A Grid of Java Cards^{TM*} to Deal with Security Demanding Application Domains

Eve Atallah[†] Serge Chaumette Frank Darrigade Achraf Karray[‡]
Damien Sauveron[§]
LaBRI, Laboratoire Bordelais de Recherche en Informatique
UMR 5800 – Université Bordeaux 1
351 cours de la Libération, 33405 Talence CEDEX, FRANCE.
{atallah,chaumett,darrigad,karray,sauveron}@labri.fr, http://www.labri.fr/

Abstract

More and more applications make use of hardware resources that are available all over the network at different physical places and that are the property of unknown persons or organizations. One of the major problems of such a configuration is that it requires a mutual trust between the owner of the application to be executed and the owners of the resources that execute the code. Both can be dangerous for the other. In this paper, we present the distributed secure platform that we have set up to solve this problem and some application domains where it can be helpful to enforce security.

KEYWORDS: *Java Cards, Distributed Secure Environment, Security of Applications.*

1. Introduction

Based on the multi-applicative characteristic of the Java Card technology [7, 13] and on our experience of both this technology and distributed computing, we have set up what we call the Java Card Grid project [2, 4]. This project consists in building a cluster of smart cards (*cf.* Fig. 1) and in providing a software framework for developing and managing secure applications on this cluster.

The main goals of this project are to experiment and thereafter propose innovative high level security solutions for distributed applications. Our assumption is that the next generation of computers will use hardware-supported secure environment to provide applications with a high level of security. It is a realistic assumption as can be seen when considering for instance the Trusted Computing Group [14] which develops specifications to embed on computers a Trusted Platform Module (*i.e.* a tamper proof module similar to the smart cards of our platform). This group gathers all the leaders of the smart cards, the computer software and the computer hardware markets. There is no doubt that the technology of smart cards or at least a smart card like technology will be integrated into future computers.

Figure 1. The Java Card grid.

The rest of this paper is organized as follows. In section 2 we describe the current status of our hardware and software platforms. In section 3 we present the software tools that we have developed, the aim of which is to manage the platform. Then, in the three next sections, we exhibit three different application domains in which we have experimented our platform by implementing one application per domain. The last section is dedicated to additional areas that could take advantage of the security features of our framework.

∗ Java and all Java-based marks are trademarks or registered trademarks of Sun microsystems, Inc. in the United States and other countries. The authors are independent of Sun microsystems, Inc. All other marks are the property of their respective owners.

† LaBRI (Bordeaux, FRANCE), LACO (Limoges, FRANCE) and LMSI (Limoges, FRANCE).

‡ LaBRI (Bordeaux, FRANCE) and University of Sfax, ENIS, TUNISIA

§ LMSI (Limoges, FRANCE).

2. The Java Card Grid Platform

Our actual platform can be viewed as a hardware platform and a software framework.

2.1. Hardware Platform

The hardware platform that we have set up is presented Fig. 2.

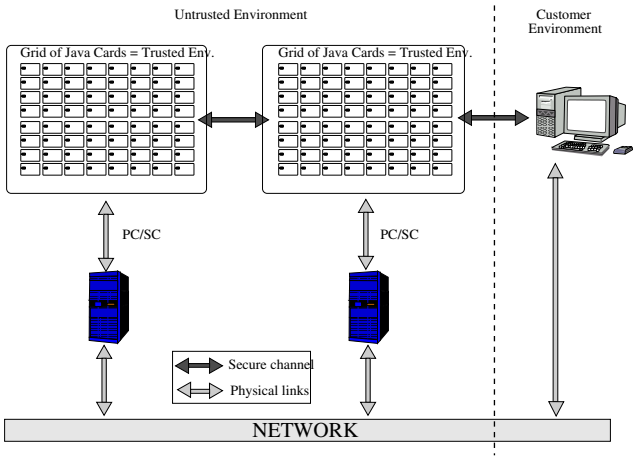

Figure 2. A hardware platform based on Java Card grids.

As this is shown on this figure, we have deployed two grids that are connected together by the network. The whole of the hardware is installed in a wall mount cabinet of 19U. Each grid is composed of:

- a PC which needs 2U;
- two 2U racks from SmartMount where each one has 8 CCID readers from SCM Microsystems, *i.e.* a total of 16 CCID readers;
- three USB 7-port hubs (placed in a empty 2U rack) to connect the readers and the PC and to power the readers;
- Java Cards of different manufacturers plugged in the readers which power them.

