Functional Documentation for EIDVirtual

Version 1.0

Prepared by: "Vincent Le Toux"

Date: 21/02/2013

Table of Contents

Revision History

This section records the change history of this document.

Name	**Date**	**Reason For Changes**	**Version**
Vincent Le Toux	21/02/2013	Creation	1.0
Thierry Martin	28/03/2013	Update	1.1
Frédéric Bourgeois	04/04/2013	Update	1.2

Description

EIDVirtual is a software used to emulate a smart card using a removable device (e.g. USB key). The software was design to support all the common smart card scenarios like :

- Smart card logon using Active Directory
- Smart card logon using EIDAuthenticate
- SSL Authentication using Internet Explorer (or any CAPI compliant browser)
- Digital Signature of emails using Outlook
- Digital Signature of documents using Word
- Encryption of documents using EFS or Bitlocker

System Specifications

Operating system supported are :

- Windows XP, Vista, Seven, 8
- Windows Server 2003, 2008, 2012

On Windows XP and Windows 2003, the "Microsoft Base Smart Card Cryptographic Service Provider Package" must be installed before.

Note : only the V5 specification of the minidriver is supported on Windows XP and Windows 2003.

Hardware

The removable device must be accessed on a read/write file system. This excludes CD-ROM (DVD-ROM, bluray, …) and read protected device. Memory cards or USB keys can be used.

.

External System Dependencies

This software depends on certain external systems to complete some or all of its tasks. This section lists those dependencies explicitly:

External System	Dependencies on that System
Windows Installer 3.0	None
Base Smart Card CSP	Windows 2003 or Windows XP SP1
UMDF Framework 1.9	Windows 2003 or Windows XP SP2

Base Smart Card CSP : http://www.microsoft.com/en-us/download/details.aspx?id=4670

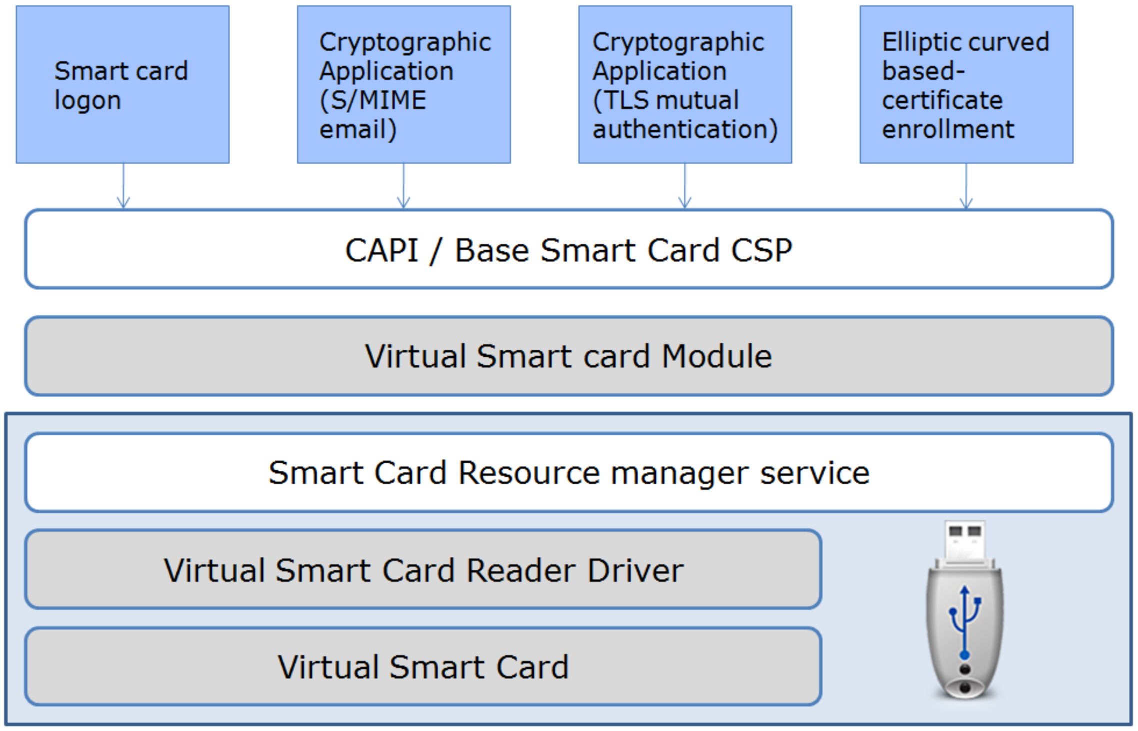
UMDF Framework 1.9: http://support.microsoft.com/kb/970159

