) Notes (optional). This can be used to add additional information regarding the revocation, in English, French or Spanish if required.

j) Date of signing (if dates are used to identify the public root key). This date is used to identify the private key that was used to sign the certificate. This is optional if the public key is included on the IDL, as it can be assumed that the date of signing is the same as the IDL issue date.

k) Digital signature (assuming a digital signature with appendix scheme).

A.5.3 Root key

Based on the discussion in A.5.1, a root key certificate should consist of the following information:

a) IA identifying details.

b) Public root key.

c) Public root key identifier (if dates are not used to identify the public root key)

d) Beginning (and ending) signing dates for which the key is valid (at the time of signing the certificate).

e) Date of signing (optional).

f) Digital signature (assuming a digital signature with appendix scheme).

Note that whenever a private document key is compromised, all the documents signed with the private key become suspect (and not only those documents signed prior to the compromise).

A.6 Key revocation

A.6.1 Root key compromise

A private root key compromise has the following far-reaching consequences:

a) None of the documents signed with the private root key can be considered "safe" any more.

b) The IA (whose private root key has been compromised) is effectively deleted from the trust network.

Due to the above, notification via publication (as defined in A.4.1) is infeasible. The fact that the private root key has been compromised can be published, but technically no VA will be able to verify such information, since there is no way in which the compromised IA can sign such publication.

In case of a compromise, the compromised IA thus has to notify all the immediate VAs of the compromise in an out-of-band fashion. Following such notification, the immediate VAs have to immediately remove the compromised IA from their trust lists (and their trust network diagrams). The compromised IA now has to create a new root key pair, and distribute the new public root key to the immediate VAs. At the same time, the compromised IA has to resign all other information that it wishes to publish, e.g. the trust list, public root keys for the IAs on the trust list, and the compromised IA's existing public document keys.

It thus is in each IA's best interest to communicate any compromise as expediently as possible to all immediate VAs. To this end, each IA should keep a list of all its immediate VAs. It is also recommended that the compromised IA re-obtain (out-of-band) the public root keys of the IAs in the compromised IA's trust list, to ensure that the public keys re-signed and re-published by the compromised IA (with the IA's new root private key) are in fact the correct keys.

NOTE The list of an IA's immediate VAs is not necessarily the same as the IA's trust list. An IA may not trust all the VAs to which it provided a copy of its public root key (by out-of-band means).

For non-immediate VAs the compromised IA's public root key is essentially revoked when the compromised IA is removed from the immediate VAs' trust lists. Each VA thus needs to carefully consider the frequency with which it updates its trust network diagram.

A.6.2 Document key compromise

The compromise of a private document key is easier to communicate and process than in the case of a private root key. When a private document key is compromised, the compromised IA (i.e. the IA whose private document key was compromised) simply re-publishes the public document key associated with the compromised private document key with the value of the "Revocation indicator" field set to "Y", and (if necessary) the value of the ending issuing date associated with the compromised public key set to the date the private document key was last used to issue an IDL.

This approach does away with the need to maintain certificate revocation lists. The compromised key is noted as such by any other VA the moment the VA updates its trust network diagram.

A.7 Trust model weak points

A.7.1 Introduction

The trust model presented in this annex is to a large extent predetermined by the design principles and constraints stated in A.2. However, since no trust model is perfect, it is important to also take note of the weak points of the model, so that these can be adequately addressed. This clause points out some of the weak points of the trust model. Weak points concerning other areas (e.g. testing, issuing, document security) are not discussed here.

A.7.2 General

One of the inherent weaknesses of the "trust by proxy" approach is that VAs may not all have the same criteria for measuring trustworthiness. If IA A trusts IA B and IA B trusts IA C, but IA A and IA B do not use the same criteria to determine trustworthiness then IA C does not necessarily comply with IA A's trustworthiness criteria, even though the "trust by proxy" approach would imply that IA A can trust IA C.

The trust chain can also become rather long, introducing ever more opportunity for a trust breakdown. A solution to this conundrum is for a VA to adapt its trust network (within the constraints imposed by cost and logistical considerations) so that higher volume IDL verifications "flow" over shorter trust chains. That is, a VA could establish direct trust relationships in such a manner that it minimizes the sum of chain lengths for all IDLs verified.

NOTE: This approach does not guarantee the lowest probability of letting a problematic IDL slip through. Knowing about the above approach, criminals may specifically target the end points of long trust chains in a VA's trust network in their attempts to circumvent the system.

If a breakdown in trust occurs or is suspected (i.e. it is determined or suspected that an IDL is verified when it shouldn't have been, or vice versa), the point where it is discovered may be several trust points removed from the IA. The longer the trust chain involved, the more cumbersome it becomes to confirm a trust breakdown and to identify the point of compromise. As described above, minimizing the sum of chain lengths for all IDLs verified can minimize this problem.

A VA may include an IA in a trust list for political reasons, even if internally the VA does not trust the IA. Even though such conduct would effectively sabotage the trust model, it cannot be ruled out. Consequently, it is important that VAs augment the trust model with other methods of establishing trust in an IDL issued by an IA.

A.7.3 Attacks

In general terms, (at least) the following attacks are possible against any two-level PKI:

a) Obtain private root key.

b) Replace private root key.

c) Replace trusted root key.

d) Obtain private document key.

e) Replace private document key.

f) Replace trusted document key.

g) Denial of service attack.

The above attacks can take many shapes and forms, some of which are unique to the trust model used, and others that would be applicable regardless of the trust model. The paragraphs that follow discuss some of the attacks that are specific to the proposed trust model.

The distributed nature of the proposed trust model requires each IA to take responsibility for the safekeeping of its own private keys. The general level of security which an IA is capable of or willing to employ, or the conscientiousness with which it follows its security procedures, may be less than would have been the case with one central (or even more than one) commercial certification authority. This can create a weak spot in the proposed trust model.

Trust in a public root key is established by exchanging such keys by out-of-band means. The main attack on such an exchange would be to replace the public key somewhere in the process. IAs should be aware of this risk, and implement appropriate procedures to secure their out-of-band key exchanges.

Several attacks against published information (e.g. trust lists, public root and document keys) are possible. The appeal of some of these attacks, and the expected duration before the attacks are uncovered, are related to the frequency with which a trusted root public key is used to verify certificates that were supposed to be signed by the trusted root public key (essentially the frequency with which the trust network diagram is verified). Other attacks (on published information) are not influenced by the frequency of trust network verification. These attacks are however likely to be uncovered over time, when it is realized that IDLs that do not verify is due to incorrect public keys being used to try and verify authentic IDLs.

A variant on the published information attack is to try and sneak the details of a fictitious IA into the trust list and public keys published by an existing IA. Although this would require some inside help (as the information has to be signed using the existing IA's private root key), this subterfuge can potentially remain undetected.

Denial of service attacks could be used on their own or in conjunction with some of the attacks mentioned above. A denial of service attack will prevent a VA from updating its trust network diagram, creating opportunities for many other attacks.

The ultimate decision on which IAs to trust and which IAs not to trust lies with the VA. Setting up and maintaining a trust network diagram thus requires active involvement from the VA. Any VA that does not take this responsibility seriously or does not allocate sufficient resources can become a liability for the whole trust network.

The bigger and more complex a trust network diagram becomes, the more opportunity there is for situations arising that require manual decision-making. For example: IA A has immediate VAs B and C. VA D's trust network diagram has paths to both VA B and VA C. For some reason, VA B removes IA A from its trust list. Does VA D now also remove IA A from its trust network diagram? Another example: After investigating IA B's security procedures and practices, VA A decides not to include IA B in its trust list. However, VA A does include IA C in its trust list, and IA C includes IA B in its trust list. VA A thus has to manually ensure that IA B is not included in its trust network diagram.

Annex B
(normative)

Basic access protection

B.1 Introduction

BAP is a mechanism and protocol to protect identity documents with a SIC against skimming attacks.

This protection is achieved by requiring the establishment of a secure channel using pre-defined key material which should only be revealed by closer physical inspection of the document, before access is granted to information stored in the SIC.

The secure channel protects the integrity and authenticity of authorized communication between an IS and an identity document. If the entropy of the key material is high enough, a certain amount of protection against eavesdropping attacks is also achieved.

NOTE 1 Because the same pre-defined key material is used for all communication sessions with a given document, this protocol does not give forward-secrecy. This means that, if knowledge about the keying material is gained, it can be used to decrypt any past sessions. However, knowledge of a particular session's session keys does not enable the decryption of past or future sessions.

NOTE 2 This protocol is largely based on ICAO's Basic Access Control, which can be viewed as an application of BAP specific to machine-readable travel documents.

B.2 Parameters

IA's implementing BAP shall select:

a) a cryptographic hash function *h*,

b) a block cipher *e* with block length *n* (in bits) and key length *k* (in bits), and

c) a cipher-based message authentication code (MAC) algorithm *m*.

Valid combinations thereof are listed in B.8.

Referring specifications shall specify the following when referencing BAP:

a) The source of K_{doc}.

b) The method(s) by which K_{doc} is entered into the IS.

c) The BAP configuration used.

NOTE ISO/IEC 18013-3 uses information printed on the IDL in machine and/or human-readable form as K_{doc}. The SAI demarcates the information used. The first byte of the input string indicates which BAP configuration is used. For further details, see 8.3 and 8.5.

B.3 Protocol

BAP comprises the following steps:

a) Document basic access keys are established using the key derivation mechanism described in B.4.

b) The IS and the SIC mutually authenticate and derive session keys. The authentication and key establishment protocol described in B.5 is used.

c) After successful authentication, subsequent communication is protected by Secure Messaging as described in B.6. Access shall only be granted as long as secure messaging is active.

B.4 Key derivation mechanism

The following key derivation mechanism is used to derive keys from a key seed (K_{seed}) for both the establishment of the document basic access keys and the establishment of the session keys for secure messaging.

A 32-bit counter c is used to allow for deriving multiple keys from a single seed. Depending on whether a key is used for encryption or MAC computation, the following values shall be used:

- c = 1 (i.e. ‘00 00 00 01’) for encryption,
- c = 2 (i.e. ‘00 00 00 02’) for MAC computation.

The following steps are performed to derive a key K from the seed K_{seed} and c using the selected cryptographic hash function *h*:

1. Let D be the concatenation of K_{seed} and c ($D = K_{seed} || c$).
2. Using *h*, calculate the hash H of D ($H = h(D)$).
3. The *k* left-most bits of H form the key K.

The document basic access keys K_{enc} and K_{mac} are derived using the mechanism described above, with c = 1 and 2 respectively. In addition, the most significant 16 bytes of the SHA-1 hash of K_{doc} is used as the value for K_{seed}.

K_{doc} should be different for every document and care should be taken to ensure that K_{doc} is sufficiently random for the intended application.

NOTE The entropy of K_{doc} is the upper bound on available entropy for the secure messaging keys. For example, if K_{doc} only provides 30 bits of entropy, the derived keys cannot contain more – even if the key size is larger.

B.5 Authentication and key establishment

Authentication and key establishment shall be provided by a three pass challenge-response protocol according to ISO/IEC 11770-2:1996 key establishment mechanism 6 using the selected block cipher *e*. A cryptographic checksum according to the selected MAC algorithm *m* is calculated over and appended to the cipher texts. The modes of operation described in B.7 shall be used. Exchanged nonces shall have a size of 64 bits, exchanged keying material shall be *k* bits long. Distinguishing identifiers shall not be used.

