

On Development of Inspection System for Biometric Passports Using Java

Luis Terán and Andrzej Drygajlo

Speech Processing and Biometrics Group
Swiss Federal Institute of Technology Lausanne (EPFL)
Switzerland
{luis.terantamayo, andrzej.drygajlo}@epfl.ch
<http://scgwww.epfl.ch/>

Abstract. Currently it is possible to implement Biometric Passport applets according to ICAO specifications. In this paper, an ePassport Java Card applet, according to ICAO specifications using the Basic Access Control security, is developed. A system for inspection of the ePassport applet, using Java, in order to test its functionalities and capabilities is also implemented. The simulators, which are developed in this paper, can display the communication between the inspection system and the Java Cards, which could be real or emulated cards.

Keywords: Biometrics, ePassport, Java Card, Inspection System, Basic Access Control.

1 Introduction

Over the last two years, Biometric Passports have been introduced in many countries to improve the security in Inspection Systems and enhance procedures and systems that prevent identity and passport fraud. Along with the deployment of new technologies, countries need to test and evaluate its systems since the International Civil Aviation Organization (ICAO) provides the guidelines, but the implementation is up to each issuing country. The specific choice of each country as to which security features to include or not include makes a major difference in the level of security and privacy protection available.

Table 1. ePassport Deployments

Country	RFID Type	Deployment	Security	Biometric
Italy	14443	2006	Passive, Active Authentication, BAC	Photo
U.S.	14443	2005	Passive, Active Authentication, BAC	Photo
Netherlands	14443	2005	Passive, Active Authentication, BAC	Photo
Germany	14443	2005	Passive, Active Authentication, BAC	Photo

In this paper we focus in the development of an Inspection System for Biometric Passports. The standards and specifications of Machine Readable Travel Documents (MRTD's) are described in Section 2. The technologies used for the implementation of the inspection system for Biometric Passports, and a description of the implementation and its utilization are described in Sections 3. The conclusions are drawn in Section 4. Table 1 shows some examples of ePassport implementations and the security features selected.

2 ICAO Specifications

The Machine Readable Travel Document (MRTD) is an international travel document, which contains machine-readable data of the travel document holder. MRTDs facilitate the identification of travelers and enhance the security levels. MRTDs are developed with the assistance of the ICAO's Technical Advisory Group for Machine Readable Travel Documents (ICAO TAG/MRTD) and the ISO Working Group 3 (JTC1/SC17/WG3).

MRTDs such as passports, visas or other travel documents, have the following key issues to be considered: Global Interoperability, Uniformity, Technical Reliability, Practicality, and Durability. ICAO has elaborated different technical reports. The main technical reports, which are part of ICAO specifications and considered in this paper, are: Logical Data Structure Technical Report and Public Key Infrastructure Technical Report.

2.1 Logical Data Structure

The Logical Data Structure (LDS) technical report specifies a global and interoperable data structure for recording identity details including biometric data. The data stored in the LDS is read electronically and designed to be flexible and expandable for future needs. A series of mandatory and optional elements has been defined for LDS, which are used in MRTDs. The use of biometric data is optional except the use of the encoded face.

Each data group is stored in different Elementary Files (EF's). The structure and coding of data objects are defined in ISO/IEC 7816-4 [1]. Each data object is encoded using Tag - Length - Value (TLV). Any data object is denoted {T - L - V} with a tag field followed by a length field encoding a number, which represents the size of the value field. If the size is equal to zero, the data field is absent.

A constructed data object is denoted {T - L - V {T1 - L1 - V1}...{Tn - Ln - Vn}}, which represent a concatenation and interweaving of data objects. This type of structure is used in Data Groups containing more than one value field, which are preceded by specific Tag and Length field. Figure 1 shows the complete structure of LDS, which includes mandatory and optional data elements defined for LDS (version 1.7)

- The inspection system computes the SHA_1 value of *MRZ_information*.
- The inspection system uses 16 bytes (most significative) from the hash value of *MRZ_information* in order to compute $K_{IFD/ICC}$

$$K_{IFD/ICC} = \text{Trunc}_{16}(\text{SHA}_1(\text{MRZ_information}))$$

Compute session keys from seed key $K_{IFD/ICC}$

- The inspection system concatenates the key seed $K_{IFD/ICC}$ with a value c in order to compute K_{ENC} and K_{MAC} ($c = 1$ for K_{ENC} and $c = 2$ for K_{MAC}) and takes the hash value of it.

$$\begin{aligned} \text{HASH1} &= \text{Trunc}_{16}(\text{SHA}_1(K_{IFD/ICC} \parallel 00\ 00\ 00\ 01)) \\ \text{HASH2} &= \text{Trunc}_{16}(\text{SHA}_1(K_{IFD/ICC} \parallel 00\ 00\ 00\ 02)) \end{aligned}$$

- The first 8 bits of HASH1 is set as $K_a(\text{ENC})$ and the last 8 bits of *HASH1* are set as $K_b(\text{ENC})$ in a two keys triple DES cryptographic algorithm.
- The first 8 bits of HASH2 is set as $K_a(\text{MAC})$ and the last 8 bits of *HASH2* are set as $K_b(\text{MAC})$ in a two keys triple DES cryptographic algorithm.