Architecture overview

There are two ways possible to emulate a smart card in Windows. The first one is to write a full CSP (Cryptographic Service Provider), and the second one is to implement only a subset by using the existing CSP named "Base Smart card CSP". However, the Base Smart Card CSP requires a handle to the smart card resource manager which requires itself to have a virtual smart card reader. A virtual smart card reader requires to write a driver, either using kmdf (in kernel mode) or umdf (in user mode).

Because the smart card logon monitors smart card insertion and removal, writing only a CSP was not a solution. That's why it was decided to use the "Base Smart Card CSP" and to write 2 drivers :

- The virtual smart card minidriver for the Base CSP
- The virtual smart card reader for the smart card resource manager.


Smart card logon
Cryptographic Application (S/MIME email)
Cryptographic Application (TLS mutual authentication)
Elliptic curved based-certificate enrollment
CAPI / Base Smart Card CSP
Virtual Smart card Module
Smart Card Resource manager service
Virtual Smart Card Reader Driver
Virtual Smart Card

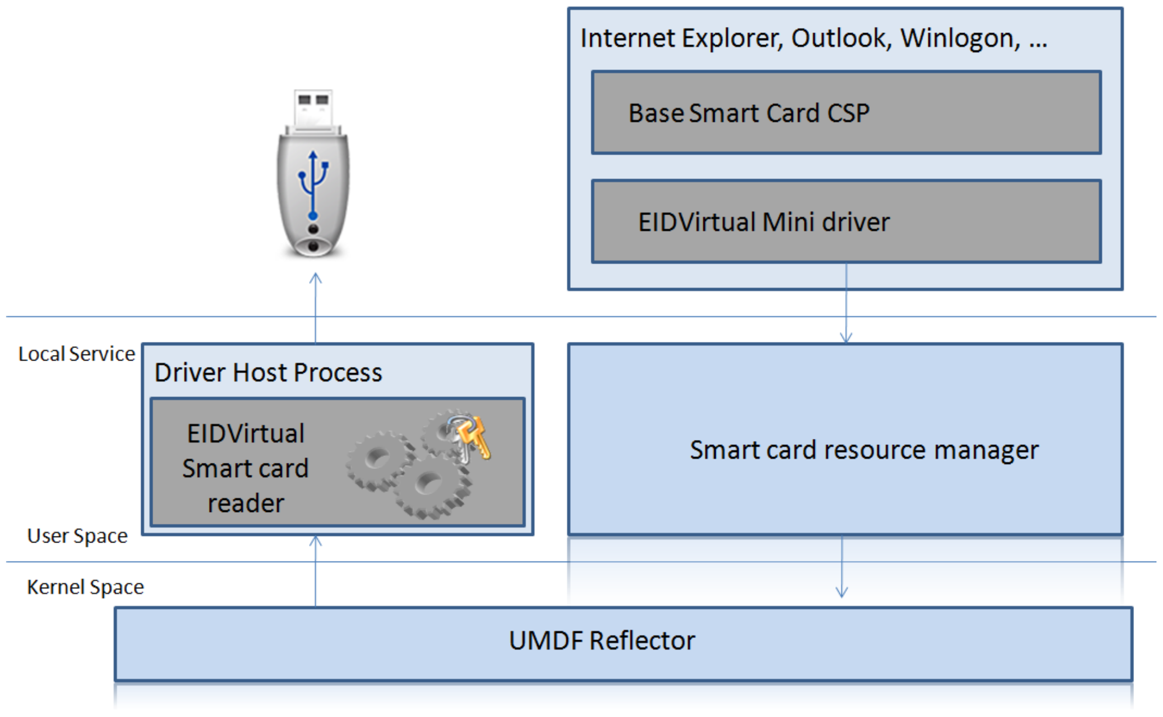
Because the minidriver runs under the current application context, it was decided for security reasons to perform the cryptographic operations inside the virtual smart card reader driver.

