Université catholique de Louvain

ePassport Viewer
User Manual

Version 0.2c

Jean-Francois Houzard – jhouzard@gmail.com
Olivier Roger – olivier.roger@gmail.com

May 2009

INFORMATION SECURITY GROUP

`http://sites.uclouvain.be/security/epassport.html`
`http://code.google.com/p/epassportviewer/`

Contents

Chapter 1

Introduction

1.1 Purpose

This program reads ePassports and display the results on screen. It allows any ePassport holder to view its content.

1.2 Motivation

For our master thesis we developed an Application Programming Interface (API) capable of reading the ePassport Radio Frequency Identification (RFID) tag. It is called pyPassport[1]. This API is based on the DOC 9303[2] standard from International Civil Aviation Organization (ICAO). Additional information about our work can be found at `http://sites.uclouvain.be/security/epassport.html`.
This application is a simple example of usage of this API. We wanted it to be cross-platform, simple to install and to use.

1.3 Disclaimer

Before using pyPassport, you must be sure that you are allowed to read the contactless tag of your passport, according to the laws and regulations of the country that issued it.

[1]`http://code.google.com/p/pypassport/`
[2]`http://www2.icao.int/en/MRTD/Pages/Doc9393.aspx`

1.4 Application Features

ePassport Viewer:

1. reads the passport and displays the results.
2. can perform security mechanisms as explained in Doc 9303 standard.
3. can save data in many file formats (PDF, XML, Binary).
4. can extract data contained in the RFID tag (Face, Signature, ...).
5. can analyze an ePassport and report specificities generating a “Finger Print”.
6. is cross platform (Windows, Linux, Mac).
7. is open-source.

1.5 History

0.2c beta (2009-05-18)
Add an option to do a fingerprint of the passport.

0.1 beta (2009-05-01)
First public release. GPL 3 license.

Chapter 2

Technical Details

This program is written in Python[1].
Similar API exists in other languages like Java[2], and C++[3].
More information about similar works can be found on the official website.

2.1 Supported Platforms

ePassport Viewer can be executed on any platform capable of running Python 2.5/2.6 (and C based module) and PIL[4]. It has already been successfully run on Windows, Linux (Ubuntu) and Mac OSX.

2.2 Required Material

To read your passport with this application, an RFID reader compliant with the PS/SC standard is needed. The application has been successfully tested with the following readers:

- ACR[5] 122;
- OMNIKEY[6] 5321. The reader must be plugged in before running the application.

[1] http://www.python.org/
[2] http://www.jmrtd.org/
[3] http://www.waazaa.org/wzmrtd/index.php
[4] http://www.pythonware.com/products/pil/
[5] http://www.acs.com.hk/
[6] http://www.omnikey.com/

2.3 License

ePassport Viewer is released under GNU/GPL 3 license terms. Full text is available at `http://www.gnu.org/licenses/gpl-3.0.txt` or in the LICENSE file present in the downloaded archive.

2.4 Download

All files are available in source and binary formats on the project website `http://code.google.com/p/epassportviewer/`. The next section contains installation instructions for the targeted platforms.

2.5 Installation

Installation has been made as easy as possible. The tool is available either as a stand-alone binary (no installation required) or as source files. The following sections explained how it works.

If you encounter any problem, the solution might already be in the FAQ at the end of this document. Otherwise please visit the project website and leave a complete description of the issue.

2.5.1 Binaries

A stand-alone application is available for the following platforms:

1. Windows (zip/exe)
2. Mac OSX (dmg/app)

Follows the procedure to use them.

Windows

- Unzip the archive.
- Double-click on the `epassportviewer.exe` to launch the application.
- If not yet installed on your computer, you may also need the Visual C++ Redistributable Package[7] to use the OpenSSL version included in the archive.

[7]It is available on Microsoft website.

Mac OSX

- Mount the DMG
- Drop the application in your application folder
- Use it like any other application

Linux

Debian and RPM packages, allowing automated installation, will be released when the application reaches a stable state.

