Different Smartcard-based Approaches To Physical Access Control *

Óscar Cánovas[1], Antonio F. Gómez[2], Humberto Martínez[2] and Gregorio Martínez[2]

[1] Department of Computer Engineering
[2] Department of Information and Communications Engineering
University of Murcia, Spain
ocanovas@ditec.um.es, {skarmeta, humberto, gregorio}@dif.um.es

Abstract Although the basic function of physical access control can be considered simple, that is, to allow authorized personnel to gain access to buildings and deny unauthorized entry, the problem should not be so drastically simplified. In this work, we present two solutions to this problem that make use of special devices named TICA and smartcards. These proposals follow different approaches to manage permissions and authorizations. On the one hand, we present a centralized and on-line solution that makes use of one central database containing the privileges assigned to every user. On the other hand, we describe a system where the authorization-related information is managed in a distributed way. It is an off-line solution based on a RBAC (Role Based Access Control) model that has been designed using authorization certificates.

1 Introduction

Physical security involves the provision of mechanisms to prevent unauthorized physical access to specific facilities, such as laboratories or offices. Moreover, physical access control to computing equipment is especially significant since disabling the servers, routers, or other telecommunications equipment can have a great impact on a large community of people.

Physical access control is a classic problem. In general, a common set of issues exists that must be addressed. First is the problem of key distribution, that is, how to provide the appropriate keys to the authorized users who might need them. In the same way, the keys must be canceled or revoked after some special conditions in order to avoid unauthorized accesses. Then, it is necessary to specify how the user information, i.e. personal information and access privileges, will be managed. Additionally, people use to act on some role during their everyday tasks, and it is important to reflect those roles in the access control management process.

* Partially supported by TIC2000-0198-P4-04 project (ISAIAS), and by 2I02SIU0005 project (m-PISCIS)

Traditionally, the main proposals have been based on a centralized database containing information about valid users. For example, we can imagine a user having some piece of identifying digital data, most probably unique, which is presented to a special device located at the entrance of a particular building or room. The device does not know which keys are valid ones hence it must perform a query to the central database asking about the user privileges. That piece of digital data can be a public key, a distinguished name, an identity certificate, a fingerprint, or simply a login name.

On the other hand, in recent years, public key cryptography has also been proposed as a tool for solving the problems related to authorization and access control. Once the questions about identity in distributed systems have been solved by the X.509 [HFS99] or OpenPGP [CDFR98] standards, determining what the identities should be allowed to do is becoming an interesting research topic. From the *Trust Management* [BFIK99] approach to specifications like SPKI/SDSI (Simple Public Key Infrastructure / Simple Distributed Security Infrastructure) [EFL+99], the main intention is to capture security-relevant information and to bind that information to public keys. Moreover, these proposals might be used to implement some of the RBAC (Role Based Access Control) models proposed by Sandhu [SCFY96]. As we will see, those RBAC models are suitable in order to manage authorization-related information in physical access control systems.

The systems we present in this paper make use of a special device called Intelligent Access Control Device (TICA) [oM01]. TICAs contain smart cards readers and are able to exchange information with an application server. They also carry out authorization decisions regarding information concerning access control policies and digital certificates. It is, currently, one of the main elements of the two systems that allow the establishment of physical access and working time control policies for the personnel in our university. On the other hand, smartcards are used as information repositories, and also as special devices performing cryptographic functions. In general, the authorization information contained in these devices is used by the users in order to demonstrate their rights. As we will see, that information can range from unique identifiers to digital certificates.

In this paper, we propose two systems for physical access control. These solutions are based on a centralized and a distributed access control management system respectively. As we will see, our RBAC and decentralized system provides some additional mechanisms which are not present in our centralized proposal, such as non-repudiation, scalability or dependence of network connectivity.