Authentication and Establishment of Session Keys

- The inspection system requests an 8 byte challenge (RND.ICC) from ICC chip, and generates a random 8 bytes (RND.IFD) together with a random 16 bytes triple DES key (K_{IFD}).
- The inspection system concatenates RND.ICC, RND.IFD, and K_{IFD} .

$$S = \text{RND.IFD} \parallel \text{RND.ICC} \parallel K_{IFD}$$

- The inspection system sends to MRTD the encrypted version of S using K_{ENC} and K_{MAC} using *MUTUAL_AUTENTICATE* function of ISO7816-4.

$$\text{cmd_data} = E[K_{ENC}](S) \parallel \text{MAC}[K_{MAC}](E[K_{ENC}](S))$$

- MRTD decrypts and verifies the received data.
- MRTD computes a 16 bytes triple DES key (K_{ICC}).
- MRTD concatenates RND.ICC, RND.IFD, and K_{ICC} .

$$R = \text{RND.ICC} \parallel \text{RND.IFD} \parallel K_{ICC}$$

- MRTD sends to the inspection system the encrypted version of R using K_{ENC} and K_{MAC} using *MUTUAL_AUTENTICATE* function of ISO7816-4.

$$\text{resp_data} = E[K_{ENC}](R) \parallel \text{MAC}[K_{MAC}](E[K_{ENC}](R))$$

- The inspection system decrypts and verifies the received data.
- Both, the inspection system and MRTD, create a 16 bytes key seed $K_{IFD/ICC}$, which is the exclusive or between K_{ICC} and K_{IFD}

$$K_{IFD/ICC} = K_{ICC} \oplus K_{IFD}$$

- Both, the inspection system and MRTD, compute the 16 byte Encryption and Authentication Session Keys (KS_{ENC} and KS_{MAC}) using $K_{IFD/ICC}$ as described previously in section “*Compute session keys from seed key $K_{IFD/ICC}$* ”.

The Secure Messaging should be implemented for the communication of Application Protocol Data Units (APDUs) between MRTDs and the inspection systems. Secure Messaging has been specified according to *CWA 14890-1: 2004, Application Interface for smart cards used as Secure Signature Creation Devices* [5]. It is required for Passive Authentication and it is optional for Active Authentication. Figure 2 shows an example of Secure Messaging using session keys KS_{ENC} and KS_{MAC} computed previously.

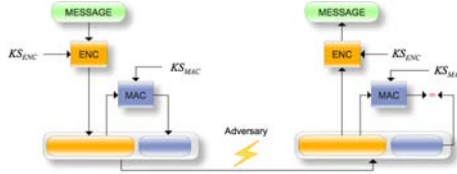


Fig. 2. Secure Messaging

3 Java Implementation

The main technologies that were considered in this paper are: Smart Cards, Java Card versions 2.2.2 and 2.2.1, and Java version 1.6 in order to provide an application and allow the user to emulate and evaluate an ePassport applet.

The Smart Card is a portable and tamper-resistant computer. It has both, processing power and memory. The physical appearance and properties of smart card are defined in the standard ISO 7816-1. Smart cards usually contain three types of memory: ROM, EEPROM, and RAM. The ROM memory is used for storing the fixed program of the card and contains the operative system routines as well as permanent data and user applications. The EEPROM memory is similar to ROM memory in that it can preserve information when the power of the memory is turned of. The main difference is that the content of this memory can be modified. We could compare EEPROM memory with the hard disk on a PC. The RAM memory is used as temporary working space for storing and modifying data.

The second technology, which is the Java Card Technology, allows Java-based applications (called applets) to run on smart cards including memory and processing capabilities, essentially Java Card Technology. The distribution of Sun Microsystems for smart cards (Java Card Development Kit V2.2.1 and V2.2.2) are used in this paper in order to develop an ePassport applet according to ICAO specifications.

The Java Card Technology defines three parts: Java Card Virtual Machine (JCVM), Java Card Runtime Environment (JCRE) and Java Card Application Programming Interface (API). The JCVM defines a subset of Java programming language. JCVM is implemented as two separated pieces. The on-card portion of JCVM includes the Java Card byte-code *interpreter*. The Java Card *converter* runs on a PC, converts and loads the class files generating a CAP (converted applet) file.

The second part, which is the JCRE, is the Java Card system that runs inside the card. It is responsible for card resource management, network communication, applet execution and security. The third part, which is the Java Card API, is a set of classes for programming smart cards according to ISO 7816 model.