2.5.2 Sources

Dependencies

ePassport Viewer is built on top of third-party open-source libraries. All of them are mandatory.

ePassport Viewer also uses third-party optional open-source tools to perform specific processing. If not installed, capabilities of ePassport Viewer are reduced.

Mandatory

The following packages are required to run the application.

- Python 2.5 or 2.6 with Tkinter (usually included).
- Python packages: pypassport[8], ReportLab[9], PIL[10].

Python packages can be installed using the code sharing Python platform PyPI[11] using the following command:

```
Python/scripts$ easy_install package_name
```

Remark 1: PyPI
PyPI is very convenient but not every package can be installed on all platforms using it. In that case other installation methods must be considered. Installation from source is certainly a good solution.

[8]http://code.google.com/p/pypassport/
[9]http://www.reportlab.org/
[10]http://www.pythonware.com/products/pil/
[11]http://pypi.python.org/pypi

For Windows, user binary build exists for all packages.
On Mac OSX you can use i-Installer[12], fink[13] or macport[14].

Remark 2: ReportLab
ePassport Viewer does not require the ReportLab library to be installed with the renderPM module.

Optional

To complete the installation, two additional packages should be installed:

1. OpenSSL[15] for certificate verification (already included in Mac OSX);
2. GeoJasper[16] to manipulate images in JPEG2000 format.

Running the application

ePassportViewer does not require any installation. Simply run the entry point located in the `ePassportViewer.py` file:

```
epassportviewer-x.y.z$ python ePassportViewer.py
```

[12]`http://ii2.sourceforge.net/`
[13]`http://www.finkproject.org/`
[14]`http://www.macports.org/`
[15]`http://www.openssl.org/`
[16]`http://dimin.m6.net/software/geojasper/`

Chapter 3

Usage

3.1 Start the application

First, you need to install the application. This manual supposes the application is running. Please refer to the previous section if you cannot start it.

The figure 3.1 shows the application started for the first time.

3.2 Enter the MRZ

The machine readable zone[1] (MRZ) is usually composed of two lines (sometimes three) visible on the data page of the passport. To communicate with the ePassport, the application needs the second line to be inputted entirely (44 characters) in the text fields. Figure 3.2 shows a fictive MRZ inputted in the application.

The history button shown on figure 3.2 keeps track of previously inputted MRZ to avoid error-prone and time-consuming input.

3.3 Reading the ePassport

Entering the MRZ is the only mandatory step to read an ePassport. To start the reading, press the Return key or hit the Read button.
A dialog box will pop-up and ask you to drop your passport on a RFID reader. When done, the application will read the ePassport entire content and perform the selected security mechanisms, i.e. active and passive authentication.

During the reading, the dialog box shown in Figure 3.3 indicates to the user the advancement and current task of the operation. Once a data section is read