This paper is organized as follows. Section 2 provides a description of the main elements composing a TICA device. Section 3 outlines the main features of the smartcards we have used to implement our systems. Section 4 presents the design and the implementation of the centralized system. Section 5 contains the details about the distributed authorization management system. It explains how some of the drawbacks related to the first solution can be overcome, and depicts some implementation issues. Finally, Section 6 makes some concluding remarks.

2 Description of the TICA

TICA devices have been developed by our university for physical access control purposes. They are located at the entrances of the different buildings and/or departments. A TICA device is composed of different elements to interact with the environment, the users, and any possible remote application. It uses Linux as operating system, and it includes a Java virtual machine. The processing module is based on a i486DX single board computer (SBC) running at 133 MHz and an interface board based on the PIC-16F877 microcontroller running at 20 MHz. The following elements are connected to the TICA:

- Ethernet port to access the local area network.
- Smartcard (ISO/IEC 7816) compatible reader.
- 4x4 matrix keyboard to introduce commands and personal identification number (PIN).
- 4x20 LCD display to show application messages.
- Beeper for user operations feed-back.
- I/O port to connect sensors (temperature, switches, etc) and actuators (relays).
- Additional ports are available for future or client-specific expansions.

It is worth noting some examples of the functionality of the TICAs. They can open doors, check if the door is opened or closed, check if the box of the device itself is being opened or forced, check the internal temperature, etc. They can also perform additional operations like lighting control, alarm management and presence control.

3 Smart Cards

As stated in the introduction of this paper, one of the most important components we have used to design our system are TICAs –presented in the previous section– which use smart cards readers to access the information contained in the smart cards [FPMY98] owned by final users.

Regarding smart cards, we have designed and implemented the system with Java Cards [mic00b,mic00c] and full-RSA (with a 1024 bits RSA on-board implemented algorithm) smart cards, acting as RSA cryptographic devices and as user private information repositories, containing private keys, certificates, personal identification numbers, and authorization information.

Smart cards are, by definition, electronic devices similar to credit cards except they contain a microprocessor chip that can run programs and store data with a high level of security. This protection is mainly based on the physical protection provided by the smart card owner (who normally stores it in his own wallet) and certain security levels managed by the microprocessor.

This security is defined to protect the information itself and to protect the communication with the outside world. For this last case, the RSA (normal and Java Card) microprocessor cards we have used to develop this architecture are

able, using on-board software stored in ROM memory, to run symmetric (shared keys between sender and receiver) cryptographic algorithms such as DES an 3-DES and thus authenticate its communication with any device sharing the same keys, as it can be the case of external SAM (Secure Access Module) devices [FPMY98]. This security level is complementary to the PIN-based security access level normally used to protect the specific information contained in the smart card, as the user private key or the personal identification number.

It is also very important to remark that this kind of RSA smart cards we are using has an on-board key generation system and then the user is able to generate and use his private keys (to sign or decrypt messages), but unable to export these private keys to the outside world in any way. This is quite important, to avoid the creation of any weak point in the whole centralized and distributed physical access control systems we have designed and implemented.

In the case of Java Cards, that is, smart cards that can manage and run Java applications stored in its own memory, the security level remains the same, with symmetric algorithms for external communications, PIN codes to protect the internal data, and on-board private key generation (unable to be exported in any case). The advantage of this kind of smart cards is based on the possibility of using different Java Card applets [mic00a] to manage properly the SPKI credentials associated to different application environments.

4 A centralized solution

In this section, we are going to present the centralized system which has been used as the starting point to design our decentralized solution. This system is similar to other commercial products existing today, and the point of describing this widely and deployed solution is to present some drawbacks which can be overcome using a different approach. However, it is being successfully used in many scenarios (including our University, where about one hundred TICAs are being used for access control purposes to laboratories, buildings, and car parks), and it addresses the main issues concerning the access control problem.

Key Distribution is performed using unique identifiers. Each user has a unique identifier which is stored into his smartcard. The central database contains one table for every installed TICA. These tables are formed by different records which specify the unique identifiers of the authorized users, the set of permissions assigned to the identifier, and some additional data. When a new user is authorized to perform a specific action with a particular TICA, a new record has to be inserted in the related table.