In more detail, the IS and SIC perform the following steps[20]:

1. The IS requests a challenge RND.ICC by sending the GET CHALLENGE command.
2. The SIC generates and responds with a random nonce RND.ICC.
3. The IS performs the following operations:
   a) Generate a random nonce RND.IFD and random keying material K.IFD.
   b) Generate the concatenation S = RND.IFD || RND.ICC || K.IFD.
   c) Compute the cryptogram E_IFD = $e[K_{enc}](S)$.
   d) Compute the checksum M_IFD = $m[K_{mac}](E\_IFD)$.
   e) Send a MUTUAL AUTHENTICATE command using the data E_IFD || M_IFD.
4. The SIC performs the following operations:
   a) Check the checksum M_IFD of the cryptogram E_IFD.
   b) Decrypt the cryptogram E_IFD.
   c) Extract RND.ICC from S and check if the IS returned the correct value.
   d) Generate random keying material K.ICC.
   e) Generate the concatenation R = RND.ICC || RND.IFD || K.ICC
   f) Compute the cryptogram E_ICC = $e[K_{enc}](R)$.
   g) Compute the checksum M_ICC = $m[K_{mac}](E\_ICC)$.
   h) Send the response using the data E_ICC || M_ICC.
5. The IS performs the following operations:
   a) Check the checksum M_ICC of the cryptogram E_ICC.
   b) Decrypt the cryptogram E_ICC.
   c) Extract RND.IFD from R and check if the SIC returned the correct value.

B.6 Secure messaging

After a successful execution of the authentication protocol, both the IS and the SIC compute session keys KS_{enc} and KS_{mac} using the key derivation mechanism described in B.4 with (K.ICC ⊕ K.IFD) as key seed. All further communication shall be protected by secure messaging (SM) as defined in ISO/IEC 7816-4:2005 according to the requirements below. The modes of operation described in B.7 shall be used.

[20] The abbreviations IS and SIC used here are equivalent to IFD and ICC respectively as used in ISO/IEC 7501-1 (ICAO Doc 9303-1).

B.6.1 Message structure of SM APDUs

The SM data objects shall be used according to Table B.1 in the following order:

- Command APDU: [DO‘85’ or DO‘87’] [DO‘97’] DO‘8E’.
- Response APDU: [DO‘85’ or DO‘87’] DO‘99’ DO‘8E’.

All SM data objects shall be encoded in BER-TLV as specified in ISO/IEC 7816-4:2005. The command header shall be included in the MAC calculation, therefore the class byte CLA shall be ‘0C’.

The actual value of Lc will be modified to Lc’ after application of secure messaging. In the protected command APDU the *new Le* byte shall be set to ‘00’, while the value of the original Le byte may be conveyed in the appropriate data object.

Table B.1 — Usage of SM Data Objects

	DO‘85’*	**DO‘87’**	**DO‘97’**	**DO‘99’**	**DO‘8E’**
Content	Cryptogram (plain value encoded in BER-TLV, but not including SM data objects)	Padding-content indicator byte (shall be ‘01’) followed by the cryptogram	Le (1 or 2 bytes)	Processing status (SW1-SW2)	Cryptographic checksum (MAC)
Command APDU	Mandatory if data is sent, otherwise absent.	Mandatory if data is sent, otherwise absent.	Mandatory if data is requested, otherwise absent.	Not used.	Mandatory.
Response APDU	Mandatory if data is returned, otherwise absent.	Mandatory if data is returned, otherwise absent.	Not used.	Mandatory, only absent when a SM error occurs.	Mandatory if DO‘87’ and/or DO‘99’ are present.

* DO‘85’ (odd INS byte) or DO‘87’ (even INS byte) is used

Figure B.1 shows the transformation of an unprotected command APDU to a protected command APDU in the case that data and Le are available (case 4). If no data is available (case 1 and 2), leave building DO‘87’ out. If Le is not available (case 1 and 3), leave building DO‘97’ out.

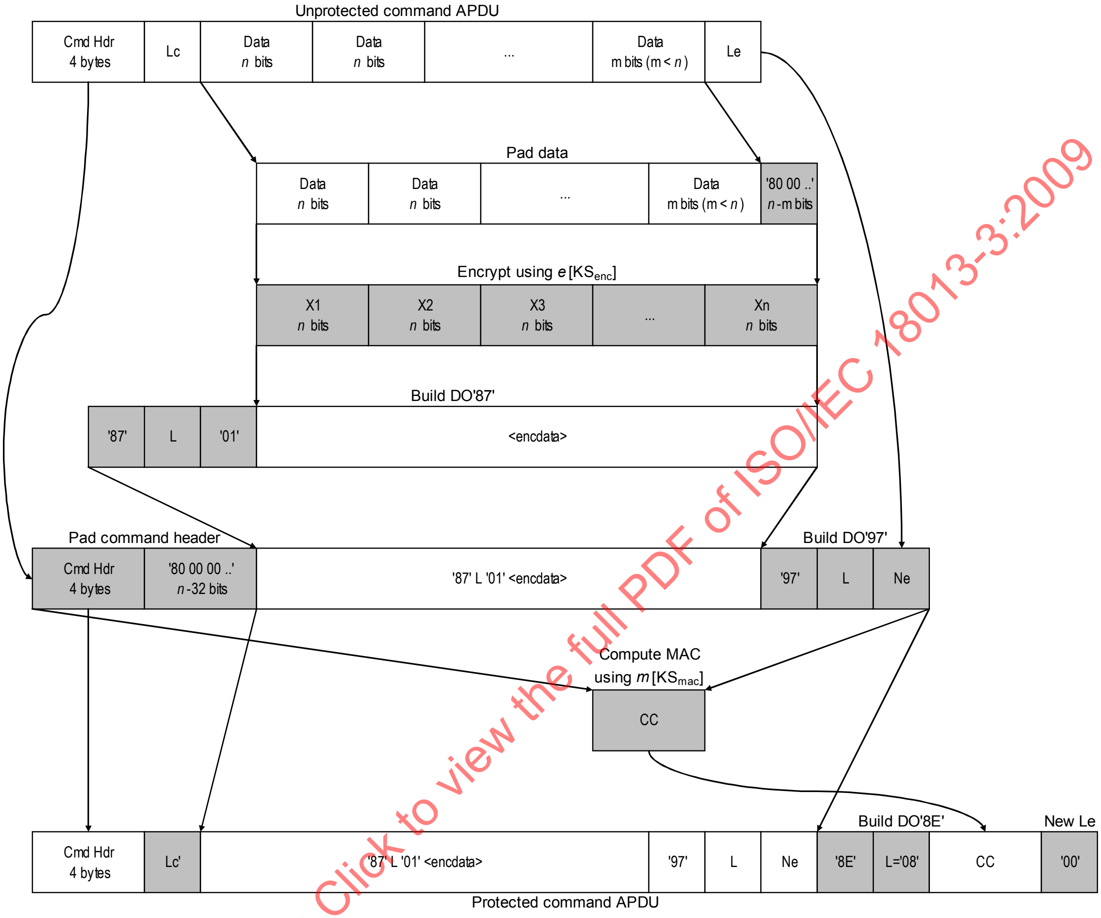


Figure B.1 — Computation of a SM command APDU

Figure B.2 shows the transformation of an unprotected response APDU to a protected response APDU in case data is available. If no data is available, leave building DO‘87’ out.

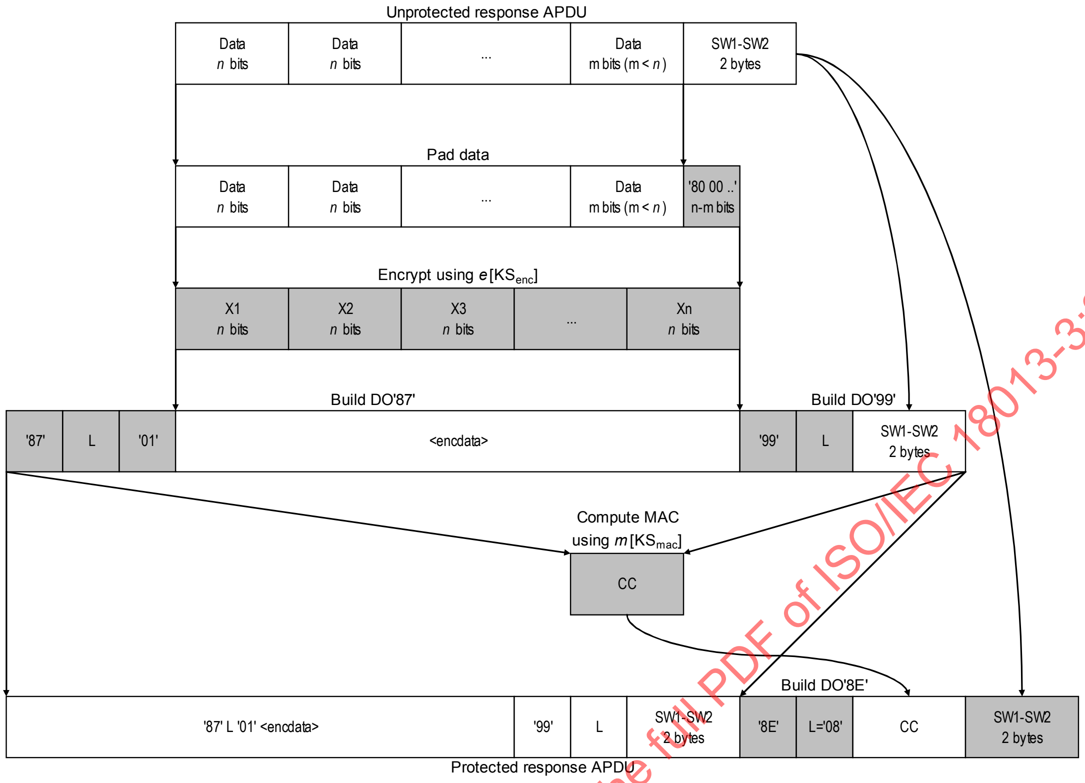


Figure B.2 — Computation of a SM response APDU

B.6.2 SM errors

When the integrated circuit in the document recognizes an SM error while interpreting a command, then the secure messaging session shall be aborted and the status words returned in plain. ISO/IEC 7816-4:2005 defines the following status bytes to indicate SM errors:

a) ‘6987’ Expected SM data objects missing

b) ‘6988’ SM data objects incorrect

B.6.3 Other errors

In the application context, other errors (i.e. status words other than ’90 00’) may occur that are protected under SM. Under these conditions SM shall not be aborted.

B.7 Modes of operation

B.7.1 Encryption

During encryption, the selected block cipher shall operate in cipher block chaining (CBC) mode with an initialization vector (IV) of *n* ‘0’ bits.

During authentication, the data to be encrypted shall only be padded if it is not a multiple of the block cipher’s block length *n*. During the computation of SM APDUs, data shall always be padded.

Padding according to ISO/IEC 9797-1:1999 padding method 2 shall be used.

B.7.2 Message authentication

Cryptographic checksums are calculated using the selected MAC algorithm with an initialization vector (IV) of *n* ‘0’ bits. The MAC length shall be 8 bytes.

After a successful authentication, the datagram to be MACed shall be prepended with an 8 byte send sequence counter (SSC). If the block length n is larger than 64 bits, the SSC shall be prepended by n-64 zero bits to form a full block. The SSC is incremented every time before a MAC is calculated, i.e. if the starting value is x, in the first command the value of SSC is x+1. The value of SSC for the first response is then x+2. The initial value of the SSC is computed by concatenating the four least significant bytes of RND.ICC and RND.IFD, respectively:

SSC = RND.ICC (4 least significant bytes) || RND.IFD (4 least significant bytes)

B.8 Basic access protection configurations

When selecting *h*, *e*, *n*, *k* and *m*, IA's shall pick a valid combination, herein called “configuration”, from the choices in this section.

Referring specifications shall either limit the choice of configurations to one, or specify a way to convey the chosen configuration to an IS.

NOTE The criteria for selecting a configuration are typically determined by the application’s security, performance and cost requirements.

The following algorithms are used by the configurations:

- SHA-1 and SHA-256 according to ISO/IEC 10118-3:2004;
- TDEA according to ISO/IEC 18033-3:2005 ("Triple DES");
- AES-128, AES-192 and AES-256 according to ISO/IEC 18033-3:2005;
- ISO/IEC 9797-1:1999 MAC algorithm 3;
- CMAC according to NIST SP 800-38B.