[1] `http://en.wikipedia.org/wiki/Machine-readable_passport`

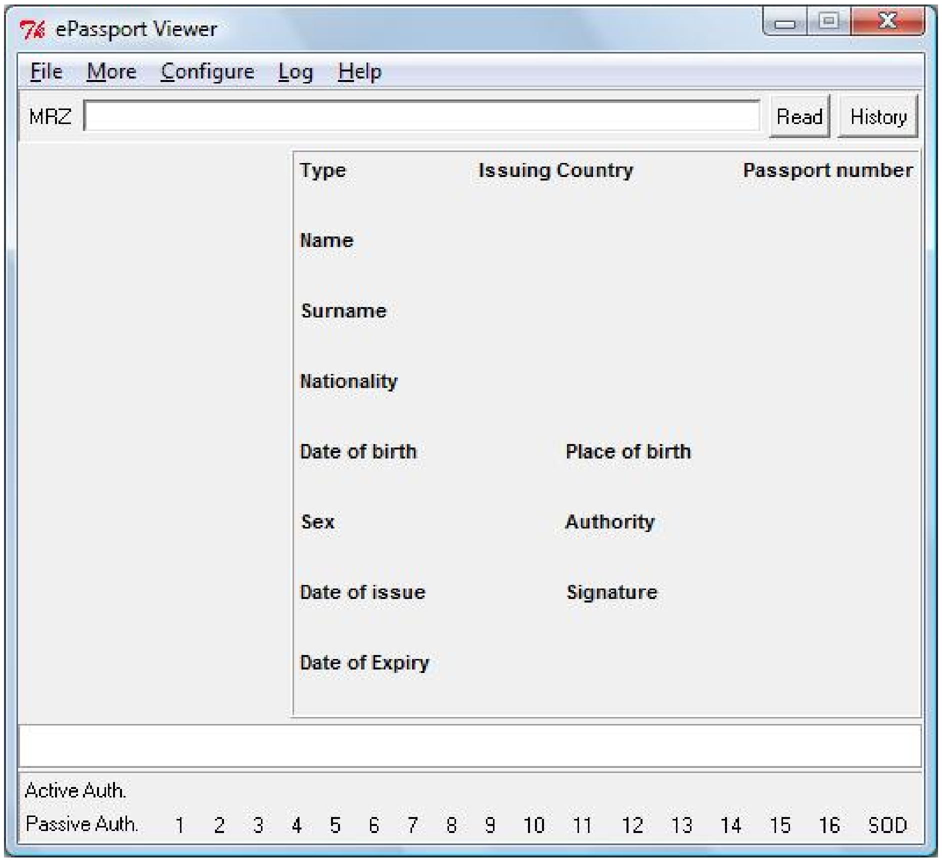
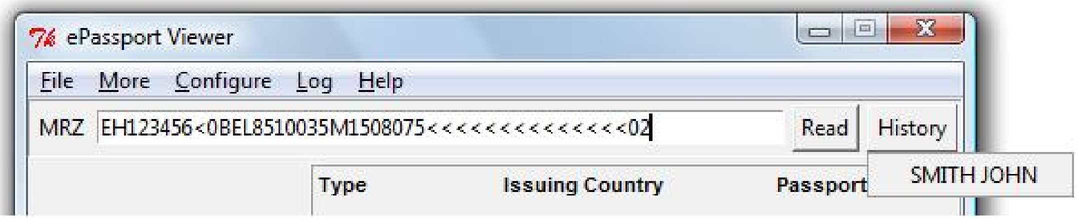


Figure 3.1: ePassport Viewer has just started

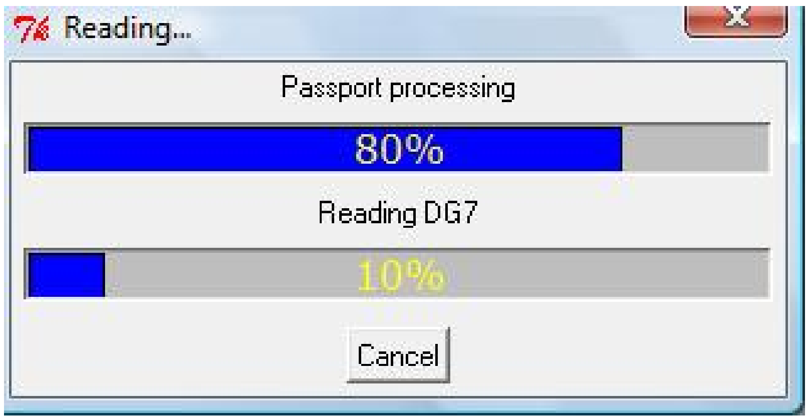


MRZ EH123456<0BEL8510035M1508075<<<<<<<<<<<<<<02

SMITH JOHN

Figure 3.2: Click on the History button to see the last inputted MRZ

the main window updates its content accordingly.