Key Revocation is very straightforward. If the authorization manager wants to revoke some permissions previously granted to a specific user for a particular TICA, all he has to do is to delete the records of the identifier involved from the database tables.

Roles are not considered. This solution binds permissions to users directly. The introduction of roles might provide the ability of establishing independent relations between users and roles, and between roles and permissions.

The principal guidelines of this type of access control management are shown in Figure 1. During an initial registration phase, the user obtains her smartcard and his unique identifier. He can also request to have access to some specific building or department. Depending on the particular access control policy, the manager might include a record in the database table related with the TICA controlling the requested access. Then, the user inserts her smartcard into the TICA and he requests an specific action (open the door, start working, etc.). Finally, the TICA makes a query to the central database asking whether or not to grant the action.

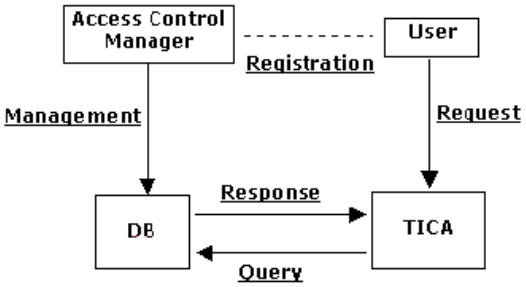

Figure 1. Centralized access control system

The client application that has been placed in the TICA devices has been fully developed in Java. It runs in the SBC board and communicates with the interface board by way of a serial port. Although the application requires being on-line in order to query the application servers, it is designed to be fault tolerant, including the possibility of maintaining a local copy of the authorization-related information in case of a network failure in order to keep on working. Furthermore, it keeps a transactions cache and can be synchronized with a NTP (Network Time Protocol) server, which is important to obtain time-consistent records. It makes use of the following technologies:

- *JavaComm*. Java technology that allows communications with other hardware components by means of serial and parallel ports. In this way, a Java application can communicate with different devices that are not connected to the LAN.
- *OCF (Open Card Framework)*. Java framework that allows a structured access to smartcards in a device independent way [For01]. Thus, TICA devices are able to use both ISO/IEC 7816 smartcards and JavaCards. Moreover OCF allows the use of different smartcard readers with the only requisite of having an OCF driver.
- *SSL* (Secure Sockets Layer). Security protocol [AFK96] used in all transactions between the TICAs and the application servers accessing the central database. These communications are client and server authenticated. Each

time a new TICA device is connected to the network, its manager has to generate a private key for the device and to request its corresponding certificate to the certification authority of the existing PKI.

Although the above presented solution has been successfully used, and it solves most of the common problems about access control, it has some drawbacks that can be fixed following a different approach.

5 Distributed management of physical access control

5.1 Motivation for a RBAC and decentralized system

Our centralized system requires permanent connectivity to the application server providing access to the database. When the connection to the database is broken, or the database is down, the whole system continues providing service, but it is based on local copies of the authorization data. Therefore, further modifications of the database will only be seen by the TICAs remaining connected. In this way, the system becomes unstable since some TICAs might deny some operations that are authorized by others, and vice versa. Besides this, there are some environments where it is not possible to have network connectivity. The access control system might be truly distributed without the need for a central point or permanent network connectivity since it is possible to perform access control decisions without any queries to external elements. In this distributed solution, the access control terminals can be off-line, and they perform their tasks in an independent and decentralized way.

If the environment in which the access control system is used grows large enough, the task of managing each TICA, the list of authorized users, or the local copies of the authorization information become too big. A better solution is defining user groups in order to assign permissions to the group name instead of trying to manage each user individually. However, if group membership is defined by the database, all problems derived by the network dependence will be present here again.

