Smart Card HOWTO

Tolga KILIÇLI

tolga@deepnight.org

Revision History

Revision 1.0.4	2001−09−19	Revised by: tk
This is the first release of Smart Card HOWTO.		

This document gives information on the smart card technology and its applications in Linux environment. Smart cards are mainly used in situations where security is an issue. But this is not the only situation where smart cards are used. They have many properties which a service provider wants to use, such as "one card for many applications".

Table of Contents

1. Introduction

For various reasons this brand new release is codenamed the *OberoN* release.

New code names will appear as per industry standard guidelines to emphasize the state−of−the−art−ness of this document.

This document was written when a friend (JaSoN) asked me if I could write a document on smart cards and applications. And all the pages here was once a bunch of papers. Thanks JaSoN...

1.1. Copyright Information

As the author of this document, I would like to list the derivative works and publications in this document.

1.2. Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author do not take any responsibility for that.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3. New Versions

This is the initial release.

The latest version number of this document can be found on my website.

1.4. Feedback

Please send your additions, comments and criticisms to the following email address : `<tolga@deepnight.org>`.

1.5. Translations

Not everyone speaks English, pointers to translations are nice. Also your translators tend to give very important inputs. If you want to translate this document in your own language please let me know so I could list it down here.

2. What is a Smart Card?

Simple plastic card, just at the size of a credit card, with a microprocessor and memory embedded inside is a smart card. Beside its tiny little structure it has many uses and wide variety of applications ranging from phone cards to digital identification of the individuals.

These application could be; identity of the customer, library card, e−wallet, keys to various doors, etc... And only one card can be issued to an end−entity for all these applications. Smart cards hold these data within different files, and , as you will read, these data is only visible to its program depending on the operating system of the card. These data files are arranged in a file system much like a Linux directory structure.

```
                        MF (Master File)
                              |
                 -------------+--------------
                 |            |             |
EF (Elementary File)    EF   DF (Dedicated File)
                                            |
                                   ---------+--------
                                   |        |       |
                                   EF       DF      EF
                                            |
                                            EF
```

MF (Master File), can be seen as the root directory where the headers of elementary files and dedicated files are contained. Dedicated files are like the ordinary directories and elementary files are just data files. The PIN is also stored in an EF but only the card has access permission to this file. The attributes of the files on UNIX environments are changed to access conditions. Many cards have access condition lists which must be fulfilled before accessing the data.

With the file system, access conditions, a microcomputer, RAM, ROM, EEPROM a smart card is just a computer running its own operating system inside your wallet.

3. Classification of Smart Cards

Due to the communication with the reader and functionality of smart cards, they are classified differently.

3.1. Contact vs Contactless

As smart cards have embedded microprocessors, they need energy to function and some mechanism to communicate, receiving and sending the data. Some smart cards have golden plates, contact pads, at one corner of the card. This type of smart cards are called *Contact* Smart Cards. The plates are used to supply the necessary energy and to communicate via direct electrical contact with the reader. When you insert the card into the reader, the contacts in the reader sit on the plates. According to ISO7816 standards the PIN connections are below:

```
     ,----,          ,----,
     | C1 |          | C5 |        C1 : Vcc = 5V      C5 : Gnd
     '----'          '----'        C2 : Reset         C6 : Vpp
     ,----,          ,----,        C3 : Clock         C7 : I/O
     | C2 |          | C6 |        C4 : RFU           C8 : RFU
     '----'          '----'
     ,----,          ,----,
     | C3 |          | C7 |
     '----'          '----'
     ,----,          ,----,
     | C4 |          | C8 |
     '----'          '----'
```

- I/O : Input or Output for serial data to the integrated circuit inside the card.
- Vpp : Programing voltage input (optional use by the card).
- Gnd : Ground (reference voltage).
- CLK : Clocking or timing signal (optional use by the card).
- RST : Either used itself (reset signal supplied from the interface device) or in combination with an internal reset control circuit (optional use by the card). If internal reset is implemented, the voltage supply on Vcc is mandatory.
- Vcc : Power supply input (optional use by the card).

The readers for contact smart cards are generally a separate device plugged into serial or USB port. There are keyboards, PCs or PDAs which have built−in readers like GSM cell phones. They also have embedded readers for GSM style mini smart cards.

Some smart cards do not have a contact pad on their surface.The connection between the reader and the card is done via radio frequency (RF). But they have small wire loop embedded inside the card. This wire loop is used as an inductor to supply the energy to the card and communicate with the reader. When you insert the card into the readers RF field, an induced current is created in the wire loop and used as an energy source. With the modulation of the RF field, the current in the inductor, the communication takes place.

