

# **E-ID**

# 1 Table of Contents

<b>1</b>	<b>Table of Contents .....</b>	<b>2</b>
<b>2</b>	<b>The use and future-use of the eID .....</b>	<b>4</b>
2.1	Objective of the eID .....	4
2.2	Visible data .....	4
2.3	Electronic data .....	4
2.4	What can be done .....	5
2.5	How is it done .....	5
2.6	Microsoft is interested in our small country's idea .....	5
2.7	Big Brother .....	5
<b>3</b>	<b>Middleware .....</b>	<b>6</b>
3.1	Introduction .....	6
3.2	Interfaces .....	7
3.2.1	The Crypto API interface .....	7
3.2.2	The PKCS#11 interface .....	9
<b>4</b>	<b>PIN / PUK Security .....</b>	<b>11</b>
4.1	Readers with PIN pad .....	11
4.2	Software compatibility with the middleware .....	12
4.3	PC/SC drivers .....	12
4.4	Integration with the middleware .....	12
4.4.1	Parameters and return code .....	13
4.4.2	Display .....	15
<b>5</b>	<b>Executive Summary .....</b>	<b>15</b>
<b>6</b>	<b>Structure and organisation .....</b>	<b>16</b>
6.1	Structure .....	16
6.2	Organisation .....	16
<b>7</b>	<b>Signature algorithm .....</b>	<b>17</b>
7.1	Key pairs .....	17
7.2	Hashing algorithm .....	17
<b>8</b>	<b>Certificate profiles .....</b>	<b>18</b>
8.1	Version .....	18
8.2	Certificates Serial Number .....	18
8.3	Signature .....	19
8.4	Issuer .....	19
8.5	Validity .....	19
8.6	Subject .....	20
8.7	Subject Public Key Info .....	21
8.8	Key usage extension .....	22
8.9	Authority and Subject Key Identifiers .....	22
8.10	NetscapeCertType .....	22

---

8.11	Policy mapping .....	23
8.12	Policy constraint .....	23
8.13	Certificate policies .....	23
8.14	Basic constraint.....	24
8.15	CRL Distribution Point.....	24
8.16	Freshest CRL - Delta CRL Distribution Point.....	25
8.17	Authority Information Access .....	25
8.18	Subject Directory attributes .....	26
<b>9</b>	<b>CRL profiles .....</b>	<b>26</b>
9.1	What is CRL?.....	26
9.1.1	Problems with all CRLs .....	27
9.2	eID CRL.....	27
9.2.1	CRL Profile.....	28
9.2.2	$\Delta$ CRL Profile .....	28
<b>10</b>	<b>OCSP Certificate.....</b>	<b>28</b>
10.1	What is OCSP? .....	28
10.1.1	Basic PKI implementation .....	29
10.1.2	Protocol Details.....	29
10.1.3	Advantages of OCSP over CRL .....	30
10.2	eID OCSP .....	30
<b>11</b>	<b>LDAP Scheme .....</b>	<b>30</b>

---

## 2 The use and future-use of the eID

The citizen will receive his identity card from his municipality, which is electronically connected to the federal government, the approved societies and the certification authorities that participate to the eID project at various levels. The National register also plays a key part in the organization as well as in the surveillance of the system.

### 2.1 Objective of the eID

The implementation of the electronic identity card (eID) is part of an e-Government project in order to simplify the administration and to modernize the public services.

The electronic identity card allows the citizen to identify himself electronically from a distance and to dispose of a legally valuable electronic signature. The result is a quick and customer-oriented service which guarantees the security of the private data of the owner.

The appearance of those applications combined with the legal recognition of the electronic signature will rapidly and safely replace a part of the paper documents by their electronic equivalent.

### 2.2 Visible data

The electronic identity card has the same size as a bank card. The basic identity informations are visible (name, first name, gender, nationality, place and date of birth, signature, national number and period of validity of the card) and you will also find a photo of the bearer. These informations (picture included) are also electronically contained in the card. The address of the owner and two electronic certificates complete the information's that are encrypted on the chip.

### 2.3 Electronic data

The chip of the electronic identity card contains the following information:

- The same informations as the one that are visible on the card (including the picture) + the address of the card owner;
- The identity and signature keys;
- The identity and signature certificates;
- The service provider of the certification;
- The information necessary for the authentication of the card;
- The information necessary for the for the protection of the electronically visible data That are encrypted on the card;
- The information necessary for the use of the corresponding qualified certificates.

The standard digital certificates allow the citizens to make an electronic signature and to identify themselves when they perform electronic transactions. They are also valid for the different e-Gov applications and for a private use, like for example:

- Sending of electronic registered mail;
- Signed e-mails;
- Online signing of contracts;
- Authentication by web browsers (e-Banking, e-Contracting...);
- etc.

## 2.4 What can be done

At this moment there are yet 6 E-government services available for people with an eID card.:

- ordering a licence plate for your car
- multifunctional tax return
- reporting of labour accidents
- VAT return
- starting an enterprise
- online election results

The possibilities in the future are enormous. A few examples:

- Digitally signing contracts  
Demanding birth certificates for new born children
- Instead of buying expensive foreign travel passports just ticking your card at a terminal in the airport
- Internet security (secure chat rooms etc)
- Banc account access for online trade

## 2.5 How is it done

The only thing needed is a card reader and a terminal with acces to the e-government.

Plug the card in the reader use your PIN code to validate it and its ready to use. For use by government services as the police or the registry office. As for home pc's people logged in on their pc with their eID card will have more credibility online.

The card itself and its PIN code prove that this card belongs to you and that it is you who is using it.

## 2.6 Microsoft is interested in our small country's idea

Bill Gates: "Microsoft™ is a believer of the possibilities of the Belgian electronic identity card and will contribute heavy support to the project."

Now already Microsoft™ has added a function to Microsoft Office 2003™ which uses the eID card. It is now possible to digitally sign an email and a word document. This firstly gives the recipient the certainty that the email came from that person himself and secondly the digital signature has the same legal value as a normal signature which gives a really added value to such digital documents. Imagine people signing contracts from within their comfy chair at home. Bill himself also wanted to use such a card in the future on his world wide used chat program "MSN Messenger™" so people can really know who they are chatting too. This could be a good help for fighting adult pedophiles impersonating children by example.

## 2.7 Big Brother

A lot of people might have the "Big Brother is watching us" feeling. This feeling is totally not justified. The eID card does not contain more personal information than the current non-electronic identity. The card is used to gain access to government networks and sevicees (because it proves our identity). It is not used to store extra information on your card, readable by anyone. Everyone can also look at his personal file and can also see who accessed it.

### 3 Middleware

#### 3.1 Introduction

The Belgian identity card middleware is software that is placed between the application implementing security features (digital signatures) and the device actually performing the cryptographic operations (the smartcard).

The middleware itself consists out of two independent interface implementations (see figure below).