Finally, the design of the current centralized system does not provide a basic security service: non-repudiation. The database and the TICAs store log files containing information about the actions that have been requested by the users (both authorized and unauthorized). However, those log files cannot be used as proof for non-repudiation. We can imagine a user being denied a particular access to a specific building or department. The log files do not demonstrate that the access was indeed performed since they are susceptible to be modified or altered by anyone gaining access to the database. Although there are well known solutions, based on encryption, to store confidential data on insecure platforms, what we really need is a piece of data generated by the user, not by the TICAs or the database itself, which could be used as irrefutable evidence, that is, a digitally-signed access control request. This set od digitally-signed requests, together with the authorization decisions, can be used later for auditing or for forensic purposes following an incident.

5.2 Architectural design of the system

Background This decentralized system is based on a RBAC (Role Based Access Control) model. The central concept of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies management of permissions since the two relations are considered completely independent. Roles can be seen as various job functions in an organization, and users can be assigned to one role depending on their responsibilities. As is commented in [SCFY96], *"the particular collection of users and permissions brought together by a role is transitory. The role is more stable because an organization's activities or functions usually change less frequently"*. We find RBAC a valuable choice in order to design a physical access control system. As we have previously noted, centralized systems become complex when the number of users is high. Complexity can be reduced first assigning users to specific roles, and then determining role permissions. Roles provide several advantages beyond indirection. For example, as we will see, a particular user acting as role researcher has a role membership certificate stored in the smart card, and that certificate does not contain any information about role permissions. Modification of permissions related to the role does not imply modification of the certificate stored in the smart card, and therefore management is greatly simplified.

However, RBAC models can be implemented in different ways. For example, it is possible to design a RBAC system where relations among users and roles (and roles and permissions) are defined as database tables, which are stored in a central server. While problems derived from scalability are reduced, some of them still remain related to central-server dependence and network connectivity.

We have designed a system where each TICA is the beginning of the authorization path, and not only the enforcement point. The device is able to make the security decision regarding the authorization data presented by the user requesting the access. Role membership and the assignment of permissions to roles are encoded in digital certificates, which are distributed among the related users and TICAs. Revocation of such documents is managed using two different approaches. First, it is possible to use short lived certificates [Mye98]. In absence of network connectivity, an off-line device cannot retrieve a regularly-updated CRL, or perform an OCSP (Online Certificate Status Protocol) [MAM+99] query. However, certificates with a short validity interval can naturally bound the effect of revocation, and they constitute a good alternative for this type of scenarios. On the other hand, TICAs are able to retrieve information from external entities since they have an Ethernet port to access the local area network. If network connectivity is possible, TICAs might periodically check a CRL or perform OCSP queries.

Our solution has been designed using the SPKI/SDSI public key infrastructure. The crucial component to a PKI is the key pair, and in SPKI/SDSI the public and private keys are central. In contrast to classic PKI theory, SPKI/SDSI emphasizes that the public key is the principal. Of course, the notion of an owner of a public key is allowed, but is not necessary. In fact, in order to determine

whether an individual has access to a particular building it might not be mandatory to authenticate the user identity.

SPKI/SDSI defines three forms of credential: ID, Attribute and Authorization. An ID credential binds a name (an identity or a group name) to a public key, an attribute credential binds an authorization to an ID, and an authorization credential binds an authorization to a public key. Some RBAC models can be implemented using these types of credentials. ID credentials are useful to implement role membership (two individuals follow the same role if they have ID certificates sharing the same ID), and to bind an identity to a public key (however, we use them only for group membership purposes). Attribute certificates can be used to specify the permission set assigned to a particular role. Finally, authorization certificates are useful in order to establish a direct relation between a public key and an authorization (names are not used).

Architectural elements In this section we are going to give a brief description about the core entities of the distributed management system. We introduce why they are necessary and how they interoperate in a typical scenario.