The smart card reader driver is a UMDF driver. It runs as a service under the UMDF driver host process running with the local system account. All IOCTL (driver commands) must be completed within one minute else the UMDF driver host process is terminated. It makes live reverse engineering difficult to perform and improve the security of the solution.

The choice of the UMDF framework ensures that a BUG in the virtual smart card reader won't produce a crash : in case of failures, the UMDF process is restarted.

The IOCTL sent by the smart card resource manager go into the kernel space and is redirected to the UMDF driver host process by a system driver named "UMDF Reflector". This component is part of the UMDF framework.

The following schema describe the middleware and the associated security context between the end application and the USB key.



Files

EIDVirtual is made of the following files :

- EIDVirtualSmartCardReader[32|64].dll which is a UMDF driver simulating a smart card reader ("EIDVirtual Smart card reader")
- EIDVirtualSmartCardMinidriver[32|64].dll which is a Base Smart card CSP minidriver ("EIDVirtual Mini Driver")
- EIDVirtualWizard.exe which is the program design to “format” a new smart card
- EIDVirtualManager.exe which is a program design to manage the certificates and the cryptographic container
- EIDVirtualActivation.exe which is a program design to activate the software
- EIDInstall.exe which is the installer

Virtual smart card

Answer To Reset (ATR)

An ATR is a sequence identifying uniquely a type of smart card. It is used by the operating system to associate a smart card to its driver.

The ATR of the virtual smart card is: 3B 8C 01 4D 79 53 6D 61 72 74 4C 6F 67 6F 6E A5

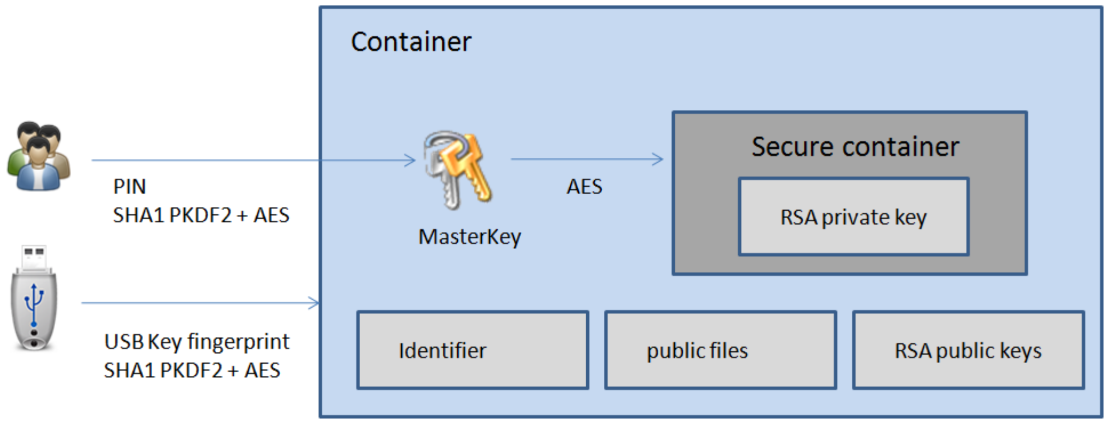
The technical characteristics of this sequence is :

TS = 0x3B	**Direct Convention**
T0 = 0x8C	**Y(1): b1000, K: 12 (historical bytes)**
TD(1) = 0x01	**Y(i+1) = b0000, Protocol T=1**
Historical bytes	**4D 79 53 6D 61 72 74 4C 6F 67 6F 6E (MySmartLogon)**
TCK = 0xA5	**checksum**

It is defined as a smart card using the T1 communication protocol.

Data storage

The container is encrypted with a AES 256 by key derived from a USB Key fingerprint. A copy of the container to another device won't be read by the virtual reader because the fingerprint of the key won't match the key used to encrypt the container. The key derivation mechanism (SHA1 PKDF2 with 4096 rounds) prohibits brute force attacks.



