

# On Securing Unix Systems with Smart Cards

Dirk große Osterhues, Alexander Wiesmaier and Roberto Araújo

Department of Cryptography and Computeralgebra  
Technische Universität Darmstadt  
Hochschulstr. 10; D-64283 Darmstadt, Germany

**Abstract.** The subject of this paper is to give an overview of the state of the art in securing Unix-based workstations by means of PKIs and smart cards. Firstly, we discuss the basic principles of the operating mode and the existing standards. Following this, we explore technical demands of smart cards and the respective software. We focus on a survey of available card readers, smart cards and the current state of open source-projects regarding the use of smart cards. An evaluation of the present state of smart card-development marks the end of the work.

**Keywords.** Smart Cards, Card Readers, Unix

## 1 Introduction

### 1.1 Why smart cards?

Rising threats to IT security create the need for more sophisticated measures to achieve the required security goals. The three most important IT security goals are privacy, confidentiality and non-repudiability (see Eckert [1]).

Because the classic password protection has turned out to be insufficient regarding attacks like e. g. brute force or social engineering, different approaches have been taken with different results. *Biometry*, which is easy to handle for the user, has not yet been developed enough to be both cheap and reliable. *Magnetic cards* are cheap and easy to handle, but they are also easy to duplicate and thus vulnerable to forgery. Finally, smart cards (chip cards with their own cryptographic processor) seem to be able to solve this problem. They represent the most suitable tradeoff between cost, security and comfortable usage. Note that comfort is also a security relevant issue. Although average PC-users think of themselves as being security aware, only a fraction undertake real efforts to enhance their own security situation (as mentioned by the German BSI in [2]). Due to convenience, most users prefer a rather insecure environment, which is easy to handle, instead of a secure one, which is often perceived as constrictive. Thus, nowadays comfortability got very important for supporting high security.

Smart cards are comparatively cheap, relatively easy to handle, reliable and secure. Since there are known theoretical attacks as SPA (Simple Power Analysis) and DPA (Differential Power Analysis) [3], a lot of efforts have been taken into developing adequate countermeasures [4]. So currently, smart cards can be referred to as the most secure storage for cryptographic keys.

One of the reasons for the high security level of smart cards is the two-factor protection. This means that the user has to *own* the card and to *know* the PIN. The strong disadvantages are fairly high costs for setting up and maintaining a hierarchical PKI (Public Key Infrastructure) based on X.509-certificates, which doesn't make smart cards attractive for the SOHO-user (Small Office/Home Office) right now. Although it is possible to use self-signed certificates, reliable interaction with other users and systems can only be assured with hierarchical PKIs. New approaches have been taken especially in the last year by introducing a smart card that supports gpg-keys [5] and thus enabling the usage of a Web of Trust.

## 1.2 Motivation and intention

Smart card-applications still lack a general standard. Several approaches have been taken in implementing suitable projects for different operating systems.

We give an overview of the possibilities in securing Unix-based systems with smart cards nowadays and in the future. By using the term "Unix" we describe all posix-compliant systems, except cygwin [6] for Windows<sup>®</sup>. Because Unix-systems can be found mostly in heterogeneous environments we further check how the tested projects work together with common Windows<sup>®</sup> applications.

## 1.3 Requested security features

This paper deals with the view on available and approved hardware and furthermore with the question which security features are met with existing smart card-related software-projects. These targets are mainly:

**PAM-connection** For workstation-related logon-authentication, nowadays PAM (Pluggable Authentication Modules) is the first choice. PAM covers nearly every authentication task on a local system and applications can make use of PAM through its modules.

Logons to desktop environments as well as `remote login-` or `sudo`(superuser do)-operations can be secured by PAM. Our demand is a PAM-module to secure these operations by means of the user's smart card.

**E-mail security** E-mail security operations like signing and decrypting have to be possible with the user's smart card. The respective email clients usually demand a PKCS#11-module to access the smart card.

**Web-authentication** Web-authentication by means of smart cards is a very important case. Users often need strong security, e.g. for online-banking and many of these applications use public key-authentication. But the user's private key on a hard disk represents a huge security risk. Such keys are far more secure on smart cards.