- *Users.* A user is a person with a smart card containing a RSA or DSA key pair. The public key might be included in a X.509 identity certificate issued by a particular certification authority, but it is not mandatory (i.e. the smartcard might contain no information about the user's identity). The smartcard must store additional authorization, attribute or ID certificates. Users are able to use some of the available TICA to gain access to a particular building, or in order to perform requests associated with presence control mechanisms. They submit a digitally signed request to the TICA specifying the action being demanded and all certificates related to that action (which are stored into the smartcard).
- *Naming Authorities (NA).* They are responsible for issuing ID certificates for the users. This type of certificates can be used to define group (or role) membership, or simply in order to assign a name to a particular public key. The way a NA is able to determine whether a user is a member of a particular group is application-specific. When a role membership request is granted by a NA, the related certificate is stored in the user smartcard, and it can be published in a public data repository.
- *Authorization Authorities (AA).* Authorization and attribute certificates are issued by AAs. There are two possible ways to generate these type of certificates. On the one hand, users can directly request authorization certificates for a particular set of TICAs. If the request is granted, certificates are issued and stored in their smartcards. On the other hand, some special users (system managers) can also request attribute certificates, that is, authorization certificates where the subject is not a particular public key, but a role name defined by a specific NA. Those attribute certificates are normally stored in the TICAs since they are supposed to be long term certificates (role functions do not usually change very often).

– *TICAs.* Our system is primarily based on delegation chains [Aur98], and TICAs are the beginning of the trust path. TICAs can establish their own access control conditions, trusted entities, and authorization mechanisms. Each TICA issues an authorization certificate for a set of specific AAs. These certificates basically give the AAs total authority over the device, and also the permission to further delegate the access control is granted. TICAs can also delegate the authority by means of ACL (Access Control List) entries containing the same information included in those certificates. The main difference is that ACLs are local information not digitally signed, and cannot be published.

Generation of the certificates involved is performed using a distributed credential management system [CG02]. This system addresses some problems related to scalability, certificate distribution, and interoperability. We define how certification requests can be expressed, how different security policies can be enforced using this system, and which are the entities involved in a certification scenario.

Figure 2 shows how the system entities are related. In this scenario, the TICA issues several authorization certificates to different AAs. Those certificates give the AAs a set of privileges over the device and the permission to delegate them. Then, authorization authorities create new attribute certificates giving a subset of such permissions to the roles defined by any of the existing naming authorities. Normally, those certificates have the delegation flag deactivated (we have designed a $RBAC_0$ system, but it might be extended in order to support role hierarchies). Roles are managed by NAs. They issue ID certificates in order to state that a particular user has been assigned to a specific role. Finally, users digitally sign access requests, which are presented to the TICAs together with the digital certificates stored in their smartcards. The request contains a time stamp reflecting the moment when it was formulated, and an authorization tag representing the action being demanded. The path displayed by the figure can be verified by the TICA because it originates from the TICA itself (this kind of situation is called an authorization loop).

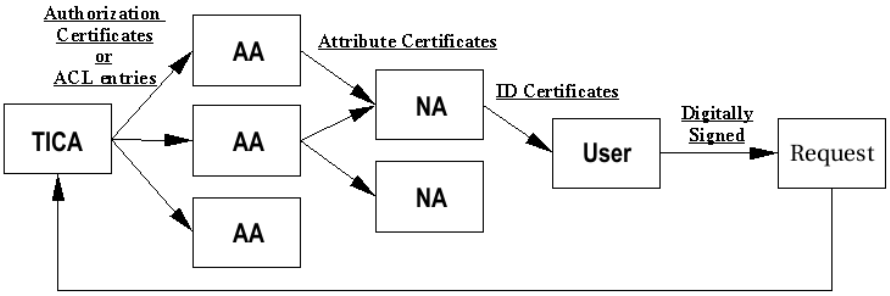

Figure 2. Delegation path

The certificate chain can be also used as non-repudiation proof. As we mentioned previously, our centralized system did not provide irrefutable evidence about access requests. However, in this distributed approach, the authorization sequence is composed by digitally signed statements. Digitally-signed requests can be used to demonstrate that a particular request was indeed formulated, and results obtained from the certificate chain discovery method can determine whether the request was granted. Obviously, existence of an accepting certification path does not prove a door was opened. The important part is that the TICA behavior should be verifiable by a neutral third party. As we will see, the software controlling the TICA is digitally signed and it has been developed using CDSA (Common Data Security Architecture). CDSA provides services which can be used to ensure component integrity and trusted identification of the component's source.

