

INGI2144 Project (2011-2012)
RFID Loyalty Card for the LLN Sandwich Shops

Nicolas Maître, Bernard Paulus, Arnaud Theismann

December 12, 2011



UCL
**Université
catholique
de Louvain**

Contents

1	Introduction	2
2	Software Implementation	2
2.1	Architecture	2
2.2	External tools	3
2.3	Protection against human mistakes and bugs	3
3	Security analysis	4
3.1	Actual weaknesses of the system	4
3.2	Towards a better solution	4
3.2.1	With same technology and file/data structure	4
3.2.2	With same technology only	4
4	Conclusion	4

1 Introduction

The aim of this project was to design and implement a RFID-based system able to manage electronic loyalty cards for sandwich vendors in Louvain-La-Neuve.

In this document, we first present the architecture of our software and an overview of its implementation, then we give a security analysis of the whole system, along with ideas to improve it.

2 Software Implementation

2.1 Architecture

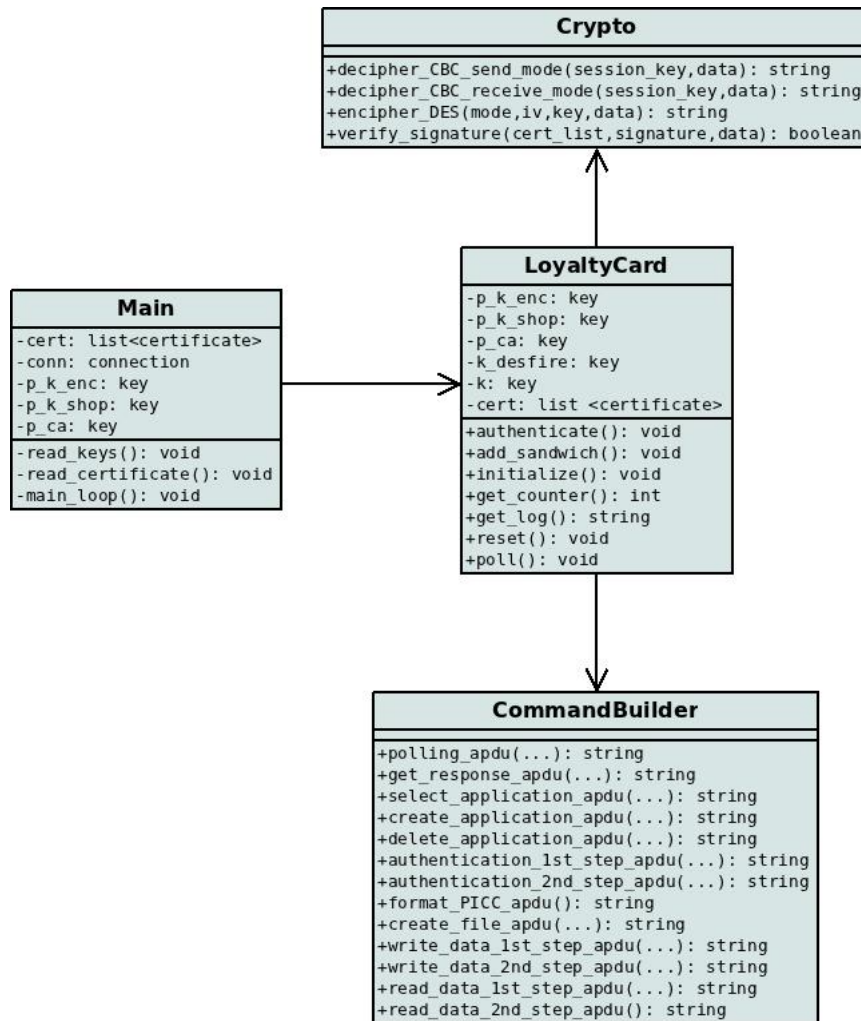


Figure 1: Architecture of the software

As depicted on figure 1, our python application is divided in 4 modules. **LoyaltyCard** class aims at representing an RFID loyalty card and offers methods to interact with.

Main is responsible to create and initialize instances of **LoyaltyCard** and offers to the user a command prompt to interact with the DesFire tag. Moreover, it is up to this module to load from different files the keys and certificates needed for the management of a loyalty card.

The **Crypto** module offers high-level cryptographic methods that are used by the **LoyaltyCard**

module. These methods essentially call other methods from external crypto libraries (see next section).

Finally, `CommandBuilder` is a low-level module that allows to build APDUs to communicate with the tag.

2.2 External tools

In order to communicate with the RFID card, we have used `pyscard` (<http://pyscard.sourceforge.net>) which is a python wrapper for `pcsc-lite`.

`PyCrypto` library (<https://www.dlitz.net/software/pycrypto>) has been used as it provides methods for dealing with DES/3DES and RSA encryption and decryption. This python library is very interesting from a performance point-of-view since speed-critical operations like ciphers and hash functions are actually written in C.

In addition to this, since `PyCrypto` is not able to read certificates in X.509 format, we have used `M2Crypto` (<http://chandlerproject.org/bin/view/Projects/MeTooCrypto>) for this purpose. Retrieving public keys from certificates and verifying signatures is therefore handled thanks to this library which is in fact a wrapper for `OpenSSL`.

2.3 Protection against human mistakes and bugs

A few measures have been implemented to deal with bugs and human mistakes. First, it is impossible for the user to crash the application by performing a command when there is no RFID tag in the reader's field. Indeed, the presence of a tag is checked after every user input. If no tag is detected, a timer of 3 seconds is launched at the end of which a warning message is printed to the standard output. No further operation can be made as long as the method `poll()` does not return.

Similarly, the presence of an RFID reader is checked at application start-up. The command prompt is enabled only after the system has detected a reader. The presence of a reader is then checked every second to prevent the user from performing an operation when there is no reader. If such a case occurs, a warning message is printed to the standard output.

Finally, we ask confirmation when the user wants to reset the loyalty card to factory settings. This prevents him from erasing the whole tag memory by mistake.

3 Security analysis

3.1 Actual weaknesses of the system

3.2 Towards a better solution

3.2.1 With same technology and file/data structure

3.2.2 With same technology only

4 Conclusion