Table B.2 — BAP configuration 1

One-byte identifier	'31'
OID	`bap-config-1`
Hash Algorithm *h*	SHA-1
Block Cipher *e*	TDEA using keying option 2. The left-most 64 bits of the 128-bit key form K1, while the right-most 64 bits form K2.
Block Length *n*	64 bits
Key Length *k*	128 bits Note that only 112 bits are effectively used by this block cipher as keying material; certain implementations may require adjustment of the remaining parity bits.
MAC Algorithm *m*	ISO/IEC 9797-1:1999 MAC algorithm 3 with block cipher TDEA and padding method 2. TDEA is used with the keying option that K1=K2=K3 (reduces to DEA). For MAC calculation, the left-most 64 bits of the 128-bit key form K, while the right-most 64 bits form K’. The resulting MAC algorithm is also known as “Retail MAC”.

BAP configuration 1 is equivalent to Basic Access Control (BAC) as described in ICAO Doc 9303-1 (ISO/IEC 7501-1), Volume II.

Table B.3 — BAP configuration 2

One-byte identifier	'32'
OID	bap-config-2
Hash Algorithm *h*	SHA-1
Block Cipher *e*	AES-128
Block Length *n*	128 bits
Key Length *k*	128 bits
MAC Algorithm *m*	CMAC using AES-128

Table B.4 — BAP configuration 3

One-byte identifier	'33'
OID	bap-config-3
Hash Algorithm *h*	SHA-256
Block Cipher *e*	AES-192
Block Length *n*	128 bits
Key Length *k*	192 bits
MAC Algorithm *m*	CMAC using AES-192

Due to the explicit exclusion of AES-192 in Suite B of the United States National Security Agency, BAP configuration 3 is not recommended.

Table B.5 — BAP configuration 4

One-byte identifier	'34'
OID	bap-config-4
Hash Algorithm *h*	SHA-256
Block Cipher *e*	AES-256
Block Length *n*	128 bits
Key Length *k*	256 bits
MAC Algorithm *m*	CMAC using AES-256

The following ASN.1 object identifiers are used to refer to the different BAP configurations:

```
   bap-config-1 OBJECT IDENTIFIER ::= {
      iso(1) standard(0) driving-licence(18013) part-3(3) security-mechanisms(2) id-sm-
BAP(1) 1
   }

   bap-config-2 OBJECT IDENTIFIER ::= {
      iso(1) standard(0) driving-licence(18013) part-3(3) security-mechanisms(2) id-sm-
BAP(1) 2
   }

   bap-config-3 OBJECT IDENTIFIER ::= {
      iso(1) standard(0) driving-licence(18013) part-3(3) security-mechanisms(2) id-sm-
BAP(1) 3
   }

   bap-config-4 OBJECT IDENTIFIER ::= {
      iso(1) standard(0) driving-licence(18013) part-3(3) security-mechanisms(2) id-sm-
BAP(1) 4
   }
```

B.9 Card Commands

B.9.1 GET CHALLENGE

The GET CHALLENGE command receives a (true) random challenge from the card for authentication in the subsequent MUTUAL AUTHENTICATE command.

Table B.6 — Command APDU: GET CHALLENGE

CLA	As defined in ISO/IEC 7816-4:2005	
INS	0x84	GET CHALLENGE
P1	0x00	No information given
P2	0x00	(any other values reserved for future use)
Lc field	Absent	
Data field	Absent	
Le field	0x08	

Table B.7 — Response APDU: GET CHALLENGE

Data field	8-byte random challenge (RND.ICC)	
SW1-SW2	'9000' Other	Normal processing Operating system dependent error

B.9.2 MUTUAL AUTHENTICATE

The MUTUAL AUTHENTICATE command is used to submit the host cryptogram to the card and receive the card cryptogram for mutual authentication.

Table B.8 — Command APDU: MUTUAL AUTHENTICATE

CLA	As defined in ISO/IEC 7816-4:2005
INS	0x82 MUTUAL AUTHENTICATE
P1	0x00 reference algorithm implicitly known
P2	0x00 qualifier reference implicitly known
Lc field	Length of subsequent data field.
Data field	Host cryptogram including MAC (E_IFD || M_IFD).
Le field	0x28 (configurations 1 and 2) 0x38 (configurations 3 and 4)

Table B.9 — Response APDU: MUTUAL AUTHENTICATE

Data field	Card cryptogram including MAC (E_ICC || M_ICC)	
SW1-SW2	'9000' '6300' Other	Normal processing Verification failed. Host cryptogram or MAC verification failed Operating system dependent error

B.10 Worked examples

This section provides worked examples for all configurations of BAP. Note that not all steps are explicitly shown.

B.10.1 Example Using Configuration 1

Static document keying material:

K_{doc} = '239AB9CB282DAF66231DC5A4DF6BFBAE'

Computation of basic access keys:

Input: K_{seed} = K_{doc}

Encryption Key (K_{enc}) computation:

1. Concatenate K_{seed} and c (c = 1):
   D = '239AB9CB282DAF66231DC5A4DF6BFBAE
   00000001'

2. Calculate the hash of D:
$H_{SHA-1}(D)$ = ‘AB94FCEDF2664EDFB9B291F85D7F77F2
7F2F4A9D’

3. Form key:
K_{enc} = ‘AB94FCEDF2664EDFB9B291F85D7F77F2’

$K_1 = K_3$ = ‘AB94FCEDF2664EDF’
K_2= ‘B9B291F85D7F77F2’

Message Authentication Key (K_{mac}) computation:

4. Concatenate K_{seed} and c (c = 2):
D = ‘239AB9CB282DAF66231DC5A4DF6BFBAE
00000002’

5. Calculate the hash of D:
$H_{SHA-1}(D)$ = ‘7862D9ECE03C1BCD4D77089DCF131442
814EA70A’

6. Form key:
K_{mac} = ‘7862D9ECE03C1BCD4D77089DCF131442’

K = ‘7862D9ECE03C1BCD’
K’ = ‘4D77089DCF131442’

Authentication and Establishment of Session Keys:

IS:

1. Request an 8 byte random challenge from the document’s SIC:

Command APDU:

CLA	INS	P1	P2	Le
‘00’	‘84’	‘00’	‘00’	‘08’

Document SIC:

2. Generate random challenge and return it to IS:
RND.ICC = ‘4608F91988702212’

Response APDU:

Response Data Field	SW1	SW2
RND.ICC	‘90’	‘00’

IS:

3. Generate an 8-byte random challenge and 16-byte random keying material:
RND.IFD = ‘781723860C06C226’
K.IFD = ‘0B795240CB7049B01C19B33E32804F0B’

4. Concatenate RND.IFD, RND.ICC and K.IFD:
S = ‘781723860C06C2264608F91988702212
0B795240CB7049B01C19B33E32804F0B’

5. Encrypt S using TDEA with key K_{enc}:
E_IFD = ‘72C29C2371CC9BDB65B779B8E8D37B29
ECC154AA56A8799FAE2F498F76ED92F2’

6. Compute “Retail MAC” over E_IFD with key K_{mac}:
M_IFD = ‘5F1448EEA8AD90A7’

7. Construct command data for MUTUAL AUTHENTICATE and send command to the document’s SIC:
cmd_data = ‘72C29C2371CC9BDB65B779B8E8D37B29
ECC154AA56A8799FAE2F498F76ED92F2
5F1448EEA8AD90A7’

Command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘00’	‘84’	‘00’	‘00’	‘28’	cmd_data	‘28’

Document SIC:

8. Generate 16-byte random keying material:
K.ICC = ‘0B4F80323EB3191CB04970CB4052790B’

9. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘0036D272F5C350ACAC50C3F572D23600’

10. Derive session keys:
KS_{enc} = ‘969EC03B1CBFE9DDD11AB1FED206EBE4’
KS_{mac} = ‘F0CA1E1EB5ADF208816B88DD579CC1F8’

11. Initialize send sequence counter:
SSC = ‘887022120C06C226’

12. Concatenate RND.ICC, RND.IFD and K.ICC:
R = ‘4608F91988702212781723860C06C226
0B4F80323EB3191CB04970CB4052790B’

13. Encrypt R using TDEA with key K_{enc}:
E_ICC = ‘46B9342A41396CD7386BF5803104D7CE
DC122B9132139BAF2EEDC94EE178534F’

14. Compute “Retail MAC” over E_ICC with key K_{mac}:
M_ICC = ‘2F2D235D074D7449’

15. Construct response data and send response APDU to the IS:
resp_data = ‘46B9342A41396CD7386BF5803104D7CE
DC122B9132139BAF2EEDC94EE178534F
2F2D235D074D7449’

Response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

16. Calculate XOR of K.IFD and K.ICC:
K_{seed} = '0036D272F5C350ACAC50C3F572D23600'

17. Derive session keys:
KS_{enc} = '969EC03B1CBFE9DDD11AB1FED206EBE4'
KS_{mac} = 'F0CA1E1EB5ADF208816B88DD579CC1F8'

18. Initialize send sequence counter:
SSC = '887022120C06C226'

Secure Messaging:

IS

1. SELECT EF.COM (file identifier = '01 1E'):

Unprotected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field
'00'	'A4'	'02'	'00'	'02'	'01 1E'

a) Mask class byte and pad command header:
cmd_header = '0CA4020C80000000'

b) Pad data:
p_data = '011E800000000000'

c) Encrypt p_data using TDEA with KS_{enc}:
enc_data = '6375432908C044F6'

d) Build DO'87':
DO87 = '8709016375432908C044F6'

e) Concatenate cmd_header and DO87:
M = '0CA4020C800000008709016375432908
C044F6'

f) Compute "Retail MAC" of M with KS_{mac}:
- Increment SSC:
SSC = '887022120C06C227'
- Concatenate SSC and M:
N = '887022120C06C2270CA4020C80000000
8709016375432908C044F6'
- Compute MAC:
CC = 'BF8B92D635FF24F8'

g) Build DO'8E':
DO8E = '8E08BF8B92D635FF24F8'

h) Construct command data:
cmd_data = '8709016375432908C044F68E08BF8B92
D635FF24F8'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘A4’	‘02’	‘0C’	‘15’	cmd_data	‘00’

Document SIC:

2. Set EF.COM as the currently selected file and send affirmative response to IS:

Unprotected response APDU:

SW1	SW2
‘90’	‘00’

a) Build DO‘99’:
DO99 = ‘99029000’

b) Compute “Retail MAC” of DO99 with KS_{mac}:
- Increment SSC:
SSC = ‘887022120C06C228’
- Concatenate SSC and DO99:
N = ‘887022120C06C22899029000’
- Compute MAC:
CC = ‘FA855A5D4C50A8ED’

c) Build DO‘8E’:
DO8E = ‘8E08FA855A5D4C50A8ED’

d) Construct response data:
resp_data = ‘990290008E08FA855A5D4C50A8ED’

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

3. READ BINARY of the first 4 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
‘0C’	‘B0’	‘00’	‘00’	‘04’

a) Mask class byte and pad command header:
cmd_header = ‘0CB0000080000000’

b) Build DO ‘97’:
DO97 = ‘970104’

c) Concatenate cmd_header and DO97:
M = ‘0CB0000080000000970104’

d) Compute “Retail MAC” of M with KS_{mac}:
- Increment SSC:
  SSC = '887022120C06C229'
- Concatenate SSC and M:
  N = '887022120C06C2290CB0000080000000970104'
- Compute MAC:
  CC = 'ED6705417E96BA55'

e) Build DO‘8E’:
DO8E = '8E08ED6705417E96BA55'

f) Construct command data:
cmd_data = '9701048E08ED6705417E96BA55'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘00’	‘0D’	cmd_data	‘00’

Document SIC:

4. Return 4 bytes of EF.COM starting at offset 0:

data = '600D5F01'

