ADoCSI: Towards an Alternative Mechanism for Disseminating Certificate Status Information

John ILIADIS [1,2], **Stefanos GRITZALIS**[1,2], **Dimitris GRITZALIS**[3], **Sokratis KATSIKAS** [1]

[1] Dept. of Information & Communication Systems, University of the Aegean
30 Voulgaroktonou St., Athens GR-11472, Greece
e-mail: {jiliad,sgritz,ska}@aegean.gr

[2] Department of Informatics, Technological Educational Institute (T.E.I.) of Athens,
Ag.Spiridonos St., Aegaleo, GR-12243, GREECE
e-mail: sgritz@teiath.gr

[3] Dept. of Informatics, Athens University of Economics & Business
76 Patission St., Athens GR-10434, Greece
e-mail: dgrit@aueb.gr

Abstract

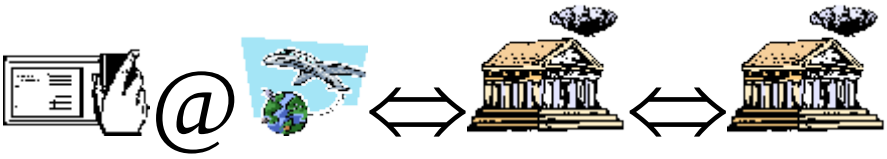
Keywords

Certificate, Certificate Revocation, Certificate Status, Certificate Revocation List, Agents.

1.Introduction

The banking sector has been using electronic transactions at a satisfactory functional and security level for quite a while. Security-aware and capable service providers (i.e. the banks) have been providing their customers with electronic services, and the latter have been