Passport processing

80%

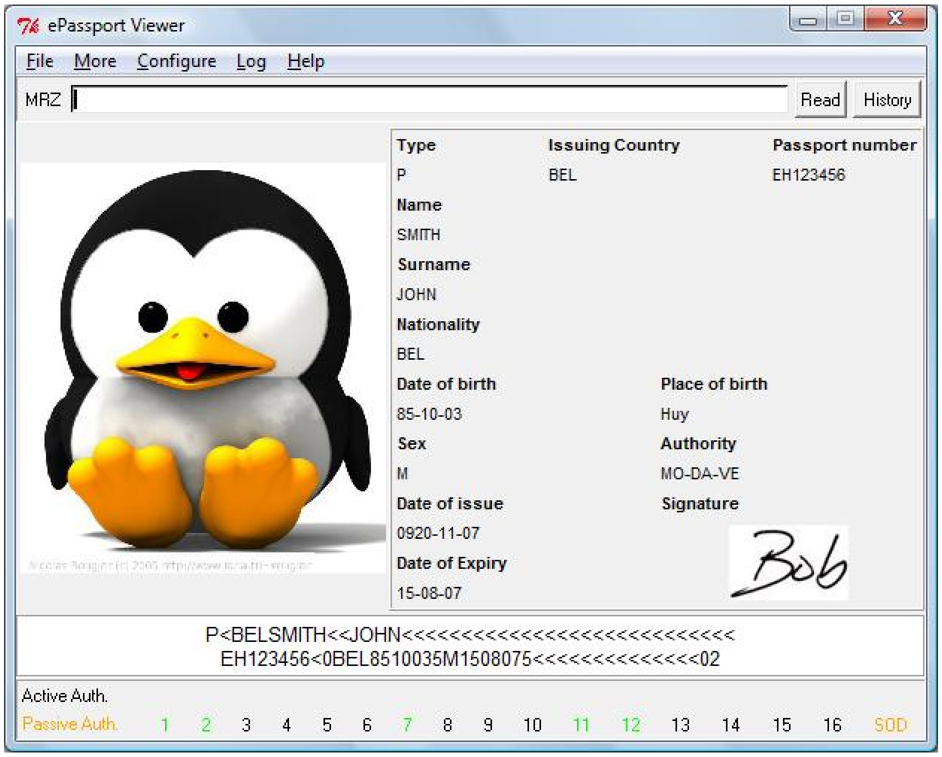
Reading DG7

10%

Figure 3.3: The dialog shows the reading progress

3.4 Analyzing the displayed results

Figure 3.4 shows the result of the author passport reading.


ePassport Viewer
File More Configure Log Help
MRZ
Read
History
Nicolas Bougier (c) 2005 http://www.tosa.fr/~nbugier

Type	Issuing Country	Passport number
P	BEL	EH123456
Name		
SMITH		
Surname		
JOHN		
Nationality		
BEL		
Date of birth	Place of birth	
85-10-03	Huy	
Sex	Authority	
M	MO-DA-VE	
Date of issue	Signature	
0920-11-07	Bob	
Date of Expiry		
15-08-07		

```
P<BELSMITH<<JOHN<<<<<<<<<<<<<<<<<<<<<<<<<<<<
EH123456<0BEL8510035M1508075<<<<<<<<<<<<<<02
```

Active Auth.
Passive Auth. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 SOD

Figure 3.4: Result of a successful reading

Picture and data are presented like in the paper version of the passport. The ePassport contains more information than the ones displayed. The entire list of collected data can be found using the More / Additional Data menu.

On the bottom, a security panel displays the information concerning the security mechanisms. The Active Authentication verifies the authenticity of the ePassport while the Passive Authentication verifies the integrity of each Data Group. Data Groups are sets of interrelated data. They are identified by a number between 1 and 16.

Each color has a meaning as explicit as possible:

Black
Verification not performed. This usually happens because the feature has either been disabled in the menus or because required data are not present.

Green
Verification has been successfully performed.

Orange
Verification could not be performed. The reason is written in the log file (accessible from the **Log** menu). Usually because OpenSSL or the certificates are missing.

Red
Verification has been performed but failed.

The article "Belgian Biometric Passport does not get a pass..."[2] gives more information about ePassport security mechanisms.

Figure 3.4 can be interpreted as follows: the Active Authentication mechanism has not been performed (disabled). Passive authentication has been performed but some Data Groups are not present on the RFID tag (DG3, DG4,...).
The last piece of information concerns the Security Data Object (SOD) which could not be verified because the root certificate was not provided.
This means that the Passive Authentication has verified the integrity of the Data Groups based on the signature available in the RFID tag.