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = '600D5F0180000000'

b) Encrypt p_data using TDEA with KS_{enc}:
enc_data = 'F9435D056E27C52E'

c) Build DO‘87’:
DO87 = '870901F9435D056E27C52E'

d) Build DO‘99’:
DO99 = '99029000'

e) Concatenate DO‘87’ and DO‘99’:
M = '870901F9435D056E27C52E99029000'

f) Compute “Retail MAC” of M with KS_{mac}:
- Increment SSC:
  SSC = '887022120C06C22A'
- Concatenate SSC and M:
  N = '887022120C06C22A870901F9435D056E27C52E99029000'
- Compute MAC:
  CC = '0C15238078E0A4C9'

g) Build DO‘8E’:
DO8E = '8E080C15238078E0A4C9'

h) Construct response data:
resp_data = '870901F9435D056E27C52E990290008E
080C15238078E0A4C9'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	'90'	'00'

IS:

5. READ BINARY of the remaining 11 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
'0C'	'B0'	'00'	'04'	'0B'

a) Mask class byte and pad command header:
cmd_header = '0CB0000480000000'

b) Build DO '97':
DO97 = '97010B'

c) Concatenate cmd_header and DO97:
M = '0CB000048000000097010B'

d) Compute "Retail MAC" of M with KS_{mac}:
- Increment SSC:
SSC = '887022120C06C22B'
- Concatenate SSC and M:
N = '887022120C06C22B0CB0000480000000
97010B'
- Compute MAC:
CC = '40900A27C4C390D6'

e) Build DO'8E':
DO8E = '8E0840900A27C4C390D6'

f) Construct command data:
cmd_data = '97010B8E0840900A27C4C390D6'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
'0C'	'B0'	'00'	'04'	'0D'	cmd_data	'00'

Document SIC:

6. Return 11 bytes of EF.COM starting at offset 4:

data = '04303130305C04616B6567'

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = ‘04303130305C04616B65678000000000’

b) Encrypt p_data using TDEA with KS_{enc}:
enc_data = ‘B3CD0334417393661AA9B39206EC89CC’

c) Build DO‘87’:
DO87 = ‘871101B3CD0334417393661AA9B39206
EC89CC’

d) Build DO‘99’:
DO99 = ‘99029000’

e) Concatenate DO‘87’ and DO‘99’:
M = ‘871101B3CD0334417393661AA9B39206
EC89CC99029000’

f) Compute “Retail MAC” of M with KS_{mac}:
- Increment SSC:
SSC = ‘887022120C06C22C’
- Concatenate SSC and M:
N = ‘887022120C06C22C871101B3CD033441
7393661AA9B39206EC89CC99029000’
- Compute MAC:
CC = ‘0747E8CEC180EB48’

g) Build DO‘8E’:
DO8E = ‘8E080747E8CEC180EB48’

h) Construct response data:
resp_data = ‘871101B3CD0334417393661AA9B39206
EC89CC990290008E080747E8CEC180EB
48’

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

B.10.2 Example Using Configuration 2

Static document keying material:
K_{doc} = ‘55CA83CC52EC6454DE7AFB1D3DA66F4E’

Compute Basic Access Keys

Input: K_{seed} = K_{doc}

Encryption Key (K_{enc}) computation

1. Concatenate K_{seed} and c (c = 1):
D = ‘55CA83CC52EC6454DE7AFB1D3DA66F4E
00000001’

2. Calculate the hash of D:
H_{SHA-1}(D) = ‘EF8E4DCE3D33E8C6E690509A7DA5186C
C35B94BF’

3. Form key:
K_{enc} = ‘EF8E4DCE3D33E8C6E690509A7DA5186C’

Message Authentication Key (K_{mac}) computation

4. Concatenate K_{seed} and c (c = 2):
D = ‘55CA83CC52EC6454DE7AFB1D3DA66F4E
00000002’

5. Calculate the hash of D:
H_{SHA-1}(D) = ‘94B140FB3E4B557CCC3A225E794C077B
14518E58’

6. Form key:
K_{mac} = ‘94B140FB3E4B557CCC3A225E794C077B’

Authentication and Establishment of Session Keys:

IS

1. Request an 8 byte random challenge from the document’s SIC:

Command APDU:

CLA	INS	P1	P2	Le
‘00’	‘84’	‘00’	‘00’	‘08’

Document SIC:

2. Generate random challenge and return it to IS:
RND.ICC = ‘E72450C17A59DF40’

Response APDU:

Response Data Field	SW1	SW2
RND.ICC	‘90’	‘00’

IS:

3. Generate an 8-byte random challenge and 16-byte random keying material:
RND.IFD = ‘41C51B949D4FD786’
K.IFD = ‘766F9E2B2B503AFBEB420BED8D1A73A8’

4. Concatenate RND.IFD, RND.ICC and K.IFD:
S = ‘41C51B949D4FD786E72450C17A59DF40
766F9E2B2B503AFBEB420BED8D1A73A8’

5. Encrypt S using AES with key K_{enc}:
E_IFD = ‘80AEE3D8F601067546C4512979F1344E
F62E0045A94BD21A97E9AE2FE578AAB0’

6. Compute CMAC over E_IFD with key K_{mac}:
M_IFD = ‘320F4C5677E3618A’

7. Construct command data for MUTUAL AUTHENTICATE and send command to the document’s SIC:
cmd_data = ‘80AEE3D8F601067546C4512979F1344E
F62E0045A94BD21A97E9AE2FE578AAB0
320F4C5677E3618A’

Command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘00’	‘84’	‘00’	‘00’	‘28’	cmd_data	‘28’

Document SIC:

8. Generate 16-byte random keying material:
K.ICC = ‘C1655F49E136D12B8522B1C99510E71B’

9. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘B70AC162CA66EBD06E60BA24180A94B3’

10. Derive session keys:
KS_{enc} = ‘AB9497F5819AB69A25A7798789061CF8’
KS_{mac} = ‘1FBF06A0DE775C473D64B5E9933290D3’

11. Initialize send sequence counter:
SSC = ‘7A59DF409D4FD786’

12. Concatenate RND.ICC, RND.IFD and K.ICC:
R = ‘E72450C17A59DF4041C51B949D4FD786
C1655F49E136D12B8522B1C99510E71B’

13. Encrypt R using AES with key K_{enc}:
E_ICC = ‘649E3A8F8D48E22ADB7AAE09FE17BA43
377D9CAF94EFBD7BAF9707088DF6489F’

14. Compute CMAC over E_ICC with key K_{mac}:
M_ICC = ‘1CCE566F1FE43D65’

15. Construct response data and send response APDU to the IS:
resp_data = ‘649E3A8F8D48E22ADB7AAE09FE17BA43
377D9CAF94EFBD7BAF9707088DF6489F
1CCE566F1FE43D65’

Response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

16. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘B70AC162CA66EBD06E60BA24180A94B3’

17. Derive session keys:
KS_{enc} = ‘AB9497F5819AB69A25A7798789061CF8’
KS_{mac} = ‘1FBF06A0DE775C473D64B5E9933290D3’

18. Initialize send sequence counter:
SSC = ‘7A59DF409D4FD786’

Secure Messaging:

IS:

1. SELECT EF.COM (file identifier = ’01 1E’):

Unprotected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field
‘00’	‘A4’	‘02’	‘00’	‘02’	’01 1E’

a) Mask class byte and pad command header:
cmd_header = ‘0CA4020C800000000000000000000000’

b) Pad data:
p_data = ‘011E8000000000000000000000000000’

c) Encrypt p_data using AES with KS_{enc}:
enc_data = ‘4FF75761BC5C1ECE82AE43F70938D50F’

d) Build DO‘87’:
DO87 = ‘8711014FF75761BC5C1ECE82AE43F709
38D50F’

e) Concatenate cmd_header and DO87:
M = ‘0CA4020C800000000000000000000000
8711014FF75761BC5C1ECE82AE43F709
38D50F’

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = ‘7A59DF409D4FD787’
- Concatenate padded SSC and M:
N = ‘00000000000000007A59DF409D4FD787
0CA4020C800000000000000000000000
8711014FF75761BC5C1ECE82AE43F709
38D50F’
- Compute MAC:
CC = ‘6204A7709D0E128B’

g) Build DO‘8E’:
DO8E = ‘8E082CF23B10E4D30044’

h) Construct command data:
cmd_data = ‘8711014FF75761BC5C1ECE82AE43F709
38D50F8E086204A7709D0E128B’

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘A4’	‘02’	‘0C’	‘1D’	cmd_data	‘00’

Document SIC:

2. Set EF.COM as the currently selected file and send affirmative response to IS:

Unprotected response APDU:

SW1	SW2
‘90’	‘00’

a) Build DO‘99’:
DO99 = `'99029000'`

b) Compute CMAC of DO99 with KS_{mac}:
- Increment SSC:
  SSC = `'7A59DF409D4FD788'`
- Concatenate padded SSC and DO99:
  N = `'00000000000000007A59DF409D4FD788 99029000'`
- Compute MAC:
  CC = `'2CF23B10E4D30044'`

c) Build DO‘8E’:
DO8E = `'8E082CF23B10E4D30044'`

d) Construct response data:
resp_data = `'990290008E082CF23B10E4D30044'`

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

3. READ BINARY of the first 4 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
‘0C’	‘B0’	‘00’	‘00’	‘04’

a) Mask class byte and pad command header:
cmd_header = `'0CB00000800000000000000000000000'`

b) Build DO ‘97’:
DO97 = `'970104'`

c) Concatenate cmd_header and DO97:
M = `'0CB00000800000000000000000000000 970104'`

d) Compute CMAC of M with KS_{mac}:
- Increment SSC:
  SSC = '7A59DF409D4FD789'
- Concatenate padded SSC and M:
  N = '00000000000000007A59DF409D4FD789
  0CB00000800000000000000000000000
  970104'
- Compute MAC:
  CC = '3BEE3045A87E2A15'

e) Build DO‘8E’:
DO8E = '8E083BEE3045A87E2A15'

f) Construct command data:
cmd_data = '9701048E083BEE3045A87E2A15'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘00’	‘0D’	cmd_data	‘00’

Document SIC

4. Return 4 bytes of EF.COM starting at offset 0:

data = '600D5F01'

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = '600D5F01800000000000000000000000'

b) Encrypt p_data using AES with KS_{enc}:
enc_data = 'D60D14976646FB2304A0155F6BC6E42D'

c) Build DO‘87’:
DO87 = '871101D60D14976646FB2304A0155F6B
C6E42D'

d) Build DO‘99’:
DO99 = '99029000'

e) Concatenate DO‘87’ and DO‘99’:
M = '871101D60D14976646FB2304A0155F6B
C6E42D99029000'

f) Compute CMAC of M with KS$_{mac}$:
- Increment SSC:
  SSC = ‘7A59DF409D4FD78A’
- Concatenate padded SSC and M:
  N = ‘00000000000000007A59DF409D4FD78A
  871101D60D14976646FB2304A0155F6B
  C6E42D99029000’
- Compute MAC:
  CC = ‘D2165D993B20F73B’

g) Build DO‘8E’:
DO8E = ‘8E08D2165D993B20F73B’

h) Construct response data:
resp_data = ‘871101D60D14976646FB2304A0155F6B
C6E42D990290008E08D2165D993B20F7
3B’

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

4. READ BINARY of the remaining 11 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
‘0C’	‘B0’	‘00’	‘04’	‘0B’

a) Mask class byte and pad command header:
cmd_header = ‘0CB00004800000000000000000000000’

b) Build DO ‘97’:
DO97 = ‘97010B’

c) Concatenate cmd_header and DO97:
M = ‘0CB00004800000000000000000000000
97010B’

d) Compute CMAC of M with KS$_{mac}$:
- Increment SSC:
  SSC = ‘7A59DF409D4FD78B’
- Concatenate padded SSC and M:
  N = ‘00000000000000007A59DF409D4FD78B
  0CB00004800000000000000000000000
  97010B’
- Compute MAC:
  CC = ‘36F9817F84E009C8’

e) Build DO‘8E’:
DO8E = ‘8E0836F9817F84E009C8’

f) Construct command data:
cmd_data = ‘97010B8E0836F9817F84E009C8’

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘04’	‘0D’	cmd_data	‘00’

Document SIC:

4. Return 11 bytes of EF.COM starting at offset 4:

data = ‘04303130305C04616B6567’