The readers of smart cards usually connected to the computer via USB or serial port. As the contactless cards are not needed to be inserted into the reader, usually they are only composed of a serial interface for the computer and an antenna to connect to the card. The readers for contactless smart cards may or may not have a slot. The reason is some smart cards can be read upto 1.5 meters away from the reader but some needs to be positioned a few millimeters from the reader to be read accurately.

There is one another type of smart card, combo card. A combo card has a contact pad for the transaction of large data, like PKI credentials, and a wire loop for mutual authentication. Contact smart cards are mainly used in electronic security whereas contactless cards are used in transportation and/or door locks.

3.2. Memory vs Microprocessor

The most common and least expensive smart cards are memory cards. This type of smart cards, contains EEPROM(Electrically Erasable Programmable Read−Only Memory), non−volatile memory. Because it is non−volatile when you remove the card from the reader, power is cut off, card stores the data. You can think of EEPROM, inside, just like a normal data storage device which has a file system and managed via a microcontroller (mostly 8 bit). This microcontroller is responsible for accessing the files and accepting the communication. The data can be locked with a PIN (Personal Identification Number), your password. PIN's are normally 3 to 8 digit numbers those are written to a special file on the card. Because this type is not capable of cryptography, memory cards are used in storing telephone credits, transportation tickets or electronic cash.

Microprocessor cards, are more like the computers we use on our desktops. They have RAM, ROM and EEPROM with a 8 or 16 bit microprocessor. In ROM there is an operating system to manage the file system in EEPROM and run desired functions in RAM.

```
               ----------------
              |   8 or 16 bit   |
  Reader <===| microprocessor |-----+
               ----------------       |
                                      |
                                      |---> RAM
               NON-CRYPTOGRAPHIC      |
                     CARD             |---> ROM
                                      |
                                      +---> EEPROM
```

As seen in the diagram above all communication is done over the microprocessor, There is no direct connection between the memory and the contacts. The operating system is responsible for the security of the data in memory because the access conditions are controlled by the OS.

```
               ----------------             --------
              |   8 or 16 bit   |           | Crypto |
  Reader <===| microprocessor |-----------| Module |
               ----------------       |      --------
                                      |
                                      |---> RAM
                 CRYPTOGRAPHIC        |
                     CARD             |---> ROM
                                      |
                                      +---> EEPROM
```

With the addition of a crypto module our smart card can now handle complex mathematical computations regarding to PKI. Because the internal clock rate of microcontrollers are 3 to 5 MHz, there is a need to add a component, accelerator for the cryptographic functions. The crypto−cards are more expensive than non−crypto smart cards and so do microprocessor card than memory cards.

Depending on your application you should choose right card.

4. Operating Systems

New trend in smart card operating systems is JavaCard Operating System. JavaCard OS was developed by Sun Microsystems and than promoted to JavaCard Forum. Java Card OS is popular because it gives independence to the programmers over architecture. And Java OS based applications could be used on any vendor of smart card that support JavaCard OS.

Most of the smart cards today use their own OS for underlying communication and functions. But to give true support for the applications smart cards operating systems go beyond the simple functions supplied by ISO7816 standards. As a result porting your application, developed on one vendor, to another vendor of smart card becomes very hard work.Another advantage of JavaCard OS is, it allows the concept of post−issuance application loading. This allows you to upgrade the applications on smart card after delivering the card to the end−user. The importance is, when someone needs a smart card he/she is in need of a specific application to run. But later the demand can change and more applications could be necessary.

Another operating system for smart cards is MULTOS (Multi−application Operating System). As the name suggests MULTOS also supports multi−applications. But MULTOS was specifically designed for high−security needs. And in many countries MULTOS has achieved "ITSec E6 High" in many countries.

And also Microsoft is on the smart card highway with Smart Card for Windows.

In a point of view the above Operating Systems are Card−Side API's to develop cardlets or small programs that run on the card. Also there is Reader−Side API's like OpenCard Framework and GlobalPlatform.

5. Programming

5.1. CT−API

This API depends on the card terminal used, but supplies generic functions that allow communication with memory cards and processor cards. This API is a low level interface to the reader. But still used because it complies with the ISO7816 standards and have a simple programming logic resembling assembly. You just send code byes along with the data packets and receive the response.