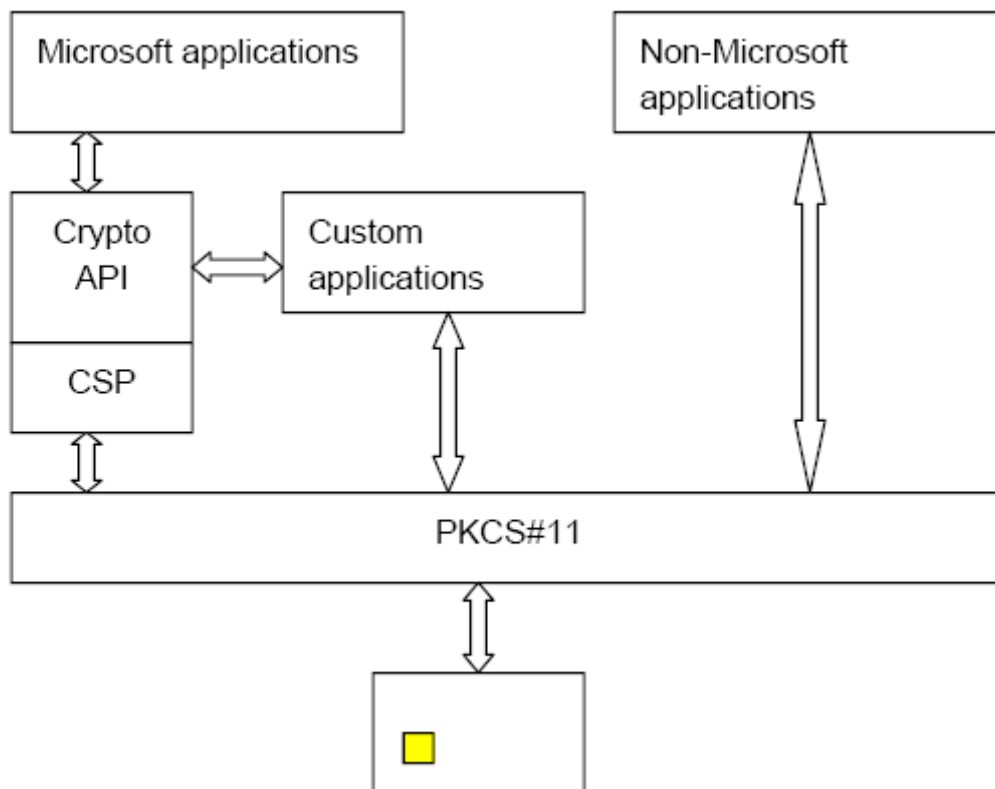
Although the implementations are independent, one makes use of the other. For the Microsoft®

standard applications (Office, Outlook...) a Cryptographic Service Provider (CSP) is created that

implements the cryptographic operations from the smartcard. An application will never call this

implementation directly but through a standard interface called Crypto API. The CSP implementation makes use of the second implemented interface, PKCS#11. This interface is used by non-Microsoft standard applications.

When a new application is created, it is up to the developer to decide which of the two interfaces will be used to offer cryptographic functionality to the user.



## 3.2 Interfaces

As described in the previous section, there are two interfaces implemented in the middleware software: one interface that can be called directly (the PKCS#11 interface) and one interface that is called indirectly (the CSP).

### 3.2.1 The Crypto API interface

The Microsoft® Cryptographic API 2.0 (CryptoAPI) enables application developers to add authentication, encoding, and encryption to their Win32®-based applications. Application developers can use functions in the CryptoAPI without knowing anything about the underlying implementation, in much the same way as they can use a graphics library without knowing anything about the particular graphics hardware configuration. The CSP part of the middleware establishes the link between the abstract CryptoAPI and the underlying PKCS#11 interface. The developer will never call any of the functions of the CSP directly but through the CryptoAPI.

The Belgian identity card (currently) only supports digital signature operations. When at a later date the Belgian government would decide to allow the user of the electronic identity card to add key material on the card that supports encryption then the CSP will be extended to allow for this

additional functionality. Furthermore the Belgian identity card contains two key pairs that can be

used for digital signatures (both authentication and non-repudiation).

Although the CSP only supports digital signatures, it is still registered as a PROV\_RSA\_FULL type of CSP. This is done in order to allow the usage of the CSP in standard Microsoft® applications.

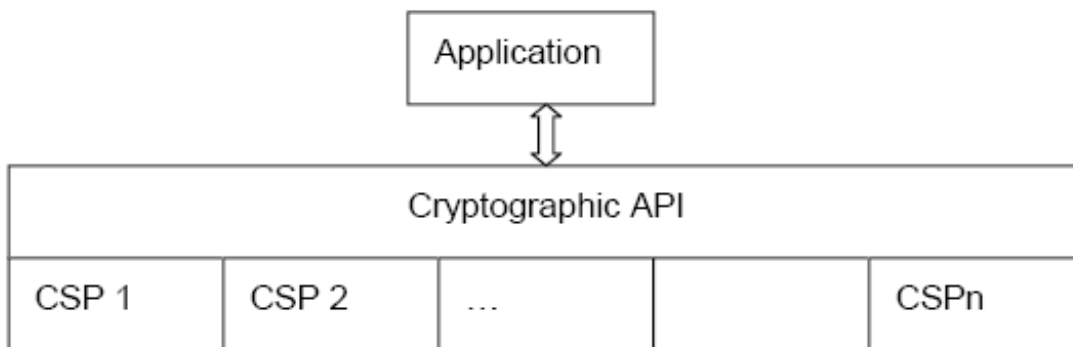
### CSP High level architecture

The Crypto application programming interface (API) provided by Microsoft is an abstract interface to cryptographic operations. It shields the user of the API of having to be knowledgeable about how the cryptographic functionality is implemented at a lower (card) level. This CryptoAPI interface is independent of the underlying cryptographic implementation. In case of the Belgian Electronic Identity card this is a smartcard. However in other applications this may even be a software solution.

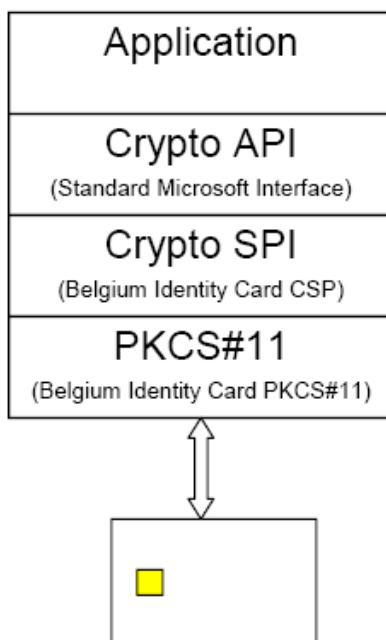
Underneath the CryptAPI interface a different interface is specified for the builders of security

implementations. This interface is called CSPI (Cryptographic Service Provider Interface). This

interface abstracts the underlying cryptographic device from the CryptoAPI interface. There can be any number of CSPI implementations available on any given system. Each CSPI implementation is (typically) specific to a given type of underlying hardware. Following figure gives an overview of this architecture:



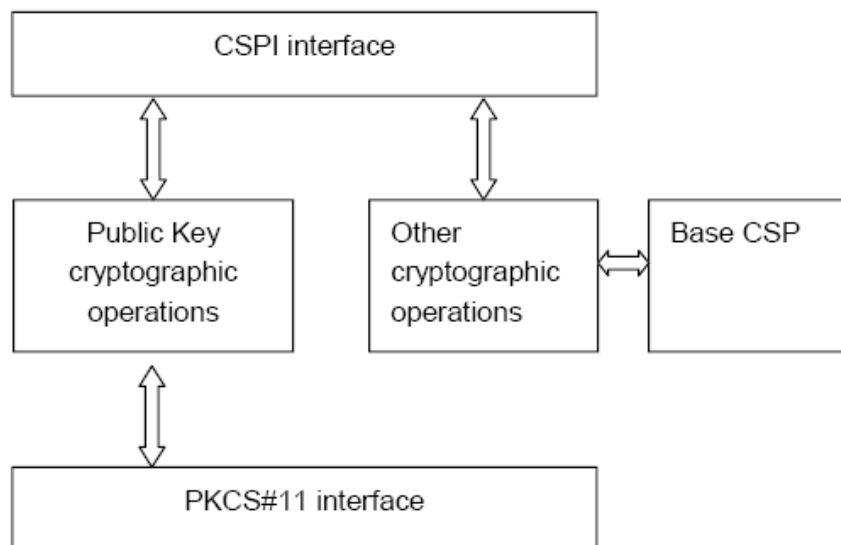
Typically each CSP orients itself to one specific smartcard type. This is because most of the time the CSP is provided by the smartcard vendor itself. The Belgian Identity Card middleware has used a different approach. In order to give the issuer of the identity card (i.e. the government) the maximum flexibility in regards to its choice of the type of smartcard used, the CSP does not implement the smartcard proprietary interface directly but through a PKCS#11 interface (for more information on the PKCS#11 interface please refer to section 0). Below is a figure with the relevant blocks:



Therefore, if at some later date the government would decide to change to a different physical smartcard then the CSP implementation will not need to change (provided the new smartcard is compatible with the existing one).

**CSP Low level architecture**

At a lower level the CSP implementation makes a translation between the standard CSPI interface and the PKCS#11 interface. Below is a figure that outlines the architecture:



Internally the CSP implements part of the PKCS#11 interface in order to perform the public key

cryptographic operations. Currently these operations are limited to digital signatures, both authentication and non-repudiation. For the non-public key cryptographic operations like calculating hash values a base CSP is used. Typically the default PROV\_RSA\_FULL CSP is used for these operations. In effect this will mostly be the Microsoft CSP.

A configuration file is maintained by the CSP. This configuration file contains card independent

settings linked to this electronic identity card. For instance the name (label) of the different keys is stored together with a link to the key ID for that key. This information is common for all Belgian electronic identity cards. So, there will not be a configuration file per card used on a given system.

User certificates are stored in the certificate stores from the operating system. For user certificates

this is the “MY” store. A friendly name is defined for each certificate (Authentication key and

Signature Key). In the Microsoft environment certificate containers are used to establish a link

between a certificate and the accompanying private key. The name of these containers is compiled out of the label of the key (Authentication or Signature) and the serial number of the card. This is done in order to allow the use more than one identity card on the same machine.

### 3.2.2 The PKCS#11 interface

The PKCS#11 (v2.11) interface is used by non-Microsoft applications like for instance Netscape.

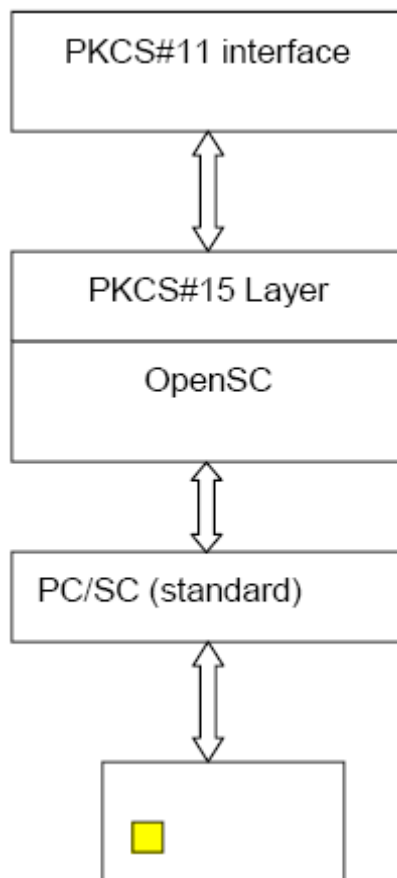
Also custom application can make use of this interface instead of the CryptoAPI interface. The PKCS#11 interface is sometimes also called Cryptoki.

A detailed description of this interface can be found on the website of RSA Laboratories (<http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/>).

In line with the desire of the government to be as open as possible, we have decided to use the open source PKCS#11 implementation of opensc ( <http://www.opensc.org> ). This implementation provides not only the implementation of the PKCS#11 interface but also supports the PKCS#15 card structure.

**PKCS#11 High level architecture**

The cryptoki interface provides an abstract interface to the smartcard for applications needing cryptographic functionality. Typically this interface is used by non-Microsoft applications like Netscape and Mozilla. Also independent solution providers can make use of this interface to implement cryptographic functionality in third party applications. Below is a figure that illustrates the PKCS#11 modules:



As indicated in the figure above, the top layer implements the standard PKCS#11 interface. It is this interface that is exposed to the applications. Below this interface layer the link to the card structure in PKCS#15 is made and the necessary commands (APDU) are sent through the standard PC/SC interface functions. In the OpenSC layer a specific card driver has been implemented in order to use the Belgian

electronic identity card. If at some later date a different card would be used, this driver would have to be replaced.

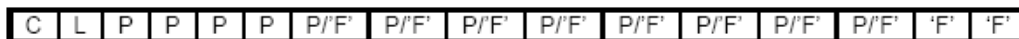
## 4 PIN / PUK Security

### 4.1 Readers with PIN pad

Readers with a PIN-pad must implement all commands (APDU) needing a PIN input without transmitting the PIN outside the reader.

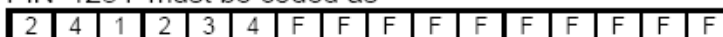
- ✓ Time-out
- ✓ PIN / PUK Input
- ✓ Incorrect PIN entry handling
- ✓ Display Clearing
- ✓ Secure PIN entry notification
- ✓ Card commands (APDU) to implement
  - Verify (PIN verify)
  - Change reference data (PIN change)
- ✓ Reader commands (APDU) to implement
  - PIN verify on reader
  - PIN change on reader
  
- ✓ PIN format

The PIN are 8-bytes strings with the following format (by nibble) as defined in the **Global PIN** section from the document "**Global Platform - Card Specification – version 2.0.1 – April 7, 2000**" and in the **F2PB** section from "**ISO 9564-1 – Banking – Personal Identification Number (PIN) management and security – 2002**":



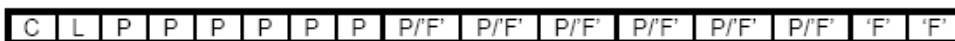
Nibble	Signification
C	Control parameter, contains always '2'
L	Length of the PIN (in nibbles) – from '4' to 'C'
P	PIN digits (minimum 4)
P/'F'	The rest of the PIN digits depending on the length, 'F' else
'F'	Padding contains always 'F'

Example: The PIN '1234' must be coded as



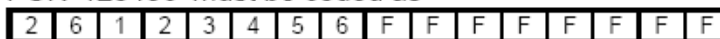
✓ PUK format

The PUK are 8-bytes strings with the following format (by nibble) as defined in the **Global PIN** section from the document “**Global Platform - Card Specification – version 2.0.1’ – April 7, 2000**” and in the **F2PB** section from “**ISO 9564-1 – Banking – Personal Identification Number (PIN) management and security – 2002**”:



Nibble	Signification
C	Control parameter, contains always '2'
L	Length of the PUK (in nibbles) – normally '6'
P	PUK digits (minimum 6)
P/'F'	The rest of the PUK digits depending on the length, 'F' else
'F'	Padding contains always 'F'

Example: The PUK '123456' must be coded as



## 4.2 Software compatibility with the middleware

This section is intended for readers that connect to a computer that will use the card through either the Microsoft CryptoAPI, or through the PKCS#11 interface.