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = ‘04303130305C04616B656780000000000’

b) Encrypt p_data using AES with KS_{enc}:
enc_data = ‘36B83A1FBAC98D89DDDA2235AD29A8BB’

c) Build DO‘87’:
DO87 = ‘87110136B83A1FBAC98D89DDDA2235AD
29A8BB’

d) Build DO‘99’:
DO99 = ‘99029000’

e) Concatenate DO‘87’ and DO‘99’:
M = ‘87110136B83A1FBAC98D89DDDA2235AD
29A8BB99029000’

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = ‘7A59DF409D4FD78C’
- Concatenate padded SSC and M:
N = ‘00000000000000007A59DF409D4FD78C87110136B83A1FBA
C98D89DDDA2235AD29A8BB99029000’
- Compute MAC:
CC = ‘C9338D9966514BBB’

g) Build DO‘8E’:
DO8E = ‘8E08C9338D9966514BBB’

h) Construct response data:
resp_data = ‘87110136B83A1FBAC98D89DDDA2235AD
29A8BB990290008E08C9338D9966514B
BB’

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

B.10.3 Example Using Configuration 3

Static document keying material:

K_{doc} = 'B59B332F43A1AC5311DE3C8CE009340F
57B53210E3A7C092'

Compute Basic Access Keys:

Input: K_{seed} = K_{doc}

Encryption Key (K_{enc}) computation:

1. Concatenate K_{seed} and c (c = 1):
D = 'B59B332F43A1AC5311DE3C8CE009340F
57B53210E3A7C09200000001'

2. Calculate the hash of D:
$H_{SHA\text{-}256}$(D) = '7C0F3DCAE808394AD1111BC35109419A
B399911165D133F493705785CA2B9FF5'

3. Form key:
K_{enc} = '7C0F3DCAE808394AD1111BC35109419A
B399911165D133F4'

Message Authentication Key (K_{mac}) computation:

4. Concatenate K_{seed} and c (c = 2):
D = 'B59B332F43A1AC5311DE3C8CE009340F
57B53210E3A7C09200000002'

5. Calculate the hash of D:
$H_{SHA\text{-}256}$(D) = '5E4DAC4C90CF1832634164A1885CB7DC
1FEDC7DA6658802452DA8116392BC9E9'

6. Form key:
K_{mac} = '5E4DAC4C90CF1832634164A1885CB7DC
1FEDC7DA66588024'

Authentication and Establishment of Session Keys:

IS:

1. Request an 8 byte random challenge from the document's SIC:

Command APDU:

CLA	INS	P1	P2	Le
'00'	'84'	'00'	'00'	'08'

Document SIC:

2. Generate random challenge and return it to IS:
RND.ICC = 'A724E1735E5D5B63'

Response APDU:

Response Data Field	SW1	SW2
RND.ICC	'90'	'00'

IS:

3. Generate an 8-byte random challenge and 24-byte random keying material:
RND.IFD = ‘DBA0881FC039F4B4’
K.IFD = ‘82D950B0EF5C5B36AF3FB362F7431AC1
A8EC1A4581CAF682’

4. Concatenate RND.IFD, RND.ICC and K.IFD; and add padding:
S = ‘DBA0881FC039F4B4A724E1735E5D5B63
82D950B0EF5C5B36AF3FB362F7431AC1
A8EC1A4581CAF6828000000000000000’

5. Encrypt S using AES with key K_{enc}:
E_IFD = ‘C940CF1679F31F2D16BC3CA2245EE97B
ECDDB10DD4FBFB66EFC8120BF6B8B956
E376B2C3AE8BBB3E86D0E4D669AD010F’

6. Compute CMAC over E_IFD with key K_{mac}:
M_IFD = ‘46CC5CC6D483E0CB’

7. Construct command data for MUTUAL AUTHENTICATE and send command to the document’s SIC:
cmd_data = ‘C940CF1679F31F2D16BC3CA2245EE97B
ECDDB10DD4FBFB66EFC8120BF6B8B956
E376B2C3AE8BBB3E86D0E4D669AD010F
46CC5CC6D483E0CB’

Command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘00’	‘84’	‘00’	‘00’	‘38’	cmd_data	‘38’

Document SIC:

8. Generate 16-byte random keying material:
K.ICC = ‘3C2F19D7B42105F25C81B07C78E57BF4
5818DB906CB9360D’

9. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘BEF649675B7D5EC4F3BE031E8FA66135
F0F4C1D5ED73C08F’

10. Derive session keys:
KS_{enc} = ‘0E6F93A7699C8EB5FE3B0CF30BC1EAF8
2796411E137058EA’
KS_{mac} = ‘86D8E8AE5752B1C8268D8566F973BCB6
E1383A083F65181A’

11. Initialize send sequence counter:
SSC = ‘5E5D5B63C039F4B4’

12. Concatenate RND.ICC, RND.IFD and K.ICC; and add padding:
R = ‘A724E1735E5D5B63DBA0881FC039F4B4
3C2F19D7B42105F25C81B07C78E57BF4
5818DB906CB9360D8000000000000000’

13. Encrypt R using AES with key K_{enc}:
E_ICC = ‘CDE9BCFCBB45405554A55500A73010C5
49632B256534DED4DB4B0CAC4A4C3D33
FBF2A3E369C7DC4920C88383F4B8013C’

14. Compute CMAC over E_ICC with key K_{mac}:
M_ICC = ‘F0FECBA285EF8503’

15. Construct response data and send response APDU to the IS:
resp_data = ‘CDE9BCFCBB45405554A55500A73010C5
49632B256534DED4DB4B0CAC4A4C3D33
FBF2A3E369C7DC4920C88383F4B8013C
F0FECBA285EF8503’

Response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

16. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘BEF649675B7D5EC4F3BE031E8FA66135
F0F4C1D5ED73C08F’

17. Derive session keys:
KS_{enc} = ‘0E6F93A7699C8EB5FE3B0CF30BC1EAF8
2796411E137058EA’
KS_{mac} = ‘86D8E8AE5752B1C8268D8566F973BCB6
E1383A083F65181A’

18. Initialize send sequence counter:
SSC = ‘5E5D5B63C039F4B4’

Secure Messaging:

IS:

1. SELECT EF.COM (file identifier = ’01 1E’):

Unprotected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field
‘00’	‘A4’	‘02’	‘00’	‘02’	’01 1E’

a) Mask class byte and pad command header:
cmd_header = ‘0CA4020C800000000000000000000000’

b) Pad data:
p_data = ‘011E8000000000000000000000000000’

c) Encrypt p_data using AES with KS_{enc}:
enc_data = ‘6106F66E07CD676ACAE7E4DCD53140F0’

d) Build DO‘87’:
DO87 = ‘8711016106F66E07CD676ACAE7E4DCD5
3140F0’

e) Concatenate cmd_header and DO87:
M = '0CA4020C800000000000000000000000
8711016106F66E07CD676ACAE7E4DCD5
3140F0'

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = '5E5D5B63C039F4B5'
- Concatenate padded SSC and M:
N = '00000000000000005E5D5B63C039F4B5
0CA4020C800000000000000000000000
8711016106F66E07CD676ACAE7E4DCD5
3140F0'
- Compute MAC:
CC = 'B1E9772E43E3C30E'

g) Build DO'8E':
DO8E = '8E08B1E9772E43E3C30E'

h) Construct command data:
cmd_data = '8711016106F66E07CD676ACAE7E4DCD5
3140F08E08B1E9772E43E3C30E'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
'0C'	'A4'	'02'	'0C'	'1D'	cmd_data	'00'

Document SIC:

2. Set EF.COM as the currently selected file and send affirmative response to IS:

Unprotected response APDU:

SW1	SW2
'90'	'00'

a) Build DO'99':
DO99 = '99029000'

b) Compute CMAC of DO99 with KS_{mac}:
- Increment SSC:
SSC = '5E5D5B63C039F4B6'
- Concatenate padded SSC and DO99:
N = '00000000000000005E5D5B63C039F4B6
99029000'
- Compute MAC:
CC = '1EA61B5F61D6482C'

c) Build DO'8E':
DO8E = '8E081EA61B5F61D6482C'

d) Construct response data:
resp_data = '990290008E081EA61B5F61D6482C'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

3. READ BINARY of the first 4 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
‘0C’	‘B0’	‘00’	‘00’	‘04’

a) Mask class byte and pad command header:
cmd_header = `'0CB00000800000000000000000000000'`

b) Build DO ‘97’:
DO97 = `'970104'`

c) Concatenate cmd_header and DO97:
M = `'0CB00000800000000000000000000000`
`970104'`

d) Compute CMAC of M with KS_{mac}:
- Increment SSC:
  SSC = `'5E5D5B63C039F4B7'`
- Concatenate padded SSC and M:
  N = `'00000000000000005E5D5B63C039F4B7`
  `0CB00000800000000000000000000000`
  `970104'`
- Compute MAC:
  CC = `'92F0C396BC39656B'`

e) Build DO‘8E’:
DO8E = `'8E0892F0C396BC39656B'`

f) Construct command data:
cmd_data = `'9701048E0892F0C396BC39656B'`

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘00’	‘0D’	cmd_data	‘00’

Document SIC:

4. Return 4 bytes of EF.COM starting at offset 0:

data = `'600D5F01'`

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = '600D5F01800000000000000000000000'

b) Encrypt p_data using AES with KS_{enc}:
enc_data = 'CE7B3391BE1A2270A0772F99A90606CA'

c) Build DO‘87’:
DO87 = '871101CE7B3391BE1A2270A0772F99A9
0606CA'

d) Build DO‘99’:
DO99 = '99029000'

e) Concatenate DO‘87’ and DO‘99’:
M = '871101CE7B3391BE1A2270A0772F99A9
0606CA99029000'

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = '5E5D5B63C039F4B8'
- Concatenate padded SSC and M:
N = '00000000000000005E5D5B63C039F4B8
871101CE7B3391BE1A2270A0772F99A9
0606CA99029000'
- Compute MAC:
CC = '9C1B04AD8021F295'

g) Build DO‘8E’:
DO8E = '8E089C1B04AD8021F295'

h) Construct response data:
resp_data = '871101CE7B3391BE1A2270A0772F99A9
0606CA990290008E089C1B04AD8021F2
95'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

5. READ BINARY of the remaining 11 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
‘0C’	‘B0’	‘00’	‘04’	‘0B’

a) Mask class byte and pad command header:
cmd_header = '0CB00004800000000000000000000000'

b) Build DO ‘97’:
DO97 = `'97010B'`

c) Concatenate cmd_header and DO97:
M = `'0CB00004800000000000000000000000`
`97010B'`

d) Compute CMAC of M with KS$_{mac}$:
- Increment SSC:
  SSC = `'5E5D5B63C039F4B9'`
- Concatenate padded SSC and M:
  N = `'00000000000000005E5D5B63C039F4B9`
  `0CB00004800000000000000000000000`
  `97010B'`
- Compute MAC:
  CC = `'0228369B4D5B8D1B'`

e) Build DO‘8E’:
DO8E = `'8E080228369B4D5B8D1B'`

f) Construct command data:
cmd_data = `'97010B8E080228369B4D5B8D1B'`

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘04’	‘0D’	cmd_data	‘00’

Document SIC:

6. Return 11 bytes of EF.COM starting at offset 4:

data = `'04303130305C04616B6567'`

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = `'04303130305C04616B65678000000000'`

b) Encrypt p_data using AES with KS$_{enc}$:
enc_data = `'D3B264B934AB8E7BDA7DCEEEDA79AA39'`

c) Build DO‘87’:
DO87 = `'871101D3B264B934AB8E7BDA7DCEEEDA`
`79AA39'`

d) Build DO‘99’:
DO99 = `'99029000'`

e) Concatenate DO‘87’ and DO‘99’:
M = `'871101D3B264B934AB8E7BDA7DCEEEDA`
`79AA3999029000'`