We have also connected to one of the PC a LCD monitor and a special rackable keyboard with an integrated touchpad that we use to control the servers. This platform is shown Fig. 1.

2.2. Software Framework

The software framework is constituted of two layers: a low level layer that handles the PC, the readers and thus the smart cards and a high level layer that manages the distributed computing aspects.

At low level, the pilot PCs run Linux and we use `pcsc-lite` [8], the open source implementation of the PC/SC[1] standard, to manage the readers. We have also chosen CCID[2] readers since they are supported by an open source generic CCID driver available for `pcsc-lite` [12]. Fig. 3 illustrates interactions between the different components of the PC/SC stack with a CCID reader and a Java Card applet[3]. Since our high level framework is Java-based,

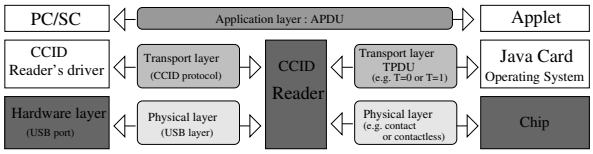

Figure 3. The PC/SC solution with a CCID reader and a Java Card.

we have added JPC/SC [10], a JNI-wrapper for PC/SC, to control the readers and to manage applications embedded on cards(*cf.* Fig. 4).

At user level, the programming API is based on Mandala [6], a general framework that has been developed in our team to support distributed computing in Java. It basically provides a RMI-like abstraction but it also offers features that are useful in our context (*i.e.* to deal with slow computing resources) such as the *active container* [5] concept it is based on, or the *asynchronism* it provides for remote method invocation [16].

The overall software framework deployed on the two hosts is shown Fig. 4.

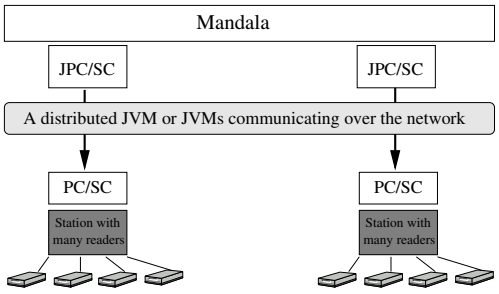

Figure 4. The software framework.

1 PC/SC [11] is a standard which provides a high level API to communicate with smart card readers.

2 CCID is a standard which defines a protocol to handle USB readers.

3 Java Card applications are also called *applets*.

3. Administration Software

Our aim is to design and implement tools to support remote administration of the grid of Java Cards, *i.e.* to monitor the topology of the grid, to deploy new applications on the Java Cards, etc.

3.1. Remote Administration Tool for the Topology

This tool enables a distant administrator to display on its monitor the topology of the grid, *i.e.* to see which readers are free and which ones contain a card. The administrator can also use it to track the evolution of the grid since any modification of the state of the grid, resulting from the introduction or the tearing of a card, is instantly notified to him. As shown Fig. 5, with a simple click he can then get useful additional information about a given card and its associated reader: communication protocol, name of the reader, name of the card, ATR[4] of the card, etc.

Figure 5. Remote administration tool for the topology.

To achieve this goal we had to solve a number of technical challenges, in particular the problem of naming the readers. It was necessary to make a certain mapping between the name attributed to a reader by PC/SC and the effective reader device in order to determine the physical position of this reader in the grid (*cf.* Fig. 6).

The problem is that the way PC/SC assigns the names to the readers is unpredictable. For example, if two identical readers are connected to a computer, the names given by PC/SC are unforeseeable and this does not allow proper identification, *i.e.* it is not possible to work out which name corresponds to which reader. Moreover, the name of the same reader can be different between two sessions of

4 Answer To Reset.

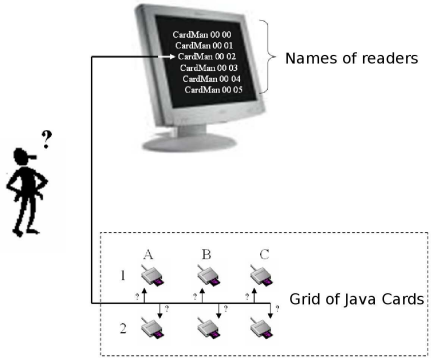

Figure 6. Relation between name reader and the physical position of this reader in the grid.