5.2. PC/SC

PC/SC Workgroup is responsible for the development of the PC/SC Specifications. Under Windows, MacOS and Linux corresponding APIs could be found. Under Linux, pcsc−lite suit could be downloaded from http://www.linuxnet.com.

5.3. OpenCard

OpenCard Framework, OCF, is an object−oriented framework for smart card communications. OCF uses Java's inter−operability between environments to deploy architecture and APIs for application developers and service providers.

5.4. GlobalPlatform

GlobalPlatform was formed in 1999 by organizations those were interested in issuing multiple application smart cards. The major goal of GlobalPlatform is to define the specifications and infrastructure for multi−application smart cards.

5.5. To Sum Up

As you could understand from above, the standardization period of smart cards is not finished. The demand on smart cards is growing on the basis of end−user and developer. In my opinion, if you are a developer or in a decision making position, you should carefully analyse all the standards as well as the companies manufacturing smart cards. As a developers point of view, in the near future I think, Java will evaluate itself as the standard because of portability and cross−platform uses in spite of its slowness and fast evolution.

6. Applications on Linux

In this section there will be applications that uses smart cards for some reason on Linux environment. If you are a developer of a software and your development environment is Linux please let me know. I will add you in the list.

6.1. scas

SCAS is a simple program that checks the code inside the card with the code inside the computer. As an example of showing a way of authentication with memory cards scas is very good.

6.2. smartcard

smartcard is a general smart card utility in Linux which uses CT−API. With smartcard utility you can read/write data from/into smart cards. As long as your reader can be accessed via CT−API, smartcard can be used to control the reader. Currently smartcard could only be used with memory cards using I2C or 3W protocols. There is also a GTK+/Gnome graphical front end which support all functions of smartcard utility.

6.3. ssh−smart

ssh−smart is a basic proof−of−concept of ssh identity on smart card, as the author says. ssh−smart uses smartcard utility to communicate with the smart card. Basically, ssh−smart−add tool (perl script) call ssh−keygen to generate RSA public and private keys. Than puts the private key on the memory card. Later the ssh−smart−addagent tool can be used to extract the private key from the card to use with ssh−agent.

6.4. smarttools−rsa

This is another PAM Module for Unix systems but supports RSA authentication through your private key on the smart card. You must have a Schlumberger Cyberflex Access card or Schlumberger Cryptoflex for Windows Card and a working reader to use this tool.

6.5. smartsign

This utility is some−complete PKI integration with the smart cards. To use you must establish a working OpenCA and have Schlumberger's "Cyberflex Access 16K" smart cards. During the certification process of OpenCA, private key and public certificate can be stored on the smart card and private key, later, could be used with Netscape to sign outgoing mails and news. Also smartsign supports authentication of local users via a PAM Module through a public key authentication. Smartsign comes with gpkcs11, a PKCS#11 implementation, smastsh, a command line shell that allows browsing smart card contents, sign_sc/verify_sc to sign and verify any file with smart card.

6.6. CITI Projects

At CITI, Center for Information Technology Integration of Michigan University, there are some new projects. For example, Webcard is a web server running on a Schlumberger Cyberflex Access Java Card. Features a stripped TCP/IP stack that supports HTTP only. The system is designed to have a router which frames IP packets in ISO7816 and a Java Virtual Machine in the card. Detailed technical report can be found at http://www.citi.umich.edu/projects/smartcard/webcard/citi−tr−99−3.html

7. The Relation of Smart Cards with PKI

As we already know smart cards are secure place to hold sensitive data, such as money and identity. And if the identity is the subject we should talk about PKI, Public Key Infrastructure, and smart cards.

Think that, you are working in a company with many branch offices and many facilities. In such large companies often employers have access permissions to different physical places. Also you access the servers inside the company for various purposes like sending mail, uploading the web pages and accessing the databases of the company. Just think, one password for each server and one key for each door and some money in your wallet to buy food or drink from the local restaurant.

Actually you could just use a smart card. If you use a microprocessor card and a the cards operating software or Java cardlets permit, you could use only one card for all these. For this scenario to work, the company must establish a local CA, Certificate Authority. Below there is a diagram showing the structure of a PKI simply, as described in RFC 2459.

```
+---+
| C |                       +------------+
| e | <-------------------->| End entity |
| r |       Operational     +------------+
| t |       transactions          ^
|   |      and management         |  Management
| / |       transactions          |  transactions
|   |                             |                PKI users
| C |                             v
| R |       -------------------+--+-----------+----------------
| L |                          ^              ^
|   |                          |              |  PKI management
|   |                          v              |      entities
| R |                       +------+          |
| e | <---------------------| RA   | <---+    |
| p |  Publish certificate  +------+     |    |
| o |                                    |    |
| s |                                    |    |
| I |                                    v    v
| t |                                +------------+
| o | <------------------------------|     CA     |
| r |   Publish certificate          +------------+
| y |   Publish CRL                         ^
|   |                                       |
+---+                        Management     |
                             transactions   |
                                            v
                                        +------+
                                        |  CA  |
                                        +------+
```