**Remote-shell security** Because remote-shell-logins are usually encrypted anyway, the main focus of improving the security of these connections is on authenticating the user. Public key technologies are already widely used here. A secure storage for the respective private keys is mandatory.

**Windows<sup>®</sup>-connectivity** As some users work with more than one operating system, it is important that smart cards can be used on different operating systems simultaneously. Thus, we do not only take a look at the Unix-based systems, but also at the Windows<sup>®</sup>-platform.

**VPN-authentication** The last requested feature is VPN-authentication. Networks with strong confidentiality-requirements have to make use of smart cards for access control to ensure the highest security level.

The strength of the applied cryptography depends on the utilized algorithms and the chosen parameters, as e. g. the key length. Thus, both of them are to be considered in our survey. If a smart card or project covers even more security-related issues we will describe it in detail.

#### 1.4 Standards in the field

**PKCS#15** This standard was developed by RSA Labs [7]. PKCS#15 [8] gives a standardized description for names and positions of relevant files (certificates, cards and other files) on the card. This makes it easier to establish a platform-independent access to the data on the smart card.

**ISO 7816 1-4** The ISO 7816 1–3 standards from the International Standardization Organization (ISO) describes the different hardware attributes and form factors a smart card should have. They provide distinct information on how the shape of the card has to look like, which size it has to have and where the crypto-chip has to be located.

ISO 7816 4 describes the contents of the messages, commands and responses transmitted by the interface device to the card and conversely. Cards fulfilling this standard are sometimes also referred to as filesystem-based smart cards.

**JavaCards** JavaCards represent a special sort of smart cards, which integrate the Java Runtime Environment (JRE) on smart cards. The fact that the original specification of JavaCards lacks a description for the file management makes developers say that those cards have no real smart card-operating systems [9]. Nevertheless it is possible to emulate an ISO-filesystem on these cards through an abstraction layer. Thereby JavaCards have the advantage of giving the developer freedom regarding the position and management of data stored on the card.

## 2 Overview of tested hardware

This section firstly gives a survey on available card readers which have been tested to work with Unix-based systems. Then an overview of approved smart cards follows. This list is not complete, but provides the reader with an outline of applicable readers and smart cards.

There are four major classes of card readers:

- Class 1** -readers solely serve as connectors between the computer and the card. Each command is directly forwarded to the card. They may have a display and a keypad which then can be controlled by the computer.
- Class 2** -readers have a keypad which—at least temporary—cannot be accessed by the computer. This enables a secure entry of the PIN.
- Class 3** -readers are Class 2-readers with a display which also cannot be accessed by the computer. This enables the secure display of card and reader information. Additionally, the reader checks the commands and data transmitted to the card for plausibility.
- Class 4** -readers are Class 3-readers with a security module which signs the transactions done with this reader. This enables the identification of the reader type and serial number by the application. Thus, the application can check whether the given reader is allowed to be used for the given kind of transactions. No Class 4 readers were tested in this paper.

We also mention the possible integration of RFID-chips into the card, although this is not a main purpose of this paper. RFID chips provide useful services like cashless payment. But they also raise a lot of privacy questions. Schneier provides a link collection on topics around RFID-chips in [10].

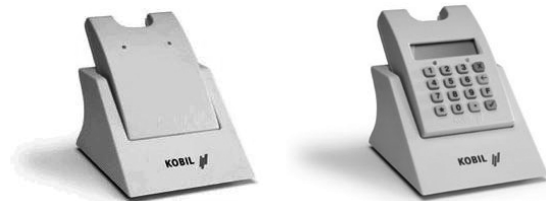
## 2.1 Overview of card readers

**Kobil CT-B1-S** Kobil’s CT-B1-s (see Fig. 1(a)) is a Class 1-reader with serial-port-connection. To operate it under Unix-systems, closed source-drivers provided by Kobil [11] for PC/SC or CT-API have to be used. Its name originates from the B1-standard developed by German Telecom [12], which represents the vertices of the Class 1-definition. Because Kobil’s focus is more on Windows®-systems than on supporting the Unix-environments, it is not well-documented for Unix-usage and thus was not tested in depth.