Authorization tags In SPKI/SDSI, credentials are expressed as tuples. Attribute and authorization certificates are specified as 5-tuples (Issuer, Subject, Propagation, Tag, Validity). *Issuer* identifies the entity that issues the authorization. *Subject* is the entity acquiring the authorization (a public key or a group of principals). *Propagation* is a boolean indicating whether the subject is allowed to further propagate the authorization. *Tag* is a free-style S-expression [RL] specifying the permission being provided. Finally, *Validity* specifies the validity period for this credential. ID credentials are defined by 4-tuples (Issuer, Name, Subject, Validity). *Issuer* identifies the entity defining the ID in its private name space. *Name* is the ID. *Subject* is the public key, or the name that will be bound to the ID. *Validity* is the validity period for this credential.

Because the SPKI certificate specification does not give any description of how the authorization information should be presented, we can choose the representation form freely. Moreover, the fact that the exact definition of the contents of the authority field is left application specific is not a problem since the source of a certificate chain and the final verifier are always the same principal.

S-expressions that we have used as authorization tags have the following format:

```
(tag (d-tica (section/tica (permissions (time-interval)))))
```

- *d-tica.* This identifies the tag as a TICA-related authorization.
- *section.* TICAs can be grouped into sections. For example, a section might be the identifier of a particular building or department. In this way, we can specify a set of TICAs using prefixes.
- *tica.* This is the identifier of a particular TICA, as for instance `library/main-door`.
- *permissions.* Permissions being granted, as for instance `open-door, start-working-day, end-working-day, etc`
- *time-interval.* Some permissions can be restricted to specific time intervals.

Example application In this section, we are going to show the elements involved in a particular delegation chain. First we have the following authorization

certificate issued by *tica1*. This certificate gives full authority to *AA1* over the *tica1* located in the *section1*, and the permission to further delegate it.

```
(tica1, AA1, true, (tag (d-tica (section1/tica1 (*)))), forever)
```

Then, *AA1* has two alternatives. First, it might issue another (short lived) authorization certificate to a specific final user *U1*.

```
(AA1, U1, false, (tag (d-tica (section1/tica1 (open-door)))),
                        12/1/01..12/31/01)
```

The second option is issuing an attribute certificate for the *R1* role members defined by *NA1*.

```
(AA1, R1 in NA1, false, (tag (d-tica (section1/tica1 (open-door)))),
                                forever)
```

NA1 issues (short lived) ID certificates for the different members of roles defined by itself. One of those certificates is the following one.

```
(NA1, R1, U1, 12/1/01..12/31/01)
```

When the user *U1* inserts his smartcard into the *tica1*, and he requests the door to be opened, the following signed statement is generated (where time-stamp represents the current instant).

```
(U1, (tag (d-tica (section1/tica1 (open-door (time-stamp))))),
                                time-stamp)
```

Now, *tica1* can use some of the available certificate chain discovery methods [Eli98] in order to determine whether a delegation path authorizing such a request exists. In this particular example, the certificates above presented will authorize the action if it is performed during the specified validity ranges.

5.3 Some implementation details

There are two main applications that have been implemented. First, we have an application that is able to generate SPKI certificates and to store them in smartcards. The other application is the software installed in the TICA devices.

Both implementations are based on CDSA (Common Data Security Architecture) [Cor01]. CDSA is a set of layered security services that provide the infrastructure for scalable, extendible and interoperable security solutions. These security services are performed by add-in security modules. The five basic service categories are: Cryptographic Service Providers (CSPs), Trust Policy Modules (TPs), Certificate Library Modules (CLs), Data Storage Library Modules (DLs), and Authorization Computation Modules (ACs). We used the Intel version 3.14 of that architecture in order to develop a new add-in multi-module which provides CSP and DL services (version 3.14 kindly added a SPKI CL module which supports attribute and ID certificates upon our request).