- end entity: user of PKI certificates and/or end user system that is the subject of a certificate;
- RA: registration authority, i.e., an optional system to which a CA delegates certain management functions; (in some implementations, where you register your self to the system)
- CA: certification authority; (Your public key, can be issue when you register yourself or can be self–issued, is signed and your certificate is issued to you at CA)
- repository: a system or collection of distributed systems that store certificates and CRLs, Certificate Revocation Lists, and serves as a means of distributing these certificates and CRLs to end entities.

In fact, this is just a simplified view of the entities PKI. The employer or the end entity just applies to the CA or RA to get a certificate A certificate is just a public key digitally signed with the issuer's, CA, private key. By signed with the CA's private key, all which trust the CA, can also trust the end entity. Your digital ID is ready. Just write your digital ID and private key to your smart card. Or a better way, new smart cards are deployed with embedded functions that generate public and private keys inside the card which means your private key is not exported to anywhere.

New deployed cards are capable of PKI functions which you do not need to export the private key to the application you use. For example when you want to send a signed mail, your mail applications first generates a hash of the document you just wrote and starts the communication with the card. Your application sends the hash value to the card which is than signed with your private key inside the card. By this way your private key is never exported to the public, your computer.

Also, while accessing your remote shell account you could use ssh, secure shell, client. In man page of OpenSSH, an authentication method for ssh protocol 2 is described. Main purpose of the method is true identification of the person trying to access the account and secure connection between the host, if the user is accepted. Theoretically, only you can know your private key. Although your private key is only readable by yourself, this could be a security risk. But if your private key is inside a smart card, this is an increased security. Of course, a smart card can get lost. But at this point another security subject is on the line, your PIN. Generally speaking, smart card's security comes from two things, one you know and one you own.

SSH is not the only application that smart cards can be used. Other applications like, money transactions on the net, identification of yourself to the website you connect can be done with smart cards. The system is more or less the same. Your identification is checked via your private key and secure session is started with your keys. Than application specific part comes which is designed and deployed by the service provider of the application. Some money transactions are just done inside the smart card but some applications just ask the card for your banking account number. There could be more methods.

Electronic locks that can communicate with a smart card can be found on the market. PKI can support, in addition to the mutual authentication between the card and the reader, access accounting in the building. Just mutual authentication can be used or the lock ask to a local server that keeps the user data and checks if the user is permitted to go behind the door. And whether the permission is granted or not the server keeps the tracks of the access trials.

With integration of smart cards into PKI world, many more applications could be built. These application are mostly security specific or to ease the life of the customers.

---

8. Further Information

In this section there are places to visit for more in−depth information.

8.1. News groups

Some news groups are:

- alt.technology.smartcards
- sci.crypt.research
- sci.crypt.random−numbers

8.2. Mailing Lists

From the MUSCLE Project, `<sclinux@linuxnet.com>`, Smart Card Developers mailing list. The subject of the list is smart card development under Unix and Mac OS. Just send `<majordomo@linuxnet.com>` with subscribe linux in the body of your mail. Also you can reach the archives at The Mail Archive. See linuxnet.com mailing list page for more information.

8.3. Web Sites

There are a huge number of informative web sites available. They could change and get outdated.

A good starting point is Movement for the Use of Smart Cards in a Linux Environment home page, an information central for documentation, project pages and much more.

Also, USENIX Workshop on Smartcard Technology can take your interest.

Please let me know if you have any other leads that can be of interest.

9. TODO

As all HOWTOs should be, this document will retain in "Under Development" phase as long as smart card technology is not obsolete.

- The part about the physical characteristics of smart cards should be re−organized.
- In the "Programming" section there must be more information about the standards of programming smart cards.
- A new section of examples must be added.
- Scenario section (e.g. Building a Corporate PKI) should be added with in−depth information. (I will add some time in a few weeks :))
- There could be a section about the tamper resistance of smart cards. How tamper resistance is supplied and how secure is smart cards against new high−tech gamers. (I have found some references and information but they must be organized before adding.)

Wow, it seems like I have many things to add :))