**Kobil KAAN Professional** Also for the Kobil KAAN Professional (see Fig. 1(b)) only closed source-drivers are available [11]. This reader is available in serial port- as well as in USB-design and offers a two-line display and a PIN-pad. Thus it is a Class 3 reader. It is evaluated to be ITSEC E2 high<sup>1</sup>.

**Towitoko Chipdrive Micro, Chipdrive extern** The card readers “Chipdrive Micro” (see Fig. 2(a)) as well as the “Chipdrive extern” (see Fig. 2(b)) from Towitoko are Class 1 readers without display or PIN-pad. They are available as serial port- and as USB-model. The manufacturer does not provide Unix drivers. Open source-drivers are available from Carlos Prados [13], with which it is possible to use them with CT-API [14] as well as with PC/SC [15], which will be discussed in Section 3.1.

<sup>1</sup> this is equivalent to CC EAL 3+



(a) CT-B1-S

(b) KAN Professional

**Fig. 1.** Card-readers from Kobil



(a) Chipdrive Micro

(b) Chipdrive Extern

**Fig. 2.** Readers from Towitoko

**Omnikey CardMan 3621 an 3821** The models CardMan 3621 (see Fig. 3(a)) and 3821 (see Fig. 3(b)) are both USB 2.0-connected devices with PIN-pad. In contrast to the 3621 (Class 2), the 3821 additionally offers a two-line display (Class 3). Both are evaluated to be Common Criteria EAL 3+.

The manufacturer offers closed source-drivers to work with PC/SC as well as with CT-API. Although those drivers work very well, there are currently two noticeable activities to develop open source-drivers.



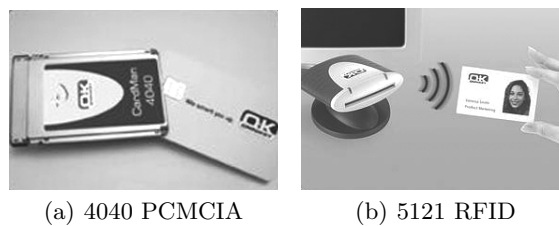
(a) 3621

(b) 3821

**Fig. 3.** Readers from Omnikey

**Omnikey CardMan 4040 PCMCIA** The only tested card reader for mobile computers is the Omnikey CardMan 4040 PCMCIA (see Fig. 4(a)). Omnikey offers PC/SC- and CT-API-drivers for Unix, as well as drivers for PC/SC on Mac OS X [16].

**Omnikey CardMan 5121 RFID** The Omnikey CardMan 5121 RFID (see Fig. 4(b)) is a Class 1-reader without PIN-pad or display, but it offers a build-in RFID-reader and is thereby the only tested hybrid-reader. Drivers for PS/SC and CT-API under Unix are available by Omnikey [17].



**Fig. 4.** Readers from Omnikey

**Axalto eGate Desktop-/Token-Connector** Axalto’s family of eGate-readers (see Fig. 5(a) and 5(b)) takes a special place in this listing. The readers are connected via USB, but do not offer any build-in USB-logic. They only work with Axalto’s “eGate”-cards, which provide the necessary USB-logic on the chip. Hence, these readers are very cheap. Both are Class 1-readers. The Desktop-Connector has the shape of a regular card reader and can take ID-1-fullsize-cards. The Token-Connector has the shape of an USB-stick and can only take cards in SIM-format, which are also available from Axalto as pre-cut versions.

Besides the above mentioned advantage of low-cost-card readers in connection with card-integrated USB-interfaces they offer the possibility to trigger an `xlock` when the card is removed. This can be achieved through hotplug-events, since removing the card also means removing the card reader. This is not possible with any of the other cards and card readers.

**Gemplus 410 serial** Gemplus’ 410 serial (see Fig. 5(c)) does not offer any drivers for Unix. But an open source-driver is available. With this driver it is possible to make use of the `pcscd-daemon`, which is discussed in Section 3.1.