PC/SC. In practice, to assign a name to a given reader, `pcsc-lite` uses a string representing the model of the reader to which it appends the two bytes identifier of the communication channel of this reader. Unfortunately, this identifier is assigned in a random way and has no relationship with the physical device reader. Our grid of Java Cards is composed of identical readers (*i.e.* the readers are the same model), thus it is not possible to determine the position of a reader by simply using the name as given by PC/SC. However, we need a way to distinguish between the readers. This can be achieved by adding to the name assigned by PC/SC, information which makes a relationship between the name and the physical reader present at a given position in the grid. The idea is to use the serial number of a reader. Indeed, each reader has a serial number that is different by definition from the serial numbers of the other readers. Thus, the knowledge of the serial numbers, which can be obtained for example by using the `libusb` library, makes it possible to distinguish between the readers present in our grid of smart cards. To achieve our goal, we have thus modified `pcsc-lite` so that the name given to a reader takes the serial number of the reader into account and introduces it in the name attributed to the reader.

3.2. Future Developments

The next tools and APIs will be dedicated to the deployment of applets on a set of Java Cards. Since most of the Java Cards are GlobalPlatform[5] [9] compliant, we will only need to develop an implementation of this standard to handle in an easy way the loading and the deletion of applications.

5 GlobalPlatform is a standard which specifies APIs to manage multi-application cards.

4. Secure Distributed Computing Applications: The Mandelbrot Set Demonstration Application

In this section we briefly present the very first application that we developed on the Java Card Grid. This distributed computing application calculates the Mandelbrot set. Even though it is not security demanding, it made it possible to validate the functional operations of the platform. However distributed computing applications which really need high levels of security can be deployed in a similar way that we have used to set up this demonstration application. The details of this demonstration application are available in [3].

4.1. Overview

To compute the Mandelbrot set, our demonstration application is based on the DJFractal project[6] [15] which uses Mandala [16] to distribute the computation over a set of Java Cards. DJFractal is composed of a GUI which displays the results of the computation, a scheduler which decides which computing resource, called in our context *fractal computer*, must compute the next data. We have deployed on each Java Card an applet which delivers results for the given input data. On the host computer there is a proxy per reader which translates incoming calls to fractal computer methods into APDUs[7] intended to the related applet and which forwards results to the scheduler and then to the GUI. Fig. 7 shows the complete architecture of DJFractal over the Java Card grid and the interaction between the different entities. Fig. 8 is a screenshot of what is displayed when the demonstration application is running.

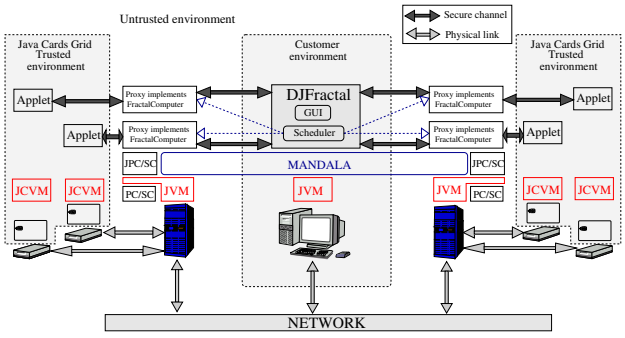

Figure 7. The DJFractal application over the Java Card grid.

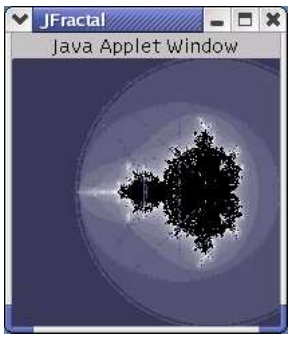

Figure 8. Screenshot of the Mandelbrot Fractal computed over the cards.

4.2. Benchmarks and Scalability

As explained above, this experimentation has been run before building the effective platform. Therefore the hardware that we used is different from the current grid. It was composed of 2 PCs, 9 USB CCID readers, 2 USB hubs with 7 slots and 9 Java Cards. We have run our demonstration several times, each time using 9 cards from a different manufacturer. We have run it on GemXpresso Pro R3 E32PK, GemXpresso Pro R3 E64PK, JCOP31bio, SmartC@fé Expert and a Java Card implementation onto Fujitsu mb94r215b. The results that we get clearly show differences of performance between the cards that were used.

Fig. 9 presents the scalability of our framework with the Java Card implementation onto Fujitsu mb94r215b (FRAM). The speed up is initially very good, and it reaches a threshold when the number of cards is increasing and then begins to decrease. When using 6 cards, there was a small problem not detected – certainly a dead card which could explain the irregularity of the curve. Soon we will achieve more significant tests of scalability on the current platform.