## 4.3 PC/SC drivers

All readers must provide a driver compliant with PC/SC version 1.1 for the target OS.

## 4.4 Integration with the middleware

In order to integrate with the middleware, the reader provider must develop, for each target Operating System, a **Dynamic Linked Library** (or equivalent) implementing the following functions:

- **SCR\_Init()**
- **SCR\_VerifyPIN()**
- **SCR\_ChangePIN()**

A **C** header file is provided at the end of the document, with all the function prototypes and the constants for the return codes and parameters.

Currently, only the Belgian Identity Application (noted below as **ID**) resides on the card; to anticipate possible other applications (implemented as separate *Data Files* or *Directories* in the card) in the future, some functions receive a parameter called **ApplicationID** that contains a string identifying the application that want to interact with one of its PIN. The goal is that the reader displays (see 3.3) the application that asks for a PIN.

### 4.4.1 Parameters and return code

- All functions must return a standard **PC/SC** return code (LONG).  
Additional return codes:
  - SCARD\_E\_CANCELLED: the user pressed **Cancel**
  - SCR\_E\_UNINITIALIZED: SCR\_Initialised( ) was not called
  - SCR\_I\_PIN\_CHECK\_FAILED: New PIN mismatch in SCR\_ChangePIN( )
- The parameter **language** is a 2-characters string as defined in the ISO 639 standard. The DLL must, at least, support the following languages:
  - **fr** French
  - **nl** Dutch
  - **de** German
  - **en** English
- The functions use the following structures:

<b>SCR_Bytes</b>		
<b>data</b>	BYTE *	The data itself
<b>length</b>	DWORD	The data length

<b>SCR_Card</b>		
<b>hCard</b>	SCARDHANDLE	PC/SC handle to the card
<b>language</b>	char *	ISO 639 2-characters code
<b>id</b>	SCR_bytes	Card type unique identifier. Usually the PKCS#15 <b>OID</b> .
<b>pinFormat</b>	void *	Not used yet; should be <b>NULL</b>

<b>SCR_Application</b>		
<b>id</b>	SCR_bytes	Application unique identifier ( <b>AID</b> ).
<b>shortString</b>	char *	3-characters string application identifier in the user's language
<b>longString</b>	char *	Long string application identifier in the user's language

<b>SCR_PinUsage</b>			
<b>code</b>	DWORD	SCR_USAGE_AUTH	Authentication / logon
		SCR_USAGE_SIGN	Non-repudiation
		SCR_USAGE_DECR	Decryption
		SCR_USAGE_PREF_MODIF	Preference file modification
		SCR_USAGE_ADMIN	Administrative maintenance
<b>shortString</b>	char *	3-characters string application identifier in the user's language	
<b>longString</b>	char *	Long string application identifier in the user's language	

### 3.2.2. SCR\_Init()

This function will be called immediately after loading the DLL. It is used to initialise the DLL and check if it supports the connected reader.

<b>Parameters</b>	<b>in/out</b>	<b>Type</b>	<b>Description</b>
szReader	in	LPCTSTR	Reader name
version	in	DWORD	Calling program interface version: <b>1</b>
supported	out	DWORD *	<ul style="list-style-type: none"> <li>- SCR_SUPPORT_OK</li> <li>- SCR_SUPPORT_INCOMPATIBLE_CALLING_VERSION</li> <li>- SCR_SUPPORT_INCOMPATIBLE_FIRMWARE</li> <li>- SCR_SUPPORT_INCOMPATIBLE_FIRMWARE_VERSION</li> </ul> <i>This allows to dynamically try all registered DLL to find the one corresponding to the reader –cf. “plug and play” mechanism.</i>

### 3.2.3. SCR\_VerifyPIN()

This function verifies a user’s PIN entered on the reader’s PIN-pad.

<b>Parameters</b>	<b>in/out</b>	<b>Type</b>	<b>Description</b>
card	in	const SCR_Card *	Handle to the card
pinID	in	BYTE	PIN identifier in the card – provided by the middleware. To specify as parameter P2 in function <b>Reader PIN Verify</b>
usage	in	const SCR_PinUsage *	Reason to enter the PIN
application	in	const SCR_Application *	Application to display on the reader’s display.
cardStatus	out	BYTE *	Card status (2 bytes). Fill with ‘O’ if no specific answer from the card

### 3.2.4. SCR\_ChangePIN()

This function changes a user’s PIN entered on the reader’s PIN-pad by a new one, also entered on the reader’s PIN-pad.

<b>Parameters</b>	<b>in/out</b>	<b>Type</b>	<b>Description</b>
card	in	const SCR_Card *	Handle to the card
PinID	in	BYTE	PIN identifier in the card. To specify as parameter P2 in function <b>Reader PIN Change</b>
application	in	const SCR_Application *	Application to display on the reader’s display.
cardStatus	out	BYTE *	Card status (2 bytes). Fill with ‘O’ if no specific answer from the card

### 4.4.2 Display

The parameters **usage** and **application** are important information for the citizen.

**usage** shows why the citizen is asked to enter his PIN (to authenticate himself, to sign a document, etc.).

The DLL has to present, on the reader's display, a string corresponding to this usage; the DLL may either display the provided **usage.longString**, or the provided **usage.shortString** if the display is very small, or make use of the provided **usage.code** to optimise it for its display.

**application** identifies the application. Currently, only the Belgian Identity application (**ID**) exists on the card, but some other ones could come later (ex: Doctors: **Doc**, Lawyers: **LAW**, etc.). Therefore this information must be given also. The same rule applies for **application.longString**, **application.shortString**, and **application.code**.

Here is an example of the information the reader must provide on its display when calling the above functions:

#### 3.3.1. SCR\_VerifyPIN()

If the display has limited space, the minimum information to display is :

**ApplicationID-Usage: PIN ? \*\*\*\***

ex: **ID-S: PIN ? \*\*\*\***

If the display has space enough, the information should be translated in the language defined in the **SCR\_Init** function, like

<b>Application:</b>	<b>Identité</b>
<b>Accès:</b>	<b>Signature (non-répudiation)</b>
<b>Entrez votre PIN:</b>	<b>****</b>

#### 3.3.2. SCR\_ChangePIN()

If the display has limited space, the minimum information to display is :

**ApplicationID-PIN: Old PIN ? \*\*\*\***    **ApplicationID-PIN: New PIN ? \*\*\*\***

ex: **Doc-PIN: Oude PIN ? \*\*\*\***    **Doc-PIN: Nieuwe PIN ? \*\*\*\***

If the display has space enough, the information should be translated in the language defined in the **SCR\_Init** function, like

	<b>PIN Verandering</b>
<b>Applicatie:</b>	<b>Advocaten</b>
<b>Oude PIN ?</b>	<b>****</b>
<b>Nieuwe PIN ?</b>	<b>****</b>
<b>Nieuwe PIN ?</b>	<b>**** (Controle)</b>

## 5 Executive Summary

A dedicated PKI infrastructure will be implemented in order to provide the necessary keys and certificates lifecycle management for the eID project.