## 2.2 Overview of available smart cards

**TCOS 2.0 Blank** It was not possible with any of the tested open source-software to write own keys to the TCOS 2.0 Blank-card. But since there is



**Fig. 5.** Readers from Axalto and Gemplus

OCF-support for the cards it is at least possible to create own structures on those cards. A proprietary Unix-driver-package is available from Kobil, but that could not be tested successfully due to problems with the middlewares accessing those drivers.

TCOS-cards can be read correctly in card readers from Towitoko, Kobil, Gemplus and Omnikey with all unix tools. The maximum key length is 1024-bit. TCOS 3.0-cards, which will support key lengths up to 2048-bit, will be available in the near future. Integration of Mifare-chips from Philips is directly supported by the manufacturer.

**TCOS 2.0 NetKey** The NetKey-edition of TCOS 2.0-cards differs from the blank card only in a few aspects. They already contain key pairs and are blocked against saving other keys when delivered. Additionally, they contain a set of applications, e. g. for time recording and door opening. It is possible to *read* these cards but not to *write* to them. Especially creating new file structures is not possible. That is why the tested open source-projects cannot deal with those cards. All other characteristics are the same as those from the TCOS 2.0 Blank-card.

**Gemplus GPK 16000** The Gemplus-card offers compatibility with PKCS#15, so the possibility of storing and reading keys and certificates with OpenSC is given. Thereby, smart card-based mail-encryption, browser-authentication and logon to a tty can be established. Only securing an IPSec-endpoint using smart cards is not possible since signature-techniques are required which the card does not offer.

The card can be read and written with card readers from Towitoko, Kobil, Gemplus and Omnikey. An integration of RFID-chips like Mifare is not offered by Gemplus.

**Gieseke & Devrient Starcos 2.3** Gieseke & Devrient's cards devise a very good picture under Unix.

A PKCS#15-conform structure can be generated on these cards facilitating OpenSC. So the desired aims like mail-encryption, browser-authentication and workstation-logon can be established. Furthermore, also the authentication for IPSec-connections works without any problems. These cards also work with all card readers except the ones from Axalto.

With the other cards, the maximum keylength is limited to 1024-bit. An extension of the maximum keylength to 2048-bit is available in form of the successor Starcos 3.0, which has just started shipping, but has not been available for intense testing yet. G&D offer their cards optionally also with integrated Mifare-chips.

**Axalto Cyberflex 32k** Because Axalto's Cyberflex 32k is a JavaCard, it is not possible to use it together with OpenSC. So using PKCS#15-structures [8] with this card is not an option with the given software at the moment<sup>2</sup>.

The card offers the possibility to store keys with a length of up to 2048-bit and to embed Mifare-chips. Yet also interesting is the fact that Axalto also offers cards with an integrated USB-interface (called Axalto eGate) to use them in combination with Axalto's low-cost card readers. Besides that, the card is readable and writable by readers from Towitoko, Kobil, Gemplus and Omnikey.

**Axalto Cryptoflex 32k** The Cryptoflex 32k is the non-Java-based counterpart to the Cyberflex from Axalto. It incorporates a classic smart card-operating system and thus it is writable by OpenSC without problems. Mail-encryption, web-authentication, tty-logon and VPN-authentication work out of the box.

A maximum keylength of 2048-bit is offered as well as the incorporation of Mifare-chips inside the card. The card is also read- and writable by all readers from Towitoko, Kobil, Gemplus and Omnikey and also by the devices from Axalto itself.

### 2.3 Overall comparison

Concluding from this Section we now give a final comparison of the card's hardware-features (see Table 1) and the protection goals feasible with them (Table 2).

The overall outcome shows that more than half of the available cards still lack support for keys longer than 1024-bit. This has to be very well evaluated in large installations, because once 1024-bit keys become weak, those installations have to be completely reprovided with new cards.