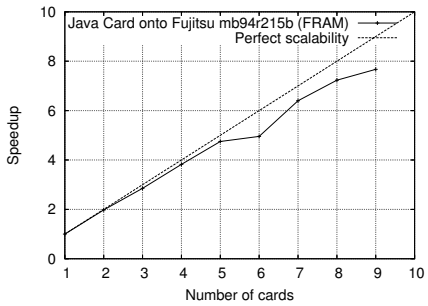

Figure 9. Scalability of the framework.

6 DJFractal is yet another fractal generator.

7 APDU is the elementary message to communicate with smart cards at the application level.

5. Secure Distributed Data-mining Applications: The FBI – Air France Demonstration Application

In this section we present an application to show the secure data-mining capability of our platform. This kind of applications enables to ensure the confidentiality of both the data and the code of a given application, each being the property of two separate entities that do not trust each other. The demonstration application is based on a scenario between the FBI and the Air France airline company.

5.1. Overview

Since September 11, 2001, for security reasons, the FBI (Federal Bureau of Investigation) wants to be able to carry out certain research in the passenger files of airline companies, for example, Air France, for the flights bound for the United States. However Air France does not want to communicate its files to the FBI because of the confidentiality of the data related to its passengers. Indeed, the regulations regarding personal data were recently enforced in the European Community by directives requiring the airline companies to preserve the confidentiality of personal information of the passengers whom they transport. Nevertheless, if Air France does not collaborate, the FBI could cancel its authorization to land on the American territory, putting a term at its commercial activity towards the United States.

A basic solution could be that the FBI provides Air France with its application. Air France could launch the application on the passenger files and get back to the FBI only the list of the potentially suspect passengers. But several problems would arise. Indeed, the FBI does not want to give any information about its algorithm, *e.g.* the discriminating criteria it uses. However, an analysis of retro-design of the program by Air France would make it possible to discover these criteria. In addition, the FBI does not want the execution of its code could be spied or modified. Therefore, the code should be executed either at the FBI or on a trusted platform. Lastly, if the airline company agreed to run the program of the FBI, it could not be sure that the code did not contain a Trojan horse so as to have access to all the data. Thus, if the company agreed to run the code, that could only be on a trusted platform where the code could not make any operation or access any piece of data without its agreement. Thus, in this contradictory context, the solution that we propose makes it possible to respect these constraints (*i.e.* to restrict the access to the passenger files, as the laws impose, still sharing them with the FBI).

5.2. Implementation Details

To deal with the constraints presented above, in this application a passenger is represented on a card by the `Passenger` class containing all his information (*i.e.* name, first name, nationality, etc.) and a given key that makes it possible for Air France to identify him. On each card, an applet called `PassengerManager` makes it possible for Air France to manage the list of its passengers and thus its passenger files are distributed over a set of Java Cards on the grid. This `PassengerManager` applet also provides an interface (*i.e.* an API) that enables the FBI to search for a given piece of information among the pieces of data specified public by Air France. To achieve this goal, the `PassengerManager` class publishes a *Shareable*[8] interface that gives a restricted access to the passenger data and does not allow to identify a passenger by name but only by the key. Thus, Air France can share the data about its passengers with the FBI using a mechanism protected and controlled by the JCRE (Java Card Runtime Environment). Once all the passengers have been entered in the database, the FBI which is represented on the Java Cards by the `CheckPassengers` applet can try to determine which are the passengers who seem suspect. For that, the FBI specifies the criteria used to decide if a given passenger is suspect or not. These criteria are transmitted to the `CheckPassengers` applets which contain the secret discriminating algorithm and which analyze the passenger files. When a given criteria is met, the FBI application comes back with a key that represents the suspicious passenger. It can then get back to Air France and ask for more information about this given passenger, information that Air France is then free to provide or not.

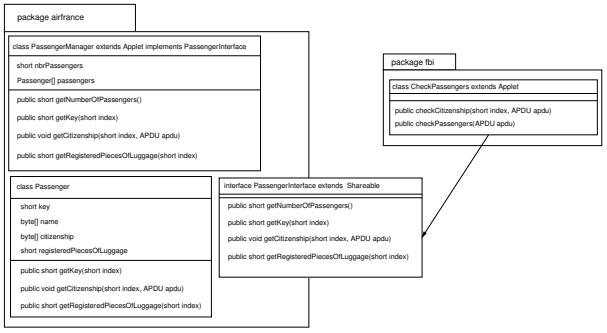

Figure 10. Sharing data between Air France and FBI.