Finally, the last technology used is Java. One of the most important decisions to be taken before starting the development of the prototype is the programming language. The requirements of the application to be developed are Graphical User Interface (GUI), communication with Data Base, and communication with Java Card Applet. Thus, in order to fulfill all requirements, Java Version 1.6 is used for the implementation.

3.1 JCWDE Simulator

The Java Card platform Workstation Development Environment (JCWDE) tool allows to emulate the running of an applet. JCWDE is not an emulation of JCVM, it uses the JCVM in order to emulate the JCRE. This simulator can not support some of the JCRE features like: package installation, persistent card state, firewall, transaction, transient array clearing, object deletion, applet deletion, package deletion, and package deletion.

The JCWDE simulator used in this paper, allows the user to test the ePassport applet. In order to generate the File System required by the ePassport applet, which contains the personal information of the ePassport owner, the Inspection System gets access to a data base, and the user selects one person from the list of people included in the system. After the selection, the Inspection System configures the required environmental variables, and generates the necessary files for the simulation.

When the configuration process has been performed, the user can test the ePassport applet using the information of the person selected. This simulator does not allow the use of Basic Access Control security method due to the limitations mentioned before. The JCWDE is used to read information from the emulated ePassport in an secure channel. This simulator can also display the communication generated between the emulated ePassport and the Inspection System. Figure 3, shows the architecture of the JCWDE simulator used in this paper.



Fig. 3. JCWDE Simulator

3.2 CREF Simulator

The C-language Java Card Runtime Environment (CREF) tool allows to emulate a real Java Card technology-based implementation. CREF has the ability to simulate, persistent memory (EEPROM), save and restore the contents of EEPROM.

The CREF simulator used in this paper, allows the user to test the ePassport applet. In order to generate the File System required by the ePassport applet, which contains the personal information of the ePassport owner, the Inspection System gets access to the data base, and the user selects one person from the list of people included in the system. After the selection, the Inspection System configures the required environmental variables, and generates the necessary files for the simulation.

When the configuration process has been performed, the user can test the ePassport applet using the information of the person selected. The CREF is used to read information from the emulated ePassport in an insecure channel. This simulator can also display the communication generated between the emulated ePassport and the Inspection System. Figure 4, shows the architecture of the CREF simulator used in this paper.



Fig. 4. CREF Simulator

3.3 Card Interface

The Inspection System can also test ePassport applets installed in a smart card. The Card Interface is used to read information from the an ePassport applet installed in a smart card in an insecure channel using the Basic Access Control security method. In order to communicate with the smart card we use an Alya Reader provided by COVADIS [8] which use a PCSC interface. The Card Interface can also display the communication generated between the ePassport applet installed in the smart card and the Inspection System. Figure 5, shows the architecture of the Simulator with Card interface used in this paper.

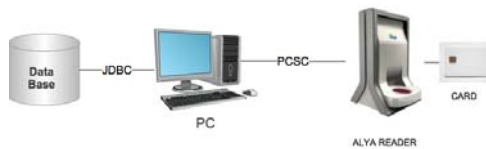


Fig. 5. Simulator with Card

4 Conclusions

In this paper it was designed an Inspection System for Biometric Passports according to ICAO specifications. The resulting system covers most of the user requirements, for instance: upload and update of information, verification, and security.

The ePassport applet was built using Java Card Development Kit version 2.2.1, which has limitations like: management of APDU commands, less cryptographic algorithms, etc.

The current system can test an ePassport applet with or without a real smart card using JCWDE and CREF simulators included in the Java Card Development Kit 2.2.2.

In order to improve the security of the system is recommended to consider a new version of Java Card Development Kit (version 2.2.2), which provides better cryptographic algorithms.

References

1. ISO. Identification cards - Integrated circuit cards with contacts - Part: 4. Organization, security and commands for Interchange. ISO/IEC 7816-4. International Organization for Standardization, Geneva, Switzerland (2005)
2. Development and Specifications of Globally Interoperable Biometric Standards for Machine Assisted Identity Confirmation using Machine Readable Travel Documents, V2.0, International Civil Aviation Organization (2004)
3. Development of a Logical Data Structure for Optional Capacity Expansion Technologies, V1.7, International Civil Aviation Organization (2004)
4. PKI for Machine Readable Travel Documents Offering ICC Read-Only Access, V1.1, International Civil Aviation Organization (2004)
5. CWA 14890-1: 2004, Application Interface for smart cards used as Secure Signature Creation Devices, Version 1 release 9 rev. 2 (December 22, 2003)
6. Chen, Z.: Java Card Technology for Smart Cards. Architecture and Programmer's Guide (2000)
7. Security and Privacy Issues in E-Passports, Ari Juele, David Molnar, and David Wagner, RSA Labs (2005)
8. ALYA reader, <http://www.covadis.ch/Alya.239.0.html>