## 3 Overview of software-projects

The software projects dealing with smart card-integration can be divided into middlewares for accessing the readers and user-space-projects using these mid-

---

<sup>2</sup> Although steps have been taken to emulate a PKCS#15-structure on the card, but this is still in beta-state)

Hardware-features	Telesec	G&D	Axalto	Axalto	Gemplus
	TCOS 2.0	Starcos 2.3	Cryptoflex	Cyberflex	GPK 16k
PKCS#15	–	+	+	+ <sup>a</sup>	+
ISO-filesystem	+	+	+	+ <sup>a</sup>	+
JavaCard	–	–	–	+	–
Mifare-Chip	+	+	+	+	–
RSA-Keylength	1024	1024	2048	2048	1024

**Table 1.** Hardware-features of tested cards

<sup>a</sup> through emulation-layer in beta-stage

Unix-protection <sup>b</sup>	Telesec	G&D	Axalto	Axalto	Gemplus
	TCOS 2.0	Starcos 2.3	Cryptoflex	Cyberflex	GPK 16k
OpenSC-Support	–	+	+	–	+
MUSCLE-Support	–	–	–	+	–
Unix-Logon	–	O	O	O	O
Mozilla-Suite	C	O	O	O	O
Netscape-Suite	C	O	O	O	O
Mozilla Firefox	C	O	O	O	O
Mozilla Thunderbird	C	O	O	O	O
VPN (IPSec-Auth)	–	O	O	–	–

**Table 2.** Features of tested cards under Unix

<sup>b</sup> caption: C = closed source (commercial), O = open source

dlewares as an abstraction-layer for storing their data on the card. This listing gives a broad picture of what is possible at the moment and which developments can be expected in the near future.

Because all user-space-projects offer command-line tools to access the smart card-based keys for crypto-operations like signing and encryption of files, those features will not be discussed more deeply. They will be taken as given.

Besides the Unix-features we give a short sideview on Windows<sup>®</sup>. We will take a look at how those cards can be integrated and how they can be utilized, i. e. which services can be facilitated.

In the first Subsection we give a condensed overview of existing standards and middlewares for transparent access of applications to the hardware. The remaining subsections deal with one user-space-project each.

### 3.1 Standards and middlewares for accessing cards and readers

**PKCS#11** PKCS#11 [18]—also called cryptographic token interface standard (cryptoki)—describes an API which offers a generic interface to cryptographic tokens. Modules for supporting PKCS#11 are provided by nearly all relevant user-space applications and are easy to integrate in common web- and mail-applications.

**CT-API** CT-API is a lightweight interface to a smart card or reader. It specifies only 3 commands. The first one is the *init* function that enables the communication with the card reader and the card. It must be called first. Then the *data* function, which sends the data either to the card or the reader. This data is formed as Application Protocol Data Unit (APDU) commands. Finally, the *close* function, which ends the communication. All projects (OpenSC, MUSCLE and OpenCA) offer support for using CT-API to access the smart card.

**OCF—OpenCard Framework** The OpenCard Framework [19] provides a Java interface for accessing smart cards and using them with user-space-software. Its compatibility with the PC/SC-standard is a great advantage. Besides the devices from Axalto, all readers were tested as working, including the Kobil-readers. Some filesystem-based cards are officially supported.

**OpenCT** The OpenCT-project [20] offers several ways to access the card reader: as an ifdhandler-driver for PCSC-lite, as a CT-API-driver or as a small and lean independent Unix-service, so applications can use it with minimal overhead. OpenCT implements drivers for several smart card readers, i. e. all of the tested readers and even more. Furthermore, a remote access to export smart card-functionality to another machine running OpenCT is available. OpenCT also has a primitive mechanism to export smart card readers to remote machines via TCP/IP.

Because it is developed by the OpenSC-team, it offers the most stable access to cards and readers for OpenSC-applications. The drawback is the fact that it offers support for only very few card readers at the moment. For example, readers from Kobil, Gemplus and Omnikey are not supported right now.