[2] `http://www.dice.ucl.ac.be/crypto/passport/index.html`

3.5 Menus

Some additional features are available via the application menu. This section will present them briefly.

File performs operation on files:

- **Load** a dump previously made with the application or a compatible one;
- **Save** all or some of the data from the read ePassport;
- **Clear** the application, resets all fields to blank;
- **Exit** the application.

More gives more information about the ePassport:

- **Additional data** shows all data present in the tag using their tag / value pair;
- **Fingerprint** analyzes particularities of the ePassport:
  - Size of each Data Group;
  - Security mechanisms available and their order;
  - and some more.

  Fingerprints can be compared to deduce information.

Configure sets particular options:

- **Reader** allows the user to select one of the connected RFID readers to perform the reading;
- **Security** gives the choice to the user to perform Passive and Active Authentication;
- **Reading mode** allows the user to choose the reading mode he wants;
- **OpenSSL** is a pointer to the OpenSSL executable. If you already have OpenSSL installed on your computer you should change this option;
- **Export Path** points to the directory where files will be saved. If not set, it will be asked on the first attempt to save;
- **Certificate Directory** points to the directory where root certificates are located. These files are somewhat hard to find. JMRTD[3] provides some of them. Before being used, certificates have to be imported into the right format. The import is done automatically when changing the directory and can be forced with the import submenu item.

Log chooses and displays the content of the log file:

[3]http://www.jmrtd.org/

ePassport API logs all verbose action of the ePassport;

Secure Messaging logs the ciphering of all transmitted APDU;

APDU logs all APDU sent and received;

Fingerprint logs ePassport specificities;

See log file opens a window to see the log file content.

Help links to various information you might need:

Help content is an offline help;

Manual links to this manual;

Website links to the website page;

About gives information about the authors and license.

Chapter 4

FAQ

Why is it called ePassport Viewer ?

Because it allows the user to see the digital content of the new biometric passports.

Why choose Python ?

ePassport Viewer uses the pyPassport API which is written in Python. Python provides: efficiency, readability, portability, large standard libraries.

What smart card readers are supported by ePassport Viewer ?

At the moment pyPassport handles generic PCSC readers. Passport readings have been successfully performed with the following smartcard readers:

- Omnikey 5321;
- ACS ACR 122.

For compatibility reasons, a specific module has been written for the latter.

My reader is not supported, how can I add support for it ?

You will have to modify the pyPassport API. Essentially the readerFactory class. As for the ACS ACR 122, a new class must implement the abstract reader class. Having the reader specs might be very precious for this operation.

Why are all security labels always orange ?

If all security labels are always orange, it means that OpenSSL is not available. if you use ePassport Viewer stand-alone for win32 then OpenSSL is embedded with the archive, if you encounter troubles, install redistributable Microsoft Visual C++ 2008[1]

[1]`http://www.microsoft.com/downloads/details.aspx?FamilyID=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF&displaylang=fr`

On startup I got a message concerning smart card, what can I do?

If you get an `smartcard.pcsc.PCSCExceptions.EstablishContextException` it means your smart card service is not running.

The smartcard service behavior depends on the platform, each one is different. The solution for each platform is described right below.

Start Smart Card service on Windows

Remark Since version 0.2b, the application tries to start the service for you.

On Windows platforms the service can be started using the command line or administration tools from the control panel.

Command line

Scripts are located in the `$Windows\System32 directory`.
The following command starts the smart card service:

```
C:\WINDOWS\system32>net start scardsvr
```

Control panel (tested on Windows XP)

In the control panel, select administration tools and then services. Search for an entry called scardSrv. You can start it with a right-click and selecting start. In the properties, you can also change it to start automatically on Windows startup.

Start Smart Card daemon on Linux

Depending on your Linux distribution, the command to start the smart card daemon can change.
On Ubuntu you can use the following command to start it :

```
/etc/init.d/pcscd start
```

Start Smart Card daemon on Mac OSX

The pcscd daemon seems to start automatically when a smart card reader is detected and stops when the reader is removed.
This means you might need to have a smart card reader to launch the application. No workaround has been found yet for this problem.