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
  SSC = '5E5D5B63C039F4BA'
- Concatenate padded SSC and M:
  N = '00000000000000005E5D5B63C039F4BA
  871101D3B264B934AB8E7BDA7DCEEEDA
  79AA3999029000'
- Compute MAC:
  CC = 'F89F71E004437C6B'

g) Build DO'8E':
DO8E = '8E08F89F71E004437C6B'

h) Construct response data:
resp_data = '871101D3B264B934AB8E7BDA7DCEEEDA
79AA39990290008E08F89F71E004437C
6B'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	'90'	'00'

B.10.4 Example Using Configuration 4

Static document keying material:
K_{doc} = '8D2F25C266CC8068F99391BF0F5CCB87
6B5F5DDB004D0E5C8BCD1D3ACF2FDADA'

Compute Basic Access Keys:

Input: K_{seed} = K_{doc}

Encryption Key (K_{enc}) computation:

1. Concatenate K_{seed} and c (c = 1):
D = '8D2F25C266CC8068F99391BF0F5CCB87
6B5F5DDB004D0E5C8BCD1D3ACF2FDADA
00000001'

2. Calculate the hash of D:
$H_{SHA-256}(D)$ = '45C1CAF8AD80EA1D16DB495CEBEE310A
E52E08E20985C60A3652BED1689DBAE8'

3. Form key:
K_{enc} = '45C1CAF8AD80EA1D16DB495CEBEE310A
E52E08E20985C60A3652BED1689DBAE8'

Message Authentication Key (K_{mac}) computation:

4. Concatenate K_{seed} and c (c = 2):
D = '8D2F25C266CC8068F99391BF0F5CCB87
6B5F5DDB004D0E5C8BCD1D3ACF2FDADA
00000002'

5. Calculate the hash of D:
$H_{SHA-256}(D)$ = '61F803835724C6B5F1364C8FACF15993
4BF5ACE6E9EB001AB801A3B6103BA2F2'

6. Form key:
K_{mac} = '61F803835724C6B5F1364C8FACF15993
4BF5ACE6E9EB001AB801A3B6103BA2F2'

Authentication and Establishment of Session Keys:

IS:

1. Request an 8 byte random challenge from the document's SIC:

Command APDU:

CLA	INS	P1	P2	Le
'00'	'84'	'00'	'00'	'08'

Document SIC:

2. Generate random challenge and return it to IS:
RND.ICC = 'E880AAE12EB3A5FB'

Response APDU:

Response Data Field	SW1	SW2
RND.ICC	'90'	'00'

IS:

3. Generate an 8-byte random challenge and 24-byte random keying material:
RND.IFD = 'B962840EFBFE80C9'
K.IFD = '1D05B3E621AC7BB4786AC1657D0C4C11
58875525EB21659D905674FCAFF94421'

4. Concatenate RND.IFD, RND.ICC and K.IFD:
S = 'B962840EFBFE80C9E880AAE12EB3A5FB
1D05B3E621AC7BB4786AC1657D0C4C11
58875525EB21659D905674FCAFF94421'

5. Encrypt S using AES with key K_{enc}:
E_IFD = '8466CB488BCDF6AEE13761878CAF1642
A22D6BAE9BD20B88E0A584AB7DA4D876
45EE71C4C8A887FFA00CD0F61BACA327'

6. Compute CMAC over E_IFD with key K_{mac}:
M_IFD = '4ECAD1955CD9987F'

7. Construct command data for MUTUAL AUTHENTICATE and send command to the document's SIC:
cmd_data = '8466CB488BCDF6AEE13761878CAF1642
A22D6BAE9BD20B88E0A584AB7DA4D876
45EE71C4C8A887FFA00CD0F61BACA327
4ECAD1955CD9987F'

Command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘00’	‘84’	‘00’	‘00’	‘38’	cmd_data	‘38’

Document SIC:

8. Generate 16-byte random keying material:
K.ICC = ‘56F1510FDCC2B01787E80D2D5E340840
20C93698AF4599C9B9B7D68EB2E958B7’

9. Calculate XOR of K.IFD and K.ICC:
K_{seed} = ‘4BF4E2E9FD6ECBA3FF82CC4823384451
784E63BD4464FC5429E1A2721D101C96’

10. Derive session keys:
KS_{enc} = ‘60BDD38EE1B27EEAC7AF9907889F2E04
74C7AF231C71705BB2A84BF87BA825FF’
KS_{mac} = ‘978E2D4BFC62716966B215A28980ED04
1756A53EBC56AE7CE9F8341167210C33’

11. Initialize send sequence counter:
SSC = ‘2EB3A5FBFBFE80C9’

12. Concatenate RND.ICC, RND.IFD and K.ICC; and add padding:
R = ‘E880AAE12EB3A5FBB962840EFBFE80C9
56F1510FDCC2B01787E80D2D5E340840
20C93698AF4599C9B9B7D68EB2E958B7’

13. Encrypt R using AES with key K_{enc}:
E_ICC = ‘F58D48343CD71F4B4A6156BA5585B179
468EBD1D18BE6F66ABA3ACD87B8D6D20
D29AD5F700EB62F36E2E292DA0DC705B’

14. Compute CMAC over E_ICC with key K_{mac}:
M_ICC = ‘31A96B74E932C35E’

15. Construct response data and send response APDU to the IS:
resp_data = ‘F58D48343CD71F4B4A6156BA5585B179
468EBD1D18BE6F66ABA3ACD87B8D6D20
D29AD5F700EB62F36E2E292DA0DC705B
31A96B74E932C35E’

Response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

IS:

16. Calculate XOR of K.IFD and K.ICC:
K_{seed} = `'4BF4E2E9FD6ECBA3FF82CC4823384451 784E63BD4464FC5429E1A2721D101C96'`

17. Derive session keys:
KS_{enc} = `'60BDD38EE1B27EEAC7AF9907889F2E04 74C7AF231C71705BB2A84BF87BA825FF'`
KS_{mac} = `'978E2D4BFC62716966B215A28980ED04 1756A53EBC56AE7CE9F8341167210C33'`

18. Initialize send sequence counter:
SSC = `'2EB3A5FBFBFE80C9'`

Secure Messaging:

IS:

1. SELECT EF.COM (file identifier = ’01 1E’):

Unprotected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field
‘00’	‘A4’	‘02’	‘00’	‘02’	’01 1E’

a) Mask class byte and pad command header:
cmd_header = `'0CA4020C800000000000000000000000'`

b) Pad data:
p_data = `'011E8000000000000000000000000000'`

c) Encrypt p_data using AES with KS_{enc}:
enc_data = `'C74A8B66F7EA68098B8B4F1E51F9BE58'`

d) Build DO‘87’:
DO87 = `'871101C74A8B66F7EA68098B8B4F1E51 F9BE58'`

e) Concatenate cmd_header and DO87:
M = `'0CA4020C800000000000000000000000 871101C74A8B66F7EA68098B8B4F1E51 F9BE58'`

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = `'2EB3A5FBFBFE80CA'`
- Concatenate padded SSC and M:
N = `'00000000000000002EB3A5FBFBFE80CA 0CA4020C800000000000000000000000 871101C74A8B66F7EA68098B8B4F1E51 F9BE58'`
- Compute MAC:
CC = `'962F4EB48D921CB2'`

g) Build DO‘8E’:
DO8E = `'8E08962F4EB48D921CB2'`

h) Construct command data:
cmd_data = '871101C74A8B66F7EA68098B8B4F1E51
F9BE588E08962F4EB48D921CB2'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
'0C'	'A4'	'02'	'0C'	'1D'	cmd_data	'00'

Document SIC:

2. Set EF.COM as the currently selected file and send affirmative response to IS:

Unprotected response APDU:

SW1	SW2
'90'	'00'

a) Build DO'99':
DO99 = '99029000'

b) Compute CMAC of DO99 with KS_{mac}:
- Increment SSC:
SSC = '2EB3A5FBFBFE80CB'
- Concatenate padded SSC and DO99:
N = '00000000000000002EB3A5FBFBFE80CB
99029000'
- Compute MAC:
CC = '4551330E6A6ADA45'

c) Build DO'8E':
DO8E = '8E084551330E6A6ADA45'

d) Construct response data:
resp_data = '990290008E084551330E6A6ADA45'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	'90'	'00'

IS:

3. READ BINARY of the first 4 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
'0C'	'B0'	'00'	'00'	'04'

a) Mask class byte and pad command header:
cmd_header = '0CB00000800000000000000000000000'

b) Build DO '97':
DO97 = '970104'

c) Concatenate cmd_header and DO97:
M = '0CB00000800000000000000000000000
970104'

d) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = '2EB3A5FBFBFE80CC'
- Concatenate padded SSC and M:
N = '00000000000000002EB3A5FBFBFE80CC
0CB00000800000000000000000000000
970104'
- Compute MAC:
CC = '7963441B61D9407A'

e) Build DO'8E':
DO8E = '8E087963441B61D9407A'

f) Construct command data:
cmd_data = '9701048E087963441B61D9407A'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
'0C'	'B0'	'00'	'00'	'0D'	cmd_data	'00'

Document SIC:

4. Return 4 bytes of EF.COM starting at offset 0:

data = '600D5F01'

Unprotected response APDU:

Response Data Field	SW1	SW2
data	'90'	'00'

a) Pad data:
p_data = '600D5F0180000000000000000000000000'

b) Encrypt p_data using AES with KS_{enc}:
enc_data = 'DBBA6E8C7C837A22FD94F7F3455A64AE'

c) Build DO'87':
DO87 = '871101DBBA6E8C7C837A22FD94F7F345
5A64AE'

d) Build DO'99':
DO99 = '99029000'

e) Concatenate DO'87' and DO'99':
M = '871101DBBA6E8C7C837A22FD94F7F345
5A64AE99029000'

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
  SSC = '2EB3A5FBFBFE80CD'
- Concatenate padded SSC and M:
  N = '00000000000000002EB3A5FBFBFE80CD
  871101DBBA6E8C7C837A22FD94F7F345
  5A64AE99029000'
- Compute MAC:
  CC = '1A4DBCFB2479EC90'

g) Build DO'8E':
DO8E = '8E081A4DBCFB2479EC90'

h) Construct response data:
resp_data = '871101DBBA6E8C7C837A22FD94F7F345
5A64AE990290008E081A4DBCFB2479EC
90'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	'90'	'00'

IS:

5. READ BINARY of the remaining 11 bytes:

Unprotected command APDU:

CLA	INS	P1	P2	Le
'0C'	'B0'	'00'	'04'	'0B'

a) Mask class byte and pad command header:
cmd_header = '0CB00004800000000000000000000000'

b) Build DO '97':
DO97 = '97010B'

c) Concatenate cmd_header and DO97:
M = '0CB00004800000000000000000000000
97010B'

d) Compute CMAC of M with KS_{mac}:
- Increment SSC:
  SSC = '2EB3A5FBFBFE80CE'
- Concatenate padded SSC and M:
  N = '00000000000000002EB3A5FBFBFE80CE
  0CB00004800000000000000000000000
  97010B'
- Compute MAC:
  CC = 'D69950D5A996A0EE'

e) Build DO'8E':
DO8E = '8E08D69950D5A996A0EE'

f) Construct command data:
cmd_data = '97010B8E08D69950D5A996A0EE'

Protected command APDU:

CLA	INS	P1	P2	Lc	Command Data Field	Le
‘0C’	‘B0’	‘00’	‘04’	‘0D’	cmd_data	‘00’

Document SIC:

6. Return 11 bytes of EF.COM starting at offset 4:

data = '04303130305C04616B6567'

Unprotected response APDU:

Response Data Field	SW1	SW2
data	‘90’	‘00’

a) Pad data:
p_data = '04303130305C04616B65678000000000'

b) Encrypt p_data using AES with KS_{enc}:
enc_data = '9D4B6092AEEC6868505D1CFDC112EA0D'

c) Build DO‘87’:
DO87 = '8711019D4B6092AEEC6868505D1CFDC1
12EA0D'

d) Build DO‘99’:
DO99 = '99029000'

e) Concatenate DO‘87’ and DO‘99’:
M = '8711019D4B6092AEEC6868505D1CFDC1
12EA0D99029000'

f) Compute CMAC of M with KS_{mac}:
- Increment SSC:
SSC = '2EB3A5FBFBFE80CF'
- Concatenate padded SSC and M:
N = '00000000000000002EB3A5FBFBFE80CF
8711019D4B6092AEEC6868505D1CFDC1
12EA0D99029000'
- Compute MAC:
CC = '7D36C96B71E02C85'

g) Build DO‘8E’:
DO8E = '8E087D36C96B71E02C85'

h) Construct response data:
resp_data = '8711019D4B6092AEEC6868505D1CFDC1
12EA0D990290008E087D36C96B71E02C
85'

Protected response APDU:

Response Data Field	SW1	SW2
resp_data	‘90’	‘00’

Annex C
(normative)

Extended access protection

C.1 Introduction

This annex describes a protocol for conditional access to an application that stores data in data groups according to a LDS.

The EAP protocol described herein is derived from and largely compatible with the Extended Access Control proposed in TR-03110 (see [2] in the Bibliography). It provides the following features:

a) Chip authentication: Originality and Confidentiality.

b) Terminal authentication: Conditional Access for up to 24 data groups.

NOTE Currently this Standard only uses 16 data groups (see ASN.1 sequence in 8.1.5.1).

The EAP specification is intended to be referenced by other specifications. As such, some aspects of the protocol shall be defined by those referring specifications, as indicated within this part of ISO/IEC 18013. Editors of such specifications should take special note of C.8.

C.2 Card-verifiable certificate format

C.2.1 General

Due to computational restrictions, certificates intended to be verified by an SIC shall be in a self-descriptive card-verifiable format according to ISO/IEC 7816-8:2004 as described in this clause.

Table C.1 shows the certificate format. All data objects are mandatory and shall be included in the fixed order reflected in Table C.1.

Table C.1 — Self-descriptive card-verifiable certificate

<table>
<tr><th>Tag</th><th>Length</th><th colspan="5">Value</th></tr>
<tr><td rowspan="12">'7F4E'</td><td rowspan="12">var.</td><td colspan="5">Certificate body</td></tr>
<tr><th>Tag</th><th>Length</th><th colspan="3">Value</th></tr>
<tr><td>'5F29'</td><td>1</td><td colspan="3">Certificate format version</td></tr>
<tr><td>'42'</td><td>var.
max 16 bytes</td><td colspan="3">Authority key identifier</td></tr>
<tr><td>'7F49'</td><td>var.</td><td colspan="3">Public key</td></tr>
<tr><td>'5F20'</td><td>var.
max 16 bytes</td><td colspan="3">Subject key identifier</td></tr>
<tr><td rowspan="4">'7F4C'</td><td rowspan="4">15</td><td colspan="3">Certificate holder authorization</td></tr>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'06'</td><td>7</td><td>Object identifier</td></tr>
<tr><td>'53'</td><td>4</td><td>Authorization</td></tr>
<tr><td>'5F25'</td><td>6</td><td colspan="3">Certificate effective date</td></tr>
<tr><td>'5F24'</td><td>6</td><td colspan="3">Certificate expiration date</td></tr>
<tr><td>'5F37'</td><td>var.</td><td colspan="5">Signature</td></tr>
</table>

NOTE For management purposes, the card-verifiable certificate may be wrapped in the card-holder certificate template (tag '7F21') as described in ISO/IEC 7816-8:2004. The tag '7F21' is never transmitted to the SIC.

C.2.1.1 Certificate format version

This field specifies the version of the certificate format. It shall be set to '00' (version 1).

C.2.1.2 Authority key identifier

This field shall be set to the value of the issuer's subject key identifier.

C.2.1.3 Public key

EAP supports both RSA and ECDSA public keys. An object identifier specifies the key type and format certificate. It also specifies the signature algorithm used by the holder; either in issued certificates, or during terminal authentication. All certificates in a chain that are not TRCA certificates shall use the same algorithm as their issuer, i.e. only the TRCA is allowed to switch to a different algorithm. Table C.2 lists the supported algorithms and corresponding OIDs.

Table C.2 — Supported algorithms

Key Type	OID	Signature algorithm
RSA	`id-pk-RSA-PKCS1-v1_5-SHA1`	RSASSA with PKCS#1v1.5 padding and SHA-1
	`id-pk-RSA-PKCS1-v1_5-SHA256`	RSASSA with PKCS#1v1.5 padding and SHA-256
	`id-pk-RSA-PSS-SHA1`	RSASSA-PSS with SHA-1
	`id-pk-RSA-PSS-SHA256`	RSASSA-PSS with SHA-256
EC	`id-pk-ECDSA-SHA1`	ECDSA with SHA-1
	`id-pk-ECDSA-SHA224`	ECDSA with SHA-224
	`id-pk-ECDSA-SHA256`	ECDSA with SHA-256

ECDSA signatures shall be produced as DER-encoded sequence of the two integers *r* and *s*, i.e. `SEQUENCE { r INTEGER, s INTEGER }`.

In case of RSASSA-PSS, the following parameters shall be used during signature generation and verification:

- Hash algorithm: Selected according to Table C.2,
- Mask generation algorithm: MGF1 using the selected hash algorithm,
- Salt length: The output length of the selected hash algorithm,
- Trailer field: 'BC'.

The respective OIDs are defined as shown below:

```
id-pk-RSA-PKCS1-v1_5-SHA1 OBJECT IDENTIFIER :: = {
   iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 20
}

id-pk-RSA-PKCS1-v1_5-SHA256 OBJECT IDENTIFIER :: = {
   iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 21
}

id-pk-RSA-PSS-SHA1 OBJECT IDENTIFIER ::= {
iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 22
}

id-pk-RSA-PSS-SHA256 OBJECT IDENTIFIER ::= {
iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 23
}

id-pk-ECDSA-SHA1 OBJECT IDENTIFIER :: = {
   iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 30
}

id-pk-ECDSA-SHA224 OBJECT IDENTIFIER :: = {
   iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 31
}

id-pk-ECDSA-SHA256 OBJECT IDENTIFIER :: = {
   iso(1) standard(0) driving-licence(18013) part-3(3) cv-certs(1) 32
}
```

C.2.1.3.1 RSA public key

An RSA public key consists of three mandatory data objects in fixed order as laid out in Table C.3.

Table C.3 — RSA public key

Tag	Length	Value		
‘7F49’	var.	Public key details		
		Tag	**Length**	**Value**
		‘06’	var.	BER-encoded object identifier from Table C.2
		‘81’	var.	Composite modulus *n*
		‘82’	var.	Public exponent *e*

The length of the modulus *n* shall be 1024, 1280, 1536, 2048, or 3072 bits; and shall be equal to or smaller than the size of the issuer’s key’s modulus.

C.2.1.3.2 EC public key

An EC public key consists of two mandatory and six optional data objects in fixed order as laid out in Table C.4. The six optional fields contain domain parameters. They shall either all be present or absent. If the domain parameters are omitted, they are assumed to be identical to the parameters of the issuer’s public key. Certificates that are not TRCA certificates shall not include domain parameters.

All values shall be produced in octet string representation as specified in X9.62.

Table C.4 — EC public key

Tag	Length	Value		
‘7F49’	var.	Public key details		
		Tag	**Length**	**Value**
		‘06’	var.	BER-encoded object identifier from Table C.2.
		‘81’	var.	Prime modulus *p* (optional)
		‘82’	var.	First coefficient *a* (optional)
		‘83’	var.	Second coefficient *b* (optional)
		‘84’	var.	Base point *G* (optional)
		‘85’	var.	Order of base point *r* (optional)
		‘86’	var.	Public point *Y*
		‘87’	var.	Co-factor *f* (optional)

C.2.1.4 Subject key identifier

This field shall contain a binary string no larger that 128-bit (16 bytes) that uniquely identifies the holder’s public key. It should be derived from the public key or a method that generates unique values.

C.2.1.5 Certificate holder authorization

This constructed field contains an object identifier and an authorization field. The object identifier specifies the format and rules for the evaluation of the authorization level and shall be set to `id-ar-Terminal`.

```
id-ar-Terminal OBJECT IDENTIFIER ::= {
   iso(1) standard(0) driving-licence(18013) part-3(3) eap(3) 1
}
```

The purpose and format of the authorization field is specified in C.2.2.

C.2.1.6 Certificate effective date

This field shall be set to the date from which the certificate is valid (0:00 UTC), encoded as YYMMDD in unpacked BCD. The year YY is encoded in two digits and shall be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

C.2.1.7 Certificate expiration date

This field shall be set to the date after which the certificate expires (0:00 UTC), encoded as YYMMDD in unpacked BCD. The year YY is encoded in two digits and shall be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

C.2.1.8 Signature

This field shall contain the digital signature over the entire certificate body data object (including tag and length).

C.2.2 Certificate authorization

C.2.2.1 Overview

Certificate authorization describes the rights asserted to the holder of the private key corresponding to the certificate.

C.2.2.2 Relative authorization

The *relative authorization* of a certificate holder is encoded in the 4 byte Authorization field of the card-verifiable certificate. It consists of the role byte (see Table C.5) and three access rights bytes (see Table C.6). These values are relative to the authorization of all previous certificates in a chain. If the OID of the certificate's authorization field is not `id-ar-Terminal`, the relative authorization shall be '00 00 00 00'.

Table C.5 — Authorization role byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	x	x	-	-	-	-	**Entity type**
-	-	1	-	-	-	-	-	— Trust root
-	-	-	1	-	-	-	-	— Authoritative time source
-	-	-	-	x	x	x	x	**Path length constraint** Maximum number of certificates that may follow this certificate in a certificate chain. A value of 1111b (0xf) indicates an unlimited number of certificates.
x	x	-	-	-	-	-	-	Reserved for future use (set to zero)

Table C.6 — Authorization access rights bytes

b24	b23	b22	…	b4	b3	b2	b1	Meaning
x	x	x	…	x	x	x	x	**Data group read access**
1	-	-	…	-	-	-	-	— Grant read access to DG24
-	1	-	…	-	-	-	-	— Grant read access to DG23
-	-	1	…	-	-	-	-	— Grant read access to DG22
-	-	-	…	-	-	-	-	— Grant read access to DG21 - DG5
-	-	-	…	1	-	-	-	— Grant read access to DG4
-	-	-	…	-	1	-	-	— Grant read access to DG3
-	-	-	…	-	-	1	-	— Grant read access to DG2
-	-	-	…	-	-	-	1	— Grant read access to DG1

NOTE The access rights field is always 24 bits long, even if the application contains fewer data groups (only 16 data groups are specified in 8.1.5.1). The access rights bits of non-existing data groups shall be set to zero.

C.2.2.3 Effective authorization

The *effective authorization* of a certificate holder is calculated by applying the relative authorization of the current certificate to the effective authorization of the previous certificate in the chain in the following way:

a) for the entity type, a bit-wise AND operation;

b) for the path length constraint, according to Table C.7;

c) for the access rights bytes, a bit-wise AND operation.

The effective authorization of the application's trust root certificate is equal to its relative authorization.

Table C.7 — Path length constraint

Effective Authorization Parent certificate	Relative Authorization Current certificate	Effective Authorization Current certificate
0x0f (1111b)	0x0f (1111b)	0x0f (1111b)
0x0f (1111b)	*m*	*m*
n	0x0f (1111b)	*n*-1
n	*m*	min(*n*-1, *m*)

NOTE The path length constraint allows a flexible PKI hierarchies. However, a fixed three tier hierarchy as prescribed in TR-03110 can be mapped to the path length constraint using the settings in Table C.8:

Table C.8 — Three tier hierarchy mapped to path length constraint

	Trust root	Authoritative time source	Path length constraint
CVCA	X	X	unlimited
DV (domestic)		X	1
DV (foreign)			1
IS			0

C.3 Chip authentication

C.3.1 Overview

Chip authentication is an ephemeral-static key agreement protocol that provides strong session keys for secure messaging. If the static key of the SIC is properly authenticated, e.g. with a digital signature from a trusted entity, it also provides authentication of the SIC to the terminal. In this case it may be used as an alternative to active authentication.

C.3.2 Specification

The IS and the SIC application perform the following steps during chip authentication:

a) The IS obtains SIC's key agreement public key (including domain parameters) and authenticates it.

b) The IS generates an ephemeral key agreement key pair and sends the public component to the SIC.

c) The SIC and the terminal both calculate a shared secret based on the key agreement protocol in use. This shared secret is then used to derive new session keys. Subsequent secure messaging is protected with the new session keys.

If subsequent secure messaging is successful, the SIC implicitly proves knowledge of the private key corresponding to the static key agreement public key.

C.3.3 Supported key agreement protocols

The key agreement protocol is chosen by the SIC, as it has a static key. Issuers may chose from the protocols listed in Table C.9.

Table C.9 — Key agreement protocols

Scheme	Terminal Public Key Format	Algorithm Identifier
DH (as specified in RFC 2631)	The number g^b mod p in big endian format and right aligned	`dhpublicnumber`
Key agreement of ElGamal type (as specified in ISO/IEC 11770-3:2008, D.3)	Octet string representation of the public point in uncompressed form according to X9.62	`id-ecPublicKey`

The Terminal Public Key Format is used when sending the terminal's ephemeral key to the SIC.

C.3.4 On-card storage of keys

The SIC's static key agreement key pair shall be stored on the SIC. The private key shall be stored in secure memory, such that it cannot be retrieved from the outside. The public key shall be stored in a data group as indicated by referring specifications. IS's can obtain the data group number from the security mechanism indicator (see C.6).

The data group shall contain a DER-encoded `SecurityInfos` structure:

```
SecurityInfos ::= SET OF SecurityInfo

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

The `SecurityInfos` set shall contain at least one `SecurityInfo` structure that defines a chip authentication public key as defined below:

The `protocol` field shall be set to `id-ICAuth`.

```
id-ICAuth OBJECT IDENTIFIER ::= {
   iso(1) standard(0) driving-licence(18013) part-3(3) eap(3) 1
}
```

The `requiredData` field shall be of type `ICAuthPublicKeyInfo`.

```
ICAuthPublicKeyInfo ::= SEQUENCE {
   keyIdentifier INTEGER (0..127) OPTIONAL,
   icAuthPublicKey SubjectPublicKeyInfo
}
```

The public key shall be stored in the `SubjectPublicKeyInfo` structure (see RFC 3280) using the algorithm identifiers listed in Table C.9 according to RFC 3279, with all domain parameters explicitly included. The optional `keyIdentifier` field shall be present if multiple chip authentication public keys are supplied within the `SecurityInfos` structure.

The `optionalData` field shall be absent.

C.3.5 Card command: MSE:SET KAT command

The MSE:SET KAT command is used to implement chip authentication, i.e. to establish new session keys for secure messaging using a key agreement protocol.

Upon reception of the terminal's key agreement public key, the SIC may optionally choose to validate it. The SIC then calculates the shared secret using the key agreement protocol chosen by the issuer. This shared secret is then used as K_{seed} to derive new session keys according to B.4 using the hash function defined by the BAP configuration that is referenced by the OID indicated by the secure messaging configuration as indicated in the security mechanism indicator (see C.8).

These session keys are used to protect subsequent commands. If secure messaging is already in progress, the response to this command is protected with the old session keys, after which they are replaced and secure messaging is restarted according to the secure messaging configuration as indicated in the security mechanism indicator (see C.6). If secure messaging was not yet in progress, it is now started according to the secure messaging configuration as indicated in the security mechanism indicator (see C.6). In either case, the Send Sequence Counter shall be set to zero (SSC = 0).

Secure messaging is only restarted if the command was successful. Should an error condition occur whilst secure messaging was active, the previously established session keys remain valid.

Table C.10 — Command APDU: MSE:SET KAT

CLA	As defined in ISO/IEC 7816-4:2005	
INS	0x22	MSE
P1	0x41	Set for computation
P2	0xA6	KAT
L_c field	Length of subsequent data field.	
Data field	0x91 0x84	Random number DO containing the terminal key agreement public key (see Table C.9) Reference DO containing the identifier of the key to be used (optional)
L_e field	Absent	

Table C.11 — Response APDU: MSE:SET KAT

Data field	Absent	
SW1-SW2	'9000' '6A80' Other	Normal processing Incorrect parameters in the command data field. Validation of terminal key agreement public key failed. Operating system dependent error

C.4 Terminal authentication

C.4.1 General

Terminal authentication is a challenge-response protocol that provides authentication of the terminal to the SIC application.

C.4.2 Specification

The IS and the SIC application perform the following steps during terminal authentication:

a) The IS sends a certificate chain to the SIC.

b) The SIC verifies the certificate chain.

c) The IS requests a challenge from the SIC.

d) The SIC presents a challenge to the IS.

e) The IS signs the challenge and sends it to the SIC.

f) The SIC verifies the signature and grants access if successful.

C.4.3 Terminal authentication concepts

C.4.3.1 Trust root certificate

The SIC has a single trust root certificate, which is issued by its TRCA. Certificate chains presented to the SIC application for terminal authentication shall start immediately below this trust root. ISs can obtain the identifier of the current trust root from the EAP security mechanism indicator (see C.6).

C.4.3.2 Alternate trust root certificate

During certificate roll-over period, there may be an alternate trust root certificate.

C.4.3.3 Currently active certificate verification key

The currently active certificate verification key is used for verifying certificates. On application selection, this key is set to the trust root certificate's public key. After successful verification of a certificate, it is set to the public key of that certificate.

C.4.3.4 Current on-card date

The current on-card date is used to verify certificate validity. Since SICs do not have a time clock, this on-card date is updated sporadically whenever a more current effective date is encountered in a certificate signed by an authoritative time source entity.

C.4.3.5 SIC identifier

The SIC identifier is used in the signature generation and verification of the challenge-response to guarantee that the message was intended for that particular SIC. Referring specifications shall indicate what data is used as SIC identifier (e.g. document number).

C.4.4 Card commands

C.4.4.1 MSE:SET DST

The MSE:SET DST command can optionally be used to select a specific certificate verification key before using the PSO: VERIFY CERTIFICATE command.. Only the currently active certificate verification key, or trust roots listed in the security mechanism indicator may be selected. After successful selection, the currently active certificate verification key is set to the newly selected certificate's public key. If selection fails, the card takes no further action.

Table C.12 — Command APDU: MSE: SET DST

CLA	As defined in ISO/IEC 7816-4:2005
INS	0x22 MSE
P1	0x81 Set for verification
P2	0xB6 DST
L_c field	Length of subsequent data field
Data field	‘83’ Reference DO containing a reference to a public key (Subject key identifier)
L_e field	Absent

Table C.13 — Response APDU: MSE:SET DST

Data field	Absent	
SW1-SW2	'9000'	Normal processing
	'6A88'	Reference data not found. Requested public key not available for selection.
	Other	Operating system dependent error

NOTE Selecting the currently active certificate verification key has no effect, since it is already implicitly selected. This option is included for compatibility with terminals that always use the explicit selection mechanism.

C.4.4.2 PSO: VERIFY CERTIFICATE

The PSO: VERIFY CERTIFICATE command verifies a card-verifiable certificate. Verification is successful if and only if:

a) the effective authorization of the parent certificate indicates a path length constraint value that is > 0,

b) the certificate’s expiration date is after the current on-card date, and

c) the certificate’s signature verifies using the currently active certificate verification key.

After successful verification, the card application takes the following steps:

a) The currently active certificate verification key is set to the public key contained in the certificate.

b) The effective authorization of the holder is calculated as described in C.2.2.3.

c) If the effective authorization of the holder indicates a trust root, and the certificate effective date is after the effective date of the current trust root, the trust root’s public key is permanently updated to the public key contained in this certificate. While the old trust root has not yet expired (roll-over period), it shall be retained as alternate trust root until it expires. There shall not be more than one alternate trust root at a time.

d) If the effective authorization of the parent certificate indicates an authoritative time source, and the certificate effective date of this certificate is after the current on-card date, the on-card date is updated to match the certificate effective date of this certificate.

If verification fails, the currently active certificate verification key is reset to the current trust root’s public key.

Table C.14 — Command APDU: PSO: VERIFY CERTIFICATE

CLA	As defined in ISO/IEC 7816-4:2005
INS	0x2A PSO
P1	0x00
P2	0xBE Verify Certificate
L_c field	Length of subsequent data field
Data field	Card-verifiable certificate (see C.2) ‘7F4E’ Certificate body ‘5F37’ Corresponding signature
L_e field	Absent

Table C.15 — Response APDU: PSO: VERIFY CERTIFICATE

Data field	Absent
SW1-SW2	'9000' Normal processing '6300' Verification failed. Certificate verification failed. Other Operating system dependent error

C.4.4.3 MSE:SET AT

The MSE:SET AT command can optionally be used to select a specific certificate verification key before using the GET CHALLENGE/EXTERNAL AUTHENTICATE command sequence. Only the currently active certificate verification key may be selected. If selection fails, the card takes no further action.

Table C.16 — Command APDU: MSE: SET AT

CLA	As defined in ISO/IEC 7816-4:2005
INS	0x22 MSE
P1	0x81 Set for verification
P2	0xA4 AT
L_c field	Length of subsequent data field
Data field	‘83’ Reference DO containing a reference to a public key (Subject key identifier)
L_e field	Absent

Table C.17 — Response APDU: MSE:SET DST

Data field	Absent
SW1-SW2	'9000' Normal processing '6A88' Reference data not found. Requested public key not available for selection. Other Operating system dependent error

NOTE This command has no effect, since the only public key that can be selected is always already implicitly selected. It is included for compatibility with terminals that always use the explicit selection mechanism.

C.4.4.4 GET CHALLENGE

This command (or a command with the same INS code) may also be used by other security mechanisms. Referring specifications shall describe how these cases, if any, are distinguished.

The GET CHALLENGE command receives a (true) random challenge from the card to be signed in the subsequent EXTERNAL AUTHENTICATE command (see B.9.1 for command specification).

C.4.4.5 EXTERNAL AUTHENTICATE

This command (or a command with the same INS code) may also be used by other security mechanisms. Referring specifications shall describe how these cases, if any, are distinguished.

The EXTERNAL AUTHENTICATE command is used to submit a signature of a challenge to the card for verification with the currently active certificate verification key. The terminal can thereby prove knowledge of the corresponding private key to the card.

The signature shall be calculated and formatted according to the algorithm indicated by the OID (see C.2.1.3) in the currently active certificate over the concatenation of the SIC identifier and the challenge. The challenge shall be obtained from the card using the GET CHALLENGE command immediately before issuing EXTERNAL AUTHENTICATE.

Referring specifications may prescribe a certain security level before this command can be issued. For example, strong secure messaging established using chip authentication may be required.

If the signature verification is successful, access to data groups is granted based on the effective authorization of the current certificate. Access to data groups that referring specifications deem readable without terminal authentication is always granted, even if the access bits for those data groups are not set.

Table C.18 — Command APDU: EXTERNAL AUTHENTICATE

CLA	As defined in ISO/IEC 7816-4:2005	
INS	0x82	EXTERNAL AUTHENTICATE
P1	0x00	Cryptographic algorithm implicitly known
P2	0x00	Key implicitly known
L_c field	Length of subsequent data field	
Data field	Signature	
L_e field	Absent	

Table C.19 — Response APDU: EXTERNAL AUTHENTICATE

Data field	Absent	
SW1-SW2	'9000'	Normal processing
	'6300'	Verification failed. Signature verification failed.
	'6982'	Security status not satisfied. Insufficient security level.
	Other	Operating system dependent error

C.5 Extended length APDUs and command chaining

Some commands used by EAP (notably MSE:SET KAT, PSO:VERIFY CERTIFICATE, and EXTERNAL AUTHENTICATE) may require the transmission of more than 255 bytes to the SIC. In such a case, the terminal shall use extended length APDUs if the SIC indicates this capability in the historical bytes (ATR or EF.ATR). If the SIC does not indicate this capability, command chaining may be used.