**PCSC-lite** The PCSC-lite-daemon, maintained by the MUSCLE-project [15], offers transparent access to supported smart cards. It provides a Windows<sup>®</sup>-like SCard-interface to readers and smart cards. Drivers for card readers can be either compiled in or loaded and unloaded during runtime as shared object-files. This gives the great advantage of flexibility, especially when using USB- or PCMCIA-card readers.

### 3.2 OpenSC

OpenSC represents a project [21] focused on the implementation of the PKCS#15-standard proposed by RSA Security. Its heart is the libopensc-library, which is accessible through the user-space utilities `pkcs15-init`, `pkcs15-tool` and `pkcs15-crypt`. `pkcs15-init` is used for initialization of the card and storing data on it, `pkcs15-tool` can be utilized for maintenance of stored keys, PINs and certificates and `pkcs15-crypt` performs signing- and encryption-operations using the card.

**PAM-module** A PAM-module is available from the project's site. With this module, logon to workstations is securable by means of smart cards as well as screen-lockings for active sessions.

**E-mail-security and web-authentication** A PKCS#11-module for Netscape- and Mozilla-suite from OpenSC can be used to secure e-mail-traffic and web-server-authentications. This module makes use of the modularity of the Security Device Manager in both suites and also works with Firefox and Thunderbird.

**VPN-protection** A lot of effort in strongSwan [22]—which is a fork-project of FreeS/WAN-project—has been taken to enhance VPN-logon security. A feature in this project has been developed allowing smart card-based initialization of VPN-connections. This makes use of existing PKCS#11-APIs, so that OpenSC-initialized cards can be used to authenticate a VPN-connection.

**OpenSSH-support** Together with the source code a patch for OpenSSH is provided offering support for smart card-based authentication to remote hosts using secure shell. A modified `ssh-keygen` extracts the public key from the card. This can be stored on the remote host and a patched `ssh`-command accesses the smart card which utilizes the private key for authentication.

**OpenSSL integration** OpenSC features an engine for OpenSSL. By loading `engine_opensc.so` into OpenSSL it is possible to use cryptographic operations in OpenSSL, based on certificates and keys on a smart card initialized with OpenSC.

**Windows<sup>®</sup>-connection** In the recent year a new project called CSP11 has evolved to establish Windows<sup>®</sup>-support for OpenSC. It is a CSP-module (Cryptographic Service Provider [23]) for the crypto-API of Windows<sup>®</sup>. Although this project is still in beta status, it enables Windows<sup>®</sup>-applications like Internet Explorer<sup>®</sup> and Outlook<sup>®</sup> to use smart cards which were initialized by OpenSC.

### 3.3 MUSCLE

The Movement for the Use of Smart Cards in a Linux Environment (MUSCLE) [24] focuses—apart from the maintenance of PCSC-lite and open source-drivers for various card readers—on the usage of JavaCards for different security targets. In recent years attempts have been taken to also support smart cards besides JavaCards, but the focus is still on JavaCards. For this reason only very few non-JavaCards are officially supported. Access to smart cards is only offered by utilizing PCSC-lite.

Tools called `xcard` for command-line-access and its gnome-based counterpart `xcardii` for gnome are responsible for the initialization and maintenance of the card and several plugins are available to implement logon and signing activities.

**PAM-modules** By utilizing the PAM module delivered by MUSCLE the possibility of securing workstation-logons is given.

**E-mail-security and web-authentication** Because a PKCS#11-library also has been written for MUSCLE-cards, cryptographic operations on e-mails and also authentication for web-applications are possible.

**OpenSSH-support** A modified OpenSSH 3.5p1 is offered by the Smart Sign-project (see Section 3.4). This SSH makes use of the MUSCLE-Card-framework, for example the PCSC-lite-daemon. By this, secure shell-logins make use of the private key stored on the user's smart card.

**Windows<sup>®</sup>-connection** Despite from what the meaning of "MUSCLE" may indicate, support for Windows<sup>®</sup>-based systems is available in form of a Cryptographic Service Provider (CSP [25]).

