



Project	EID	
Subject	Middleware Architecture Document	
Version	V1.0	
Date	06/03/2003	
From	Zetes	Patrick Andries
Nr of Pages	7	

I Document Control

I.1 Issue/Review Cycle

Dit document wordt uitgegeven door Zetes, en bij fundamentele wijziging ter review voorgelegd aan de betrokkenen.

I.2 Distribution List

Organisation	Name
Zetes	Johan Rommelaere Alex Driesen Bart Symons Patrick Andries
FedICT	Bart Sijnave
RRN	Daniel Pedoux

I.3 Version Log

Version	Date	Prepared By	Comments
1.00	06/03/2003	Zetes / P Andries	Initiated the document

I.4 Changes since previous issues



Changes in

None, this is the first version of the document.

I.5 Changes Forecast

None foreseen.

II Purpose

The purpose of this document is to outline the architecture of the middleware software. This software solution sits between the Belgian Electronic Identity card and the applications.

III Table of Contents

I	Document Control	1
I.1	Issue/Review Cycle	1
I.2	Distribution List	1
I.3	Version Log	1
I.4	Changes since previous issues	1
Changes in	2	
I.5	Changes Forecast	2
II	Purpose	2
III	Table of Contents	2
IV	Architecture	3
IV.1	Introduction	3
IV.2	Interfaces	4
The Crypto API interface	4	
The PKCS#11 interface	6	

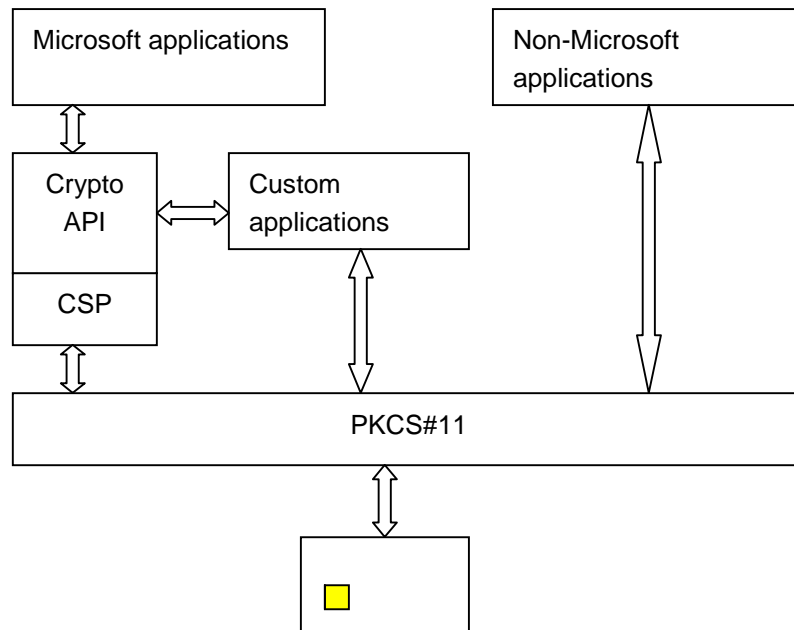
IV Architecture

IV.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.



Assumptions

It is assumed that the user of this document has a working knowledge of cryptographic operations like digital signatures, hashing operations, key material ...



IV.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).

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.

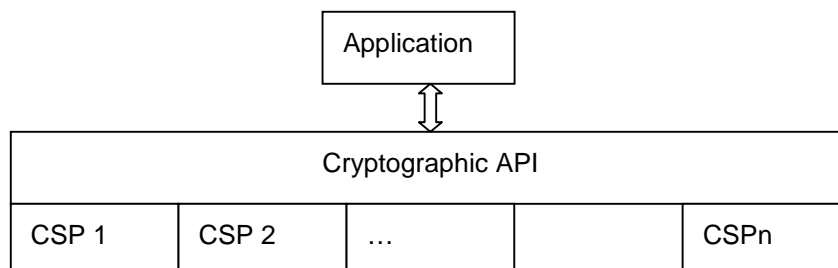
The CSP part of the middleware establishes the link between the abstract CryptoAPI and the underlying PKCS#11 interface. The developer will never call any of the functions of the CSP directly but through the CryptoAPI.

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.

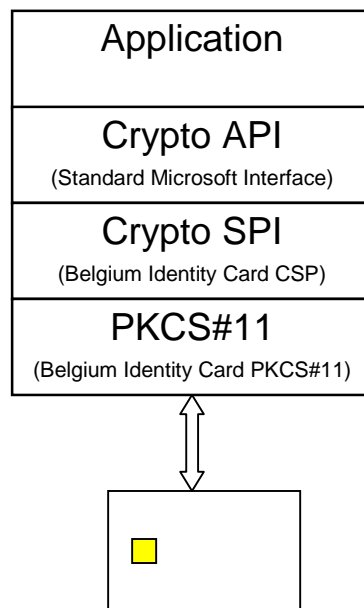
IV.2.1.1 CSP High level architecture

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:





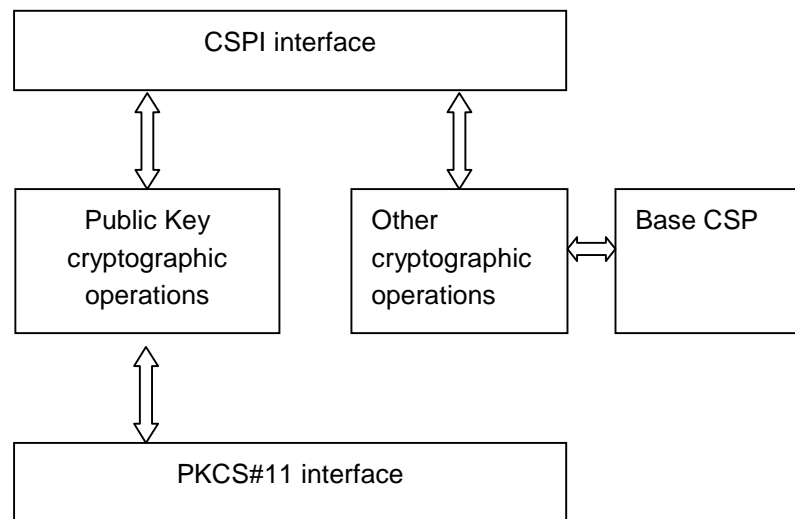
Typically each CSP orients itself to one specific smartcard type. This is because most of the time the CSP is provided by the smartcard vendor itself. The Belgian Identity Card middleware has used a different approach. In order to give the issuer of the identity card (i.e. the government) the maximum 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).

IV.2.1.2 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.

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.

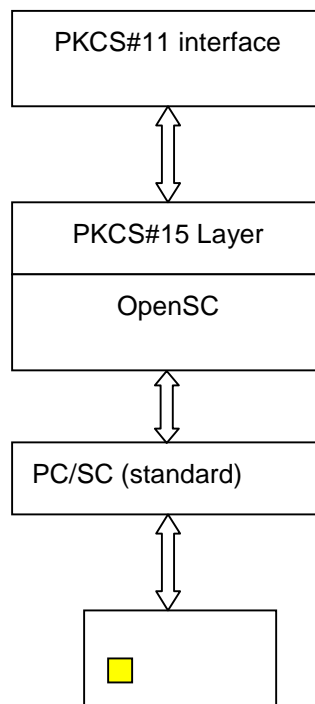
A detailed description of this interface can be found on the website of RSA Laboratories (<http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/>).

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.

IV.2.1.3 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.