8 *Shareable* interfaces are special Java Card interfaces that enable to bypass the Java Card firewall mechanisms in a controlled way.

6. Secure Distributed Storage Applications: The PKI Demonstration Application

In this section we present an application to show the secure storage capability of our platform. This kind of applications can take advantage of a secure distributed storage to ensure the confidentiality and the integrity of data. Our application consists to set up a secure PKI thanks to the possibility to store securely the private keys.

6.1. Overview

Today the security of the communications in a data-processing network is often ensured by systems of infrastructures using cryptographic key pair, called PKI (Public Key Infrastructure). These key pairs are generally generated and distributed by trusted entities called certification authorities. Among the different problems in this field the organization of these authorities, the generation, the distribution and the storage of the keys through all the elements of the network are fundamental.

In order to mitigate those problems, we have set up an application which takes advantage of the tamper-resistant aspect of the Java Cards and of the high level of security of the exchanges within the grid. Contrary to the traditional PKI systems where keys are stored on a simple hard disk, in our application key pairs and certificates of authorities are generated and recorded on cards: our solution is thus more secure since private keys never "leave out" cards.

Thus, by allowing secure generation and distribution of the keys of the different authorities, the grid facilitates the installation of structure of which it is not possible to predict the final organization when first deploying and thus in this manner the decentralization of the certification authority by entrusting the responsibilities to sub-groups authorities.

6.2. Example: a PKI for the CNRS

To illustrate our application, we took the example of a group such the CNRS (*Centre National de Recherche Scientifique*), the french national center of scientific research, which gathers several sub-groups, *i.e.* various scientific research laboratories (*e.g.* LaBRI, LIP6, XLIM). Since its organization evolves (laboratories disappear and new laboratories are created) the organization needs to be flexible. Thus to communicate in a secure manner within the sub-group (*i.e.* the laboratory) and with other sub-groups thanks to the hierarchical chaining of the certificate, each laboratory needs to set up its own certification authority. The hierarchical organization set up a tree (*cf.* Fig. 11), where each trusted entity obtains a certificate (for its generated key pair – where only the public key is exposed) issued by the authority of the higher group (except for the root).

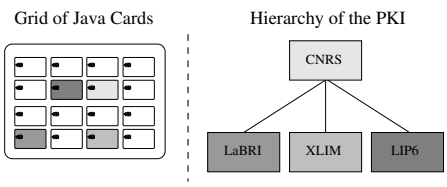

Figure 11. PKI mapping on the grid.

When members are integrated in a sub-group, for instance the LaBRI, they must get themselves from the certification authority of the LaBRI a certified key pair allowing them to communicate with the other members of all the laboratories in a secure way. In the example deployed, this distribution does not require a very significant security and each member securely gets his data through a web site able to deliver a certified key pair to those authorized and according to their group membership.

On the other hand, when a new sub-group is integrated in the organization the process (*i.e.* the generation and distribution of a key pair and a certificate) is perfectly protected in order to avoid to corrupt the whole of the communications of the group. First, an administrator of the LaBRI makes its request to the administrator of the CNRS which inserts in the grid a Java Card dedicated to the certification authority of the LaBRI. The application then achieves the following steps (*cf.* Fig. 12): an applet is installed on the card of the LaBRI to generate the key pair (1); the public key generated is sent to the card of the CNRS in order to be certified (2) (the exchange being done inside the grid, the security is assured); once the certificate is created, it is sent to the card of the LaBRI (3).

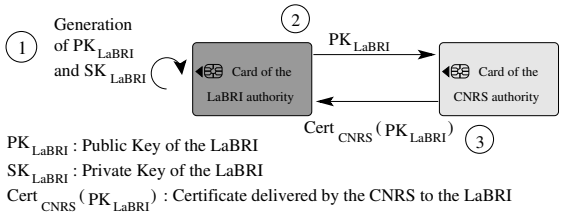

Figure 12. Certification process of sub-authority in the grid.

The access to the certification step can be protected by a PIN code so that an unauthorized person cannot obtain a certification authority card. The application also deals to maintain up to date an image of the tree with the certification authorities as well as a list of their public keys. Those last information are accessible on the web site by the authorized members.

7. Actual and Future Works

In the future we plan to use the Java Card grid to simulate algorithms dedicated to mobile networks. Indeed, an appearance of a node in these kinds of network is similar to the card insertion in a reader and the disappearance of a node to the withdrawal of card of the reader. Thus, it may be possible to simulate algorithms at lower cost (*i.e.* without having to buy many devices – *e.g.* PDA, mobile phones) and on a great number of nodes. The main advantage of this proposition is the possibility to handle all the nodes easily since all are on the same place in the grid, on the front of the experimenter.