### 3.4 Smart Sign

The Smart Sign-project [26], founded in 1999, mainly deals with the support of OpenCA [27], an open source-certification authority software. By using Smart Sign, OpenCA gains the possibility to store keys and certificates on a smart card. Even support for on-card keypair generation is implemented. Smart Sign utilizes the MUSCLE-card-API to access readers and smart cards. For this reason JavaCards, but only few ISO 7616-cards are supported. For interaction with the card a commandline-utility called `smartsh` is provided, offering access to the reader and the attached smart card.

**PAM-modules** The PAM-module from the MUSCLE-project can be used to facilitate Smart Sign-cards. Due to this fact no approaches have been taken to develop a special PAM-module for Smart Sign.

**E-mail-security and web-authentication** A PKCS#11-module is provided by the Smart Sign-project to authenticate https-connections and to enable e-mail-related crypto-operations. Furthermore, it features the deployment of certificates from OpenCA through this module onto the user's card. This makes releases of new certificates through OpenCA much more comfortable, reduces complexity in the deployment-process and enhances security by means of an encrypted deployment-tunnel.

**OpenSSH-support** As mentioned in Section 3.3 a modified OpenSSH 3.5p1 is available on the project's homepage. It offers a modified ssh-agent, which enables the user to establish ssh- and scp-connections to remote sites by using the private key on his card, that never gets exposed.

**Windows<sup>®</sup>-connection** From the Smart Sign-project only limited support for Windows<sup>®</sup>-based systems is offered. E-mail-security and web-authentication can be deployed through a PKCS#11-module, but there is no CSP-interface available to make use of Smart Sign-cards. Because access to the card is based on PCSC-lite and the MUSCLE Card-framework it is possible to utilize the MUSCLE-CSP.

### 3.5 GnuPG smart card

The GnuPG-project [28] enables access to the OpenPGP Smart Card [29]. Although it makes use of OpenCT for the access to the card, the key-storage and -usage is only possible with the OpenPGP Smart Card and with none of those introduced above. The concept of Web of Trust in GnuPG makes it very attractive for security-aware users who do not want to participate in hierarchical public-key-infrastructures or pay for a commercial certificate.

**PAM-modules** At the moment no PAM-module is available for the GnuPG smart card-functionality. Furthermore there are no visible steps taken to achieve this functionality.

**E-mail-security and web-authentication** The current GnuPG 1.9.19 features build in smart card-support. Through this it's possible to use the key on the smart card in any mail-program from the Mozilla-project [30], since the new Enigmail-plugin for these softwares offers support for smart card-based keys. Since no PKCS#11-module is available no web-based authentication can be established.

**Windows<sup>®</sup>-connection** For the Windows<sup>®</sup>-world the same can be stated as above: only e-mail-encytion can be utilized, since only the Enigmail-plugin makes use of smart card-based keys right now.

### 3.6 Overall comparison

As Table 3 shows, not all smart card-related projects have the same scope. OpenSC aims at covering all tasks and offers a standardized structure for data on the card. MUSCLE's focus is supporting JavaCards, but goals as remote-shell security or VPN-authentication are still not covered. Smart Sign mainly focuses on supporting OpenCA, although it serves nearly all required security features. The GnuPG Card-project is still too young to cover all features, but efforts are made to move into the right direction.

Security-features	OpenSC	MUSCLE	Smart Sign	GnuPG Card
PAM-connection	+	+	(+) <sup>a</sup>	-
E-mail-security	+	+	+	+
Web-authentication	+	+	+	-
Remote-shell security	+	(+) <sup>a</sup>	+	-
VPN-authentication	+	-	-	-
Windows <sup>®</sup> -connection	+	+	(+) <sup>a</sup>	(+) <sup>a</sup>

**Table 3.** Software-features of tested software-projects

<sup>a</sup> See text

## 4 Conclusions

As a result of this paper we can say that a lot of promising software projects are available and applicable. Most of them cover nearly all required security features and are supported by very active communities. Except the GnuPG Card all projects work fine with all combinations of ISO 7618 4-(filesystem-based)-cards and JavaCards get covered by the MUSCLE-project and Smart Sign in a very useful way.