Our CSP module adds support for smartcards. It performs digital signature operations that make use of the private key stored in the smartcard. In this way, our applications can use the CSSM (Common Security Services Manager) API in order to, transparently, create digital signatures generated by the stored private key. We have also developed a CSP module for smart cards with cryptographic capabilities (digital signature, encryption/decryption). Using this type of smart cards, private keys are protected from external software, which increases system security.

On the other hand, SPKI credentials are inserted and loaded from the smartcard using our own DL module. This module defines the particular database containing the authorization information, and the way data is managed.

6 Conclusions

In this paper, we have presented two different physical access control systems making use of special devices (TICAs) and RSA smart cards. As we have explained, those devices and this kind of smartcards can be used to implement access control using different approaches, ranging from secure queries to a central database, to a distributed offline system making use of authorization certificates. We think that our decentralized solution provides some additional mechanisms in relation to the centralized one, such as non-repudiation, better scalability, and suitability for offline environments. To the best of our knowledge, there is no similar RBAC and SPKI-based proposal for physical access control environments.

References

[AFK96] A. O. Alan, P. Freier, and P. C. Kocher. *The SSL Protocol Version 3.0*, 1996. Internet Draft.

[Aur98] T. Aura. On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, pages 14–26, MA USA, June 1998. IEEE Computer Society Press.

[BFIK99] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 185–210. Springer, 1999.

[CDFR98] J. Callas, L. Donnerhacke, H. Finney, and R.Thayer. *OpenPGP Message Format*, 1998. Request For Comments (RFC) 2440.

[CG02] O. Canovas and A. F. Gomez. A Distributed Credential Management System for SPKI-Based Delegation Systems. In *Proceedings of 1st Annual PKI Research Workshop*, Gaithersburg MD, USA, April 2002.

[Cor01] Intel Corporation. *Common Data Security Architecture (CDSA)*. World Wide Web, http://developer.intel.com/ial/security, 2001.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.

[Eli98] J.E. Elien. Certificate discovery using spki/sdsi 2.0 certificates. Master's thesis, Massachusetts Institue of Technology, May 1998.

[For01] Open Card Forum. *Open Card Framework*. World Wide Web, http://www.opencard.org, 2001.

[FPMY98] J. Ferrari, S. Poh, R. Mackinnon, and L. Yatawara. *Smart Cards: A Case Study*. IBM RedBooks, http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245239.pdf, October 1998.

[HFS99] R. Housley, W. Ford, and D. Solo. *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, January 1999. Request for Comments (RFC) 2459.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *OCSP: Online Certificate Status Protocol*, June 1999. Request For Comments (RFC) 2560.

[mic00a] Sun microsystems. *Java Card 2.1.1 Application Programming Interface*. World Wide Web, http://java.sun.com/products/javacard/javacard21.html, May 2000.

[mic00b] Sun microsystems. *Java Card 2.1.1 Runtime Environment (JCRE) Specification*. World Wide Web, http://java.sun.com/products/javacard/javacard21.html, May 2000.

[mic00c] Sun microsystems. *Java Card 2.1.1 Virtual Machine Specification*. World Wide Web, http://java.sun.com/products/javacard/javacard21.html, May 2000.

[Mye98] M. Myers. Revocation: Options and challenges. In *Proceedings of Financial Cryptography 98*, volume 1465 of *LNCS*, pages 165–171. Springer-Verlag, 1998.

[oM01] University of Murcia. *KRONOS Project*. World Wide Web, http://ants.dif.um.es/kronos, 2001.

[RL] R. Rivest and B. Lampson. *SDSI: A simple distributed security infrastructure*.

[SCFY96] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2), February 1996.