We also plan to deploy a mobile Java Card Grid to secure the MANETs. In this project, led in a cooperation between the LaBRI and the LMSI, we will put on each node a smart card and a reader in order to benefit from a secure process environment. All together the smart cards constitute a secure distributed environment (*i.e.* a secure network among these smart cards and a secure distributed operating system) in the space and in the time that could provide a high level security to the MANETs. Actually the LaBRI is focused on the pure MANETs (*i.e.* without routing protocol between nodes) while the LMSI works on MANETs dedicated to do collaborative works [1] with routing protocols. In a similar way, we plan to deploy such a grid in Delay Tolerant Network to explore security benefits of smart cards in such environments.

8. Conclusion

Because of the development of the network it is now possible to share resources between possibly unknown persons or organizations. Nevertheless, efficient protection mechanisms have to be provided so that this possibility is effectively used.

It is now widely admitted that to support a good level of security, dedicated pieces of hardware are required. In this paper we have presented our solution, a Java Card grid that can be used in many different domains to provide a high level of security. Today it is an original solution but in the future smart card-like devices will be available in every computer. The work presented in this paper enables to imagine what features can be offered, what kinds of applications can take advantage of such devices and how to use them to help in designing efficient and secure applications.

Thanks

Our project is supported by:

- Gemplus and IBM BlueZ Secure Systems (for the cards);
- SCM Microsystems and SmartMount (for the readers);
- Sun microsystems (for the overall platform).

We also thank: Fujitsu, Giesecke&Devrient, Oberthur Card Systems and Sharp for the Java Card samples; David Corcoran and Ludovic Rousseau for their work on `pcsc-lite` and the CCID generic driver.

References

[1] P.-F. Bonnefoi, P. Poulingeas, and D. Sauveron. MADNESS: A Framework Proposal for Securing Work in Ad Hoc Networks. In *Proceedings of CCCT'05*, Austin, Texas, USA, July 24-27 2005.

[2] S. Chaumette, P. Grange, D. Sauveron, and P. Vignéras. Computing with Java Cards. In *Proceedings of CCCT'03 and 9th ISAS'03*, Orlando, FL, USA, July 31, August 1-2 2003.

[3] S. Chaumette, P. Grange, D. Sauveron, and P. Vignéras. Secure distributed computing on a Java Card grid. In *Proceedings of 7th International Workshop on Java for Parallel and Distributed Computing*, Denver, CO, USA, April 4-8 2005.

[4] S. Chaumette and D. Sauveron. The Smart Cards Grid Project. `http://www.labri.fr/Perso/~chaumett/recherche/cartesapuce/smartcardsgri%d/documents/poster.pdf`. Poster presented at Cartes 2003.

[5] S. Chaumette and P. Vignéras. Active containers: an alternative approach to mobile agents systems. Proceedings of the Second International Symposium on Object Oriented Parallel Environments, ISCOPE 98, Santa Fe, NM, USA. Poster.

[6] S. Chaumette and P. Vignéras. A framework for seamlessly making object oriented applications distributed. In *Parallel Computing 2003*, Dresden, Germany, September 2-5 2003.

[7] Z. Chen. *Java Card™ Technology for Smart Cards: Architecture and Programmer's Guide*. The Java™ Series. Addison-Wesley, 2000.

[8] D. Corcoran, L. Rousseau, and D. Sauveron. pcsc-lite home page. `http://alioth.debian.org/projects/pcsclite/`.

[9] GlobalPlatform. GlobalPlatform. `http://www.globalplatform.org/`.

[10] IBM BlueZ Secure Systems. Java Wrappers for PC/SC. `http://www.musclecard.com/middleware/files/jpcsc-0.8.0-src.zip`.

[11] PC/SC Workgroup. PC/SC Workgroup Home. `http://www.pcscworkgroup.com/`.

[12] L. Rousseau. CCID free software driver. `http://pcsclite.alioth.debian.org/ccid.html`.

[13] Sun microsystems. *Java Card™ 2.2.1 Specifications*. Sun microsystems, 2003.

[14] Trusted Computing Group. Trusted Computing Group Home. `https://www.trustedcomputinggroup.org/home`.

[15] P. Vignéras and P. Grange. The DJFractal project. `http://djfractal.sf.net/`.

[16] P. Vignéras and P. Grange. The Mandala website. `http://mandala.sourceforge.net/`.