RSA private key are protected by a master key, itself encrypted by a key derivate from the PIN. The key derivation algorithm is SHA1 PKDF2 with 4096 rounds.

Here is an estimation of the time required to brute force offline the container a PIN if the container key is known :

Password strength	**Time**
4 number digits	1 hour
8 number digits	19 months
4 letters/numbers digits	3 months
4 digits with symbols	1 year, 4 months
8 digits with symbols	124 millions years

PIN attempts check

The PIN attempts cannot be stored in the secure container because the container could be saved before the attempt and erased after the login attempt or simply write protected.

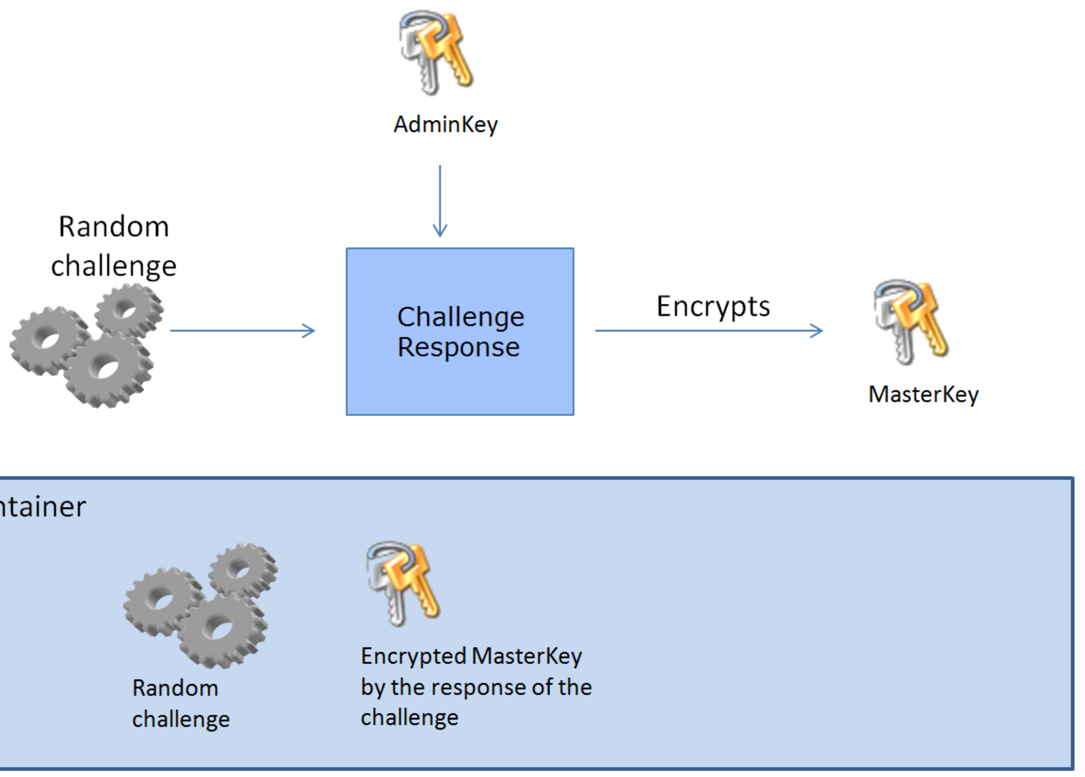
Moreover the user PIN have to be unblocked by the Admin PIN which prevent the deletion of the container.

The attempts are saved in memory by the smart card reader and reset after a reboot or after a one hour timeout.

Because the user must be able to erase an existing container, a fingerprint of the container is stored next to the PIN attempts counter in the virtual reader.

Admin Key mechanism

Because the container is not hardware protected, we prepared a design where the admin key is not stored in the secure container.



Even with the master key or the response know, the admin key cannot be recovered. However this design is vulnerable to replay attacks. That’s why we are not recommending to use an Admin Key.

Virtual Smart Card limitation

The following paragraph defines the limitation of the virtual smart card as expressed by the smart card mini driver requirements.

Functionality	Limitation
Maximum number of files	65536
Maximum file size	65536 bytes
Maximum number of container	256
Maximum number of root certificate stored	50