Regarding the compatibility, a lot of big steps have been taken during the last year to make Unix-concerned smart card-projects more platform-independent than before. Except the GnuPG Card all projects have undertaken attempts to support Windows<sup>®</sup>-based systems, although not for every purpose. But a lot of effort is currently made in solving the lack of Windows<sup>®</sup>-support.

Nevertheless, stronger endeavors have to be made to generate a better interaction between those projects. This would result in more flexibility regarding the choice of projects and by that it would give the user the possibility to benefit from each project's advantages.

## References

1. Eckert, C.: IT-Sicherheit, 3rd edition. Oldenbourg (2004)
2. German Federal Network Agency (BSI): Die Lage der IT-Sicherheit in Deutschland 2005. (<http://www.bsi.de/literat/lagebericht/lagebericht2005.pdf>)
3. Kocher, P., Jaffe, J., Jun, B.: Introduction to differential power analysis and related attacks. [www.cryptography.com/dpa](http://www.cryptography.com/dpa) (1998)
4. Okeya, K., Takagi, T.: A more flexible countermeasure against side channel attacks using window method. *Lecture Notes in Computer Science* **2779** (2003) 397–410
5. The OpenPGP Card. (<http://www.g10code.de/p-card.html>)
6. Homepage of the cygwin-project. (<http://www.cygwin.com>)
7. RSA Laboratories: Homepage of RSA Security Inc. (<http://www.rsasecurity.com/>)
8. RSA Laboratories: PKCS #15: Cryptographic Token Information Format Standard. (<http://www.rsasecurity.com/rsalabs/node.asp?id=2141>)
9. Rankl, W., Effing, W.: *Handbuch der Chipkarten*, 4th edition. Hanser (2002)
10. Schneier, B.: *Schneier on Security: RFID-Passports* (Oct 2004). (<http://www.schneier.com/blog/archives/2004/10/rfid.passports.html>)
11. GmbH, K.S.: Closed source-driver download for Kobil-cardreaders. (<http://www.kobil.de/d/index.php?m=support&s=download>)
12. Homepage of Deutsche Telekom AG. (<http://www.telekom.de>)
13. Towitoko: Open source-drivers for Towitoko-cardreaders. (<http://www.geocities.com/cprados/>)
14. Omnikey: Closed source-drivers for Omnikey CardMan 3621 and 3821. (<http://www.omnikey.com/index.php?id=114> (3821))
15. Homepage of the pcsc-daemon for Unix. (<http://www.linuxnet.com/middle.html>)
16. Omnikey: Closed source-drivers for Omnikey CardMan 4040 PCMCIA. (<http://www.omnikey.com/index.php?id=45> (4040))
17. Omnikey: Closed source-drivers for Omnikey CardMan 5121 RFID. (<http://www.omnikey.com/index.php?id=67> (RFID))
18. RSA Laboratories: PKCS #11: Cryptographic Token Interface Standard. (<http://www.rsasecurity.com/rsalabs/node.asp?id=2133>)
19. Homepage of the Open Card Framework. (<http://www.opencard.org/>)
20. Homepage of OpenCT. (<http://www.opensc.org/openct/>)
21. Homepage of OpenSC. (<http://www.opensc.org/>)
22. Homepage of strongSwan. (<http://www.strongswan.org/>)
23. Cryptographic Service Providers. ([http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/cryptographic\\_service\\_providers.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/cryptographic_service_providers.asp))
24. Homepage of MUSCLE. (<http://www.linuxnet.com/>)
25. A Microsoft CryptoAPI Cryptographic Service Provider binding to the PKCS#11-API. (<http://csp11.labs.libre-entreprise.org/>)
26. Homepage of Smart Sign. (<http://smartsign.sourceforge.net/>)
27. Homepage of OpenCA. (<http://www.openca.org/>)
28. Homepage of GnuPG. (<http://www.gnupg.org/>)
29. Commercial site for the OpenPGP Smart Card (Kernel Concepts, Germany). (<http://www.kernelconcepts.de/products/security.shtml>)
30. Homepage of Mozilla. (<http://www.mozilla.org/>)