The eID hierarchy has three levels. The first level is the Belgium Root CA; the second level contains the eID operation CAs (including the citizen CA), while the third level concerns the end entities (citizens).

Two kinds of certificates will be used: CA certificates and citizen certificates. All certificates are formatted according to version 3 of the X509 recommendation. The CRL profile as well as the OCSP profile are following also the international standards.

## 6 Structure and organisation

### 6.1 Structure

The eID hierarchy is a hybrid hierarchy that consists of a combination of 2 and 3 layer models. The hierarchy that will be present on the smartcard consists of 2 levels, A Self-Signed Belgium Root CA and an operational Citizen CA certificate. The extensions present in the end user certificate profile will allow another hierarchy to be reconstructed up to a trusted root that is present in common browsers (3-Level hierarchy).

Figure 1 details this structure.

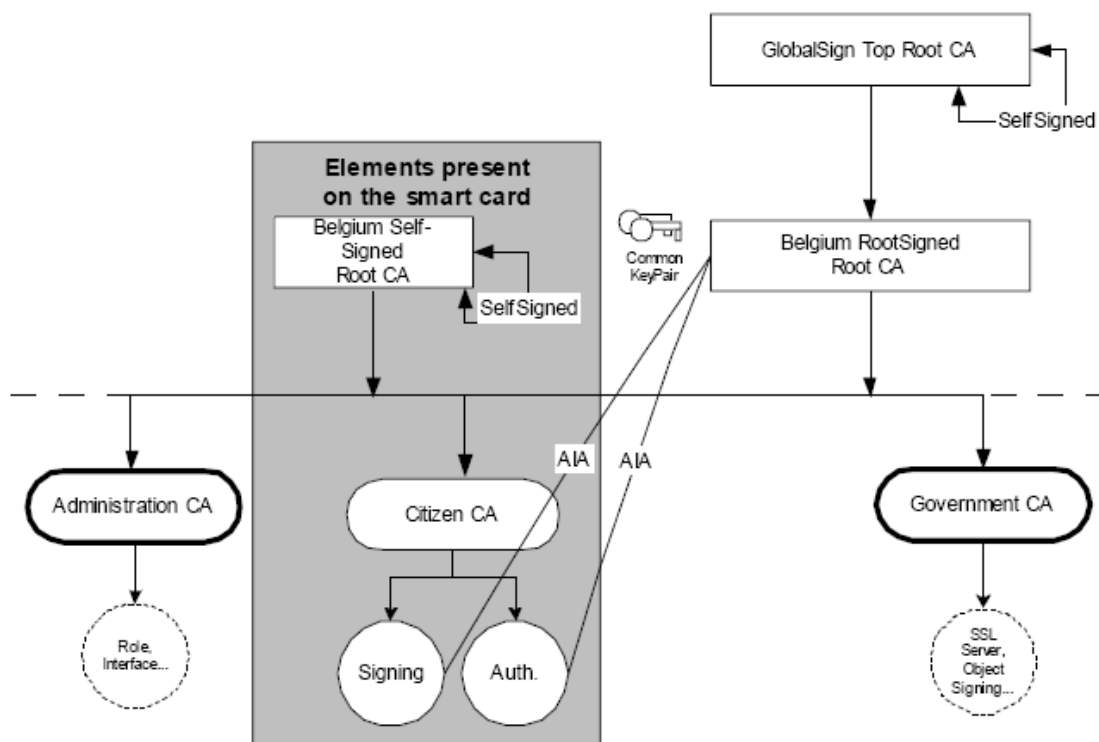


Figure 1: eID Architecture

### 6.2 Organisation

The Belgium Root CA is the entity designated by Government as primary CA to manage the other eID CAs. The Belgium Root CA is root signed by the GlobalSign Top Root under the terms of a RootSigning™ contract. The CA operator will manage the Belgium Root CA and eID CA key pairs and associated certificate in compliance with the Key Management Policy. Belgium Root CA, Citizen CA and citizen certificates are ruled by the different “eID” CPS-documents

## 7 Signature algorithm

### 7.1 Key pairs

The key pairs in this hierarchy will be generated using the RSA cryptographic algorithm. The length of the Belgium Root CA and Citizen CA certificates key pairs are 2048 bits. The citizen certificates key pairs size are 1024 bit. The lifetime of keys is specified as the period of validity of certificates associated to the keys.

The mathematical details of the algorithm used in obtaining the public and private keys are available at the RSA Web site. Briefly, the algorithm involves multiplying two large prime numbers (a [prime number](#) is a number divisible only by that number and 1) and through additional operations deriving a set of two numbers that constitutes the [public key](#) and another set that is the [private key](#). Once the keys have been developed, the original prime numbers are no longer important and can be discarded. Both the public and the private keys are needed for encryption /decryption but only the owner of a private key ever needs to know it. Using the RSA system, the private key never needs to be sent across the Internet.

The private key is used to decrypt text that has been encrypted with the public key. Thus, if I send you a message, I can find out your public key (but not your private key) from a central administrator and encrypt a message to you using your public key. When you receive it, you decrypt it with your private key. In addition to encrypting messages (which ensures privacy), you can authenticate yourself to me (so I know that it is really you who sent the message) by using your private key to encrypt a [digital certificate](#). When I receive it, I can use your public key to decrypt it. A table might help us remember this.

To do this	Use whose	Kind of key
Send an encrypted message	Use the receiver's	Public key
Send an encrypted signature	Use the sender's	Private key
Decrypt an encrypted message	Use the receiver's	Private key
Decrypt an encrypted signature (and authenticate the sender)	Use the sender's	Public key

### 7.2 Hashing algorithm

A hashfunction,  $h$ , converts any size of input into a fixed (smaller) size of output. There are six requirements for a hash function:

1. It should accept any length message as input.
2. It should produce a small fixed-size output (~100 bits).
3. It should be easy and fast to compute  $h$  for any input.
4. It should be a one-way function-hard or impossible to invert. That is, given  $h(m)$ , it should be very difficult to recover  $m$ .
5. It should be resistant to weak collisions. A collision occurs when two different inputs produce the same output.

6. It should be resistant to strong collisions-that is, it should be almost impossible to find two meaningful messages  $m_1$  and  $m_2$  such that  $h(m_1) = h(m_2)$ .

The hashing algorithm used is SHA-1 (Secure Hash Algorithm)

## 8 Certificate profiles

The different CAs are profiled according to PKIX certificate profile, and made up of three parts according to RFC 2459: tbsCertificate, Signature algorithm and Signature value.

Note: All the URI's specified in the certificate profiles are resolved by FedICT.

### 8.1 Version

The version field indicates the X.509 version of the certificate format. In the eID project, only certificates complying with version 3 of the X.509 recommendation, allowing for extensions, are used.

In the X.509 system, a CA issues a certificate binding a public key to a particular Distinguished Name in the X.500 tradition, or to an Alternative Name such as an e-mail address or a DNS-entry.

An organisation's trusted root certificates can be distributed to all employees so that they can use the company PKI system. Browsers such as Internet Explorer, Netscape/Mozilla and Opera come with root certificates pre-installed, so SSL certificates from larger vendors who have paid for the privilege of being pre-installed will work instantly; in essence the browser's owners determine which CAs are trusted third parties. Whilst these root certificates can be removed or disabled, users rarely do so.

X.509 also includes standards for certificate revocation list (CRL) implementations, an often neglected aspect of PKI systems. The IETF-approved way of checking a certificate's validity is the Online Certificate Status Protocol (OCSP).

### 8.2 Certificates Serial Number

The field certificate serial number specifies the unique, numerical identifier of the certificate within all certificates issued by the same Certification Authority (CA). The RRN<sup>1</sup> will assign a serial number to the Self-Signed Belgium Root CA and the Citizen CA. The citizen certificates serial number will be assigned by the RRN. The CA operator will have to check the uniqueness of end-user certificate serial numbers before processing the certification requests. The CA operator will create the serial number for the RootSigned Belgium Root CA certificate. All serial numbers are maximal 16 bytes long.

---

<sup>1</sup> RRN is an acronym for Rijksregister – Registre National

Serial Number			
Belgium Root CA RootSigned certificate	Belgium Root Self-Signed CA certificate	Citizen CA certificate	Citizen certificates
Generated by the CA at the time of Key Generation Process	Provided by the RRN	Provided by the RRN	Assigned by RRN in the XKMS / X-BULK requests.

Remark: if no serial number is received in the requests issued by the RRN, the CA provider will generate this number using its own allocation scheme.

### 8.3 Signature

The signature field determines the cryptographic algorithm used by a CA to sign a certificate. The algorithm identifier, which is a number registered with an internationally recognised standards organisation, specifies both the public-key algorithm and the hashing algorithm used by the CA to sign certificates. The Object Identifier for RSA with SHA1 is 1.2.840.113549.1.1.5.

Signature		
Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
RSAwithSHA1	RSAwithSHA1	RSAwithSHA1

### 8.4 Issuer

The Issuer field identifies the certification authority that has signed and issued the certificate. Issuer is structured as a “Distinguished Name”, that is a hierarchically structured name, composed of attributes, most of which are standardised in the X.500 attributes. The ones that will be used are: country, organisation, serial number, common name. The subject serial number mentioned in the issuer field is the serial number attributed by the RRN to identify the CA.

Issuer			
Belgium Root CA RootSigned certificate	Belgium Root CA SelfSigned certificate	Citizen CA certificate	Citizen certificates
CN = GlobalSign Root CA OU = Root CA O = GlobalSign nv-sa C = BE	CN: Belgium Root CA C: BE	CN: Belgium Root CA C: BE	CN: Citizen CA C: BE

### 8.5 Validity

The validity field indicates the time interval during which the citizen can use the certificate

and over which the issuing CA maintains certificate status information. The certificates can be used by a citizen, unless a certificate is suspended or revoked during its period of validity. Validity should be interpreted as the period when the (non-revoked) certificate can be trusted to perform a certain transaction. All transactions executed after this period with the certificate should be handled as not trusted.

Validity		
Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
NotBefore: 26/01/2003 23:00:00 (UTC Time)	NotBefore: 27/01/2003 00:00:00 (UTC Time).	NotBefore: Certificate issuance date and time in UTCTime.
NotAfter: 26/01/2014 23:00:00 (UTC Time)	NotAfter: 26/06/2009 23:00:00 (UTC Time)	NotAfter: issuing UTCTime + 5 years.

The FIRST eID operational certificate generated has a validity extended to 6 years and 5 months in order to avoid unnecessary additional Key Ceremonies.

### 8.6 Subject

The Subject field identifies the entity holding the private key corresponding to the public key published in the certificate. Subject is structured as a set of attributes, defined in the X.500 attributes.

Subject		
Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
countryName: BE commonName: Belgium Root CA	countryName: BE commonName: Citizen CA	countryName (Dynamic) commonName (Dynamic) surname (Dynamic) givenName (Dynamic) subjectSerialNumber (Dynamic)

The common name of each Citizen certificate must be formatted as follows:Example

Field	Length	Description	Example
C ( <i>countryName</i> )	2	countryName is the country of issuance of the certificate. It has thus always "BE" as value. However as it is a dynamic element not checked by the CA, the country code within the request may be changed by the RRN.	C=BE
CN ( <i>commonName</i> )	Max 255 Min 1	Concatenation of <ul style="list-style-type: none"> <li>- &lt;given name&gt;: first given name of the citizen</li> <li>- &lt;surname&gt;: surname of the citizen</li> <li>- (&lt;purpose&gt;): (Authentication) or (Signature)</li> </ul>	CN=John Smith (Authentication)
surname	Max 255 Min 1	Surname of the citizen	S=Smith
givenName	Max 255 Min 1	1 or 2 given names of the citizen (This field may not appear in case the citizen has no given name)	G=John William
subjectSerialNumber	Max 255 Min 1	This is a unique number for each citizen provided by the RRN (so called "Rijksregisternummer" - 11 digits long).	SN=12345678901

The CA operator does not perform a check on the content provided by the RRN, except that the subject distinguished name has to be unique.

### 8.7 Subject Public Key Info

The Subject Public Key Info field is used to carry the public key being certified and identify the algorithms with which the key has been generated.

Subject Public Key Info		
Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
RSA 2048 bits public key	RSA 2048 bits public key	RSA 1024 bits public key

### 8.8 Key usage extension

The Key Usage extension field specifies the purpose of the key contained in the certificate.

Key usage extension				
Key usage components	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates	
			Authentication	Signature
digitalSignature	not asserted	not asserted	asserted	not asserted
nonRepudiation	not asserted	not asserted	not asserted	asserted
keyEncipherment	not asserted	not asserted	not asserted	not asserted
dataEncipherment	not asserted	not asserted	not asserted	not asserted
keyAgreement	not asserted	not asserted	not asserted	not asserted
keyCertificateSigning	asserted	asserted	not asserted	not asserted
crlSigning	asserted	asserted	not asserted	not asserted
encipherOnly	not asserted	not asserted	not asserted	not asserted
decipherOnly	not asserted	not asserted	not asserted	not asserted

The digital signature bit is not asserted in the Citizen Signing Certificates for strict application of the standards, and to prevent possible mistakes with applications.

### 8.9 Authority and Subject Key Identifiers

The Authority Key Identifier extension will be present in the end user certificates, the eID operational CA certificate(s) and the Belgium Root CA certificates (Self-Signed and RootSigned).

The Subject Key Identifier will be present in the Citizen CA certificate(s) and the Belgium Root CA certificates (Self-Signed and RootSigned). It will not be present in end-user certificates.

### 8.10 NetscapeCertType

This extension can be used to limit the applications for a certificate. If the extension exists in a certificate, it will limit the uses of the certificate to those specified. If the extension is not present, the certificate can be used for all applications except Object Signing.

NetscapeCertType Key usage extension				
Netscape Key usage	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates	
			Authentication	Signature
bit-0 - SSL client	not asserted	not asserted	asserted	not asserted
bit-1 - SSL server	not asserted	not asserted	not asserted	not asserted
bit-2 - S/MIME	not asserted	not asserted	asserted	asserted
bit-3 - Object Signing	not asserted	not asserted	not asserted	not asserted
bit-4 - Reserved	not asserted	not asserted	not asserted	not asserted
bit-5 - SSL CA	asserted	asserted	not asserted	not asserted
bit-6 - S/MIME CA	asserted	asserted	not asserted	not asserted
bit-7 - Object Signing CA	asserted	asserted	not asserted	not asserted

### 8.11 Policy mapping

This extension is only useful in case of cross-certification between CAs. It makes indeed little sense to have a policy mapping between a commercial CA and a governmental CA. Also this extension is not handled by Netscape nor by Microsoft products. As such the Policy Mapping has not been implemented.

### 8.12 Policy constraint

This extension can be used in CA certificates only. It can be used to constrain path validation in two ways: to prohibit policy mapping, or to require that each certificate in a path contain an acceptable policy identifier. If present, this extension should be marked critical [X509]. For the same reasons as mentioned in chapter 5.11, the Policy Constraint has not been implemented.

### 8.13 Certificate policies

Certificate policies will be present in the eID certificates as a CPS Pointer qualifier containing a pointer to the Certification Practice Statement (CPS) published by the CA. The same sequence will be used for all eID certificates as it has been decided this qualifier will point to a web page that may reference multiple applicable documents.

CertificatePolicies				
	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates	
			Authentication	signature
policyIdentifier	2.16.56.1.1.1	2.16.56.1.1.1.2	2.16.56.1.1.1.2.2	2.16.56.1.1.1.2.1.
policyQualifiers	N/a			
policyQualifierId	CPS			
Qualifier	<a href="http://repository.eid.belgium.be">http://repository.eid.belgium.be</a>			

### 8.14 Basic constraint

The Basic Constraints extension specifies whether the subject of the certificate may act as a CA or only as an end-user. If the subject may act as a CA, then the certificate is a crosscertificate, and it may also specify the maximum acceptable length of a certificate beyond the cross-certificate. This extension should always be marked as critical; otherwise some implementations will ignore it and allow a non-CA certificate to be used as a CA certificate.

Basic constraint extension				
Basic constraint components	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates	
			Authentication	signature
CA	true	True	~	~
PathLengthConstraint	none	0	~	~

### 8.15 CRL Distribution Point

The CRL Distribution Points extension identifies the CRL distribution point or points to which a certificate user should refer to ascertain if the certificate has been revoked. A certificate user can obtain a CRL from an applicable distribution point or it may be able to obtain a current complete CRL from the authority directory entry.

CRL Distribution Point extension			
CRL Distribution Point Component	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
distributionPoint	<a href="http://secure.globalsign.net/crl/root.crl">http://secure.globalsign.net/crl/root.crl</a> CDP present in RootSigned certificate only.	<a href="http://crl.eid.belgium.be/belgium.crl">http://crl.eid.belgium.be/belgium.crl</a>	<a href="http://crl.eid.belgium.be/eidcxxxx.crl">http://crl.eid.belgium.be/eidcxxxx.crl</a>

Note: the ‘xxxx’ in the distribution point of the citizen certificates should be replaced with a number pointing to the CRL of the issuing operational Citizen CA (e.g. <http://crl.eid.belgium.be/eidc0001.crl> ).

### 8.16 Freshest CRL - Delta CRL Distribution Point

As the applications do not yet support ‘Delta CRL Distribution Point’ it was decided not to implement this now.

### 8.17 Authority Information Access

The ‘Authority Information Access’ extension indicates how to access the information and services provided by the issuer of a certificate, such as on-line validation services or LDAP server location. An HTTP reference to the issuing CA has been added as a caIssuers element in order to allow the certificate chain to be reconstructed up to a trusted root. As a shared OCSP responder will be used, OCSP validation of CA certificates is not supported.

Authority Information Access extension			
Authority Information Access component	Belgium Root CA certificate	Citizen CA certificate	Citizen certificates
accessMethod	~	~	<i>id-ad-ocsp (OCSP)</i>
accessLocation	~	~	<a href="http://ocsp.eid.belgium.be">http://ocsp.eid.belgium.be</a>
accessMethod	~	~	<i>id-ad-caIssuers (HTTP)</i>
accessLocation	~	~	<a href="http://certs.eid.belgium.be/belgiumrs.crt">http://certs.eid.belgium.be/belgiumrs.crt</a>

RFC3280 specifies: “The id-ad-caIssuers OID is used when the additional information lists CAs that have issued certificates superior to the CA that issued the certificate containing this extension. The referenced CA issuers description is intended to aid certificate users in the selection of a certification path that terminates at a point trusted by the certificate user.” It doesn’t thus make sense to put accessMethod caIssuers in the government and eID CA certificates. The LDAP access method will not be used in any of the eID certificate profiles described in this document.

## 8.18 Subject Directory attributes

The Subject Directory Attributes are applicable to citizen certificates only, and convey any desired Directory attribute values for the subject of the certificate that are complement to the information contained in the subject field. This extension is always non-critical.

No subject directory attributes will be present in the eID certificates

## 9 CRL profiles

### 9.1 What is CRL?

In the operation of some cryptosystems, usually public key infrastructures (PKIs), a certificate revocation list (CRL) is a list of certificates (more accurately: their serial numbers) which have been revoked, are no longer valid, and should not be relied upon by any system user.

There are different revocation reasons defined in RFC 3280:

1. **Revoked:** A certificate is irreversibly revoked (and entered on a CRL) if, for instance, it is discovered that the certificate authority (CA) had improperly issued a certificate or a private-key is thought to have been compromised. Certificates may also be revoked for failure of the identified entity to adhere to policy requirements such as publication of false documents, mis-representation of software behavior, or violation of any other policy specified by the CA operator or its customer. The most common reason for revocation is the user's not being in sole possession of the private key (e.g token containing the private key has been lost or stolen).
2. **Hold:** This reversible status can be used to notice the temporary invalidity of the certificate, for instance when the user is not sure if the private key has been lost. If, in this example, the private key was found again and nobody had access to it, the status can be reinstated, and the certificate is valid again, thus removing the certificate from further CRLs.

Usually, a CRL is generated on the one hand periodically after a clearly defined timeframe and (optionally) on the other hand immediately after a certificate has been revoked. The CRL is always issued by the CA which issues the corresponding certificates. All CRLs have a (often short) lifetime in which they are valid and in which they may be consulted by a PKI-enabled application to verify a counterpart's certificate prior its use. To prevent spoofing or denial-of-service attacks, CRLs are usually signed by the issuing CA and therefore carry a digital signature. To validate a specific CRL prior relying on it, the certificate of its corresponding CA is needed, which usually can be found on a (even public) directory.

Certificate expiration dates are not a substitute for a CRL as the problem may be discovered whilst the certificate has not yet expired. CRLs or other certificate validation techniques are a necessary part of any properly operated PKI as mistakes in certificate vetting and key management are expected to occur in real world operations.

In a noteworthy example, a certificate for Microsoft was mistakenly issued to an unknown individual who had successfully posed as Microsoft by the CA contracted to maintain the ActiveX 'publisher certificate' system (VeriSign). Microsoft saw the need to patch their

cryptography subsystem so it would check the status of certificates before trusting them. As a short term fix, a patch was issued for the relevant Microsoft software (most importantly Windows) specifically listing the two certificates in question as 'revoked'.

The certificates for which a CRL should be maintained are often X.509/public key certificates, as this format is commonly used by PKI schemes.

### 9.1.1 Problems with all CRLs

Best practices require that wherever and however certificate status is maintained, it must be checked whenever one wants to rely on a certificate. Failing this, a revoked certificate may be incorrectly accepted as valid.

This means that to effectively use a PKI one must have access to current CRLs (i.e. Internet access in the case of a PKI). This requirement of on-line validation negates one of the original major advantages of PKI over symmetric cryptography protocols, namely that the certificate is "self authenticating". Symmetric system, e.g. Kerberos, also depend on the existence of on-line services (Key distribution centre in the case of Kerberos).

The existence of a CRL implies the need for someone (or some organization) to enforce policy and revoke certificates deemed counter to operational policy. If a certificate is mistakenly revoked significant problems can arise.

As the certificate authority is tasked with enforcing the operational policy for issuing certificates they typically are responsible for determining if and when revocation is appropriate by interpreting the operational policy.

The necessity of consulting a CRL, or other certificate status service, prior to accepting a certificate raises a potential denial-of-service attack against the PKI akin to the denial-of-service attack on Kerberos whereby a current authentication token cannot be retrieved.

No comprehensive solution to these problems is known, though there are multiple workarounds for various aspects of it, some of which have proven acceptable in practice.

An alternative to using CRLs which is especially useful for software clients is the on-line certificate validation protocol Online Certificate Status Protocol (OCSP). OCSP has the primary benefit of requiring less network bandwidth and thus enabling real-time and near real-time status checks for high volume or high value operations.

## 9.2 eID CRL

The CRLs and  $\Delta$  CRLs will be created according to the profiles as described in the chapters 9.2.1 and 9.2.2. All CRLs and  $\Delta$  CRLs are signed by the issuing CA.

### 9.2.1 CRL Profile

Version	v2
Signature	sha1RSA
Issuer	<subject CA>
ThisUpdate	<creation time>
NextUpdate	<creation time> + 7 days
RevokedCertificates	
UserCertificate	<certificate serial number>

'nextUpdate' is the latest time that the delta CRL can be used by a citizen.

### 9.2.2 Δ CRL Profile

Version	v2
signature	sha1RSA
Issuer	<subject CA>
thisUpdate	<creation time>
nextUpdate	<creation time> + 7 days
RevokedCertificates	
userCertificate	<certificate serial number>
revocationDate	<revocation time>
crlEntryExtensions	
CRL Reason Code	certificateHold(6) (for suspended certificates) removeFromCrl(8) (to unsuspend certificates) Note: otherwise not included
crlExtensions	
Authority Key Identifier	non-critical <subject key identifier CA>
CRL Number	non-critical <The CA operator assigned unique number>
Delta CRL Indicator	critical <base CRL Number>

'nextUpdate' is the latest time that the delta CRL can be used by a citizen.

## 10 OCSP Certificate

### 10.1 What is OCSP?

The Online Certificate Status Protocol (OCSP) is an internet protocol used for obtaining the revocation status of an X.509 digital certificate. It is described in RFC2560 and is on the internet standards track. It was created as an alternative to Certificate revocation lists (CRL,

which will be discussed later on), specifically addressing certain problems associated with using CRLs in a Public Key Infrastructure (PKI). Messages communicated via OCSP are encoded in ASN.1 and are usually communicated over HTTP. The “request/response” nature of these messages leads OCSP servers being termed OCSP responders.

### 10.1.1 Basic PKI implementation

1. Alice and Bob have public key certificates issued by Ivan, the Certificate Authority (CA).
2. Alice wishes to perform a transaction with Bob and sends him her public key certificate.
3. Bob, concerned that Alice's private key may have been compromised, creates an 'OCSP request' that contains a fingerprint of Alice's public key and sends it to Ivan.
4. Ivan's OCSP responder looks up the revocation status of Alice's certificate (using the fingerprint Bob created) in his own CA database. If Alice's private key had been compromised, this is the only trusted location at which the fact would be recorded.
5. Ivan's OCSP responder confirms that Alice's certificate is still OK, and returns a signed, successful 'OCSP response' to Bob.
6. Bob cryptographically verifies the signed response (He has Ivan's public key on-hand -- Ivan is a trusted responder) and ensures that it was produced recently.
7. Bob completes the transaction with Alice.

### 10.1.2 Protocol Details

An OCSP responder may return a signed response signifying that the certificate supplied in the request is 'good', 'revoked' or 'unknown'. If it cannot process the request, it may return an error code.

The OCSP request format supports additional extensions. This enables extensive customization to a particular PKI scheme.

OCSP can be resistant to replay attacks, where a signed, 'good' response is captured by a malicious intermediary and replayed to the client at a later date after the subject certificate may have been revoked. OCSP overcomes this by allowing a nonce to be included in the request that must be included in the corresponding response.

However, the replay attack, while a possibility, is not a major threat to validation systems. This is due to the steps it takes to actually exploit this weakness. The attacker would have to be in a position to

1. Capture the traffic and subsequently replay that traffic,
2. Capture the status of a certificate whose status is about to change and
3. Conduct a transaction requiring the status of that certificate within the time frame of the validity of the response.

Since it is not often that a revoked certificate is unrevoked (only if it is suspended is it even possible) a person would have to capture a good response and wait until the certificate was revoked then replay it.

OCSP can support more than one level of CA. OCSP requests may be chained between peer responders to query the issuing CA appropriate for the subject certificate, with responders validating each other's responses against the root CA using their own OCSP requests.

An OCSP responder may be queried for revocation information by Delegated Path Validation (DPV) servers. OCSP does not, by itself, perform any DPV of supplied certificates.

### 10.1.3 Advantages of OCSP over CRL

When deploying a PKI, certificate validation using OCSP may be preferred over the use of CRLs for several reasons:

1. OCSP can provide more timely information regarding the revocation status of a certificate.
2. OCSP removes the need for clients to retrieve the (sometimes very large) CRLs themselves, leading to less network traffic and better bandwidth management.
3. Using OCSP, clients do not need to parse CRLs themselves, saving client-side complexity.
4. An OCSP responder may implement billing mechanisms to pass the cost of validation transactions to the seller, rather than buyer.
5. CRLs may be seen as analogous to a credit card company's "bad customer list" -- an unnecessarily public exposure.
6. To a degree, OCSP supports trusted chaining of OCSP requests between responders. This allows clients to communicate with a trusted responder to query an alternate responder, saving client-side complexity.

## 10.2 eID OCSP

A shared OCSP key pair will be generated under the shared security environment of GlobalSign. A branded OCSP certificate will be created within the scope of the eID project. Due to constraints linked to the use of the shared infrastructure, the OCSP certificate profile has to be identical in terms of extensions and attributes to the standard GlobalSign OCSP responder certificate profile. The only possible changes are the C and CN SDN extensions. Each operational CA will sign the key-pair of the OCSP server, so that there are as many OCSP certificates as there are operational CAs. All these OCSP certificates are using the same key-pair.

The OCSP certificate profile extension `ocspNoCheck` is used to define that there is no need for validation of the full CA chain using OCSP.

## 11 LDAP Scheme

The scheme used for the eID certificates is kept as easy as possible. The LDAP node under which the eID certificates are published is defined as follows: `dc=eid dc=belgium dc=be`. All certificates will be published under this node under a flat file structure, where every entry will have an unique 10 digits UID randomly assigned by the CA2. Besides the certificate itself, all certificate subject distinguished name (SDN) information is published in the LDAP. All Subject Distinguished Name information present in the certificates as per the end-user certificate profiles is searchable. Besides the certificates, the LDAP will also contain CRL and  $\Delta$ CRLs as they are also signed by the CA's. 2 More information about the LDAP services can be found in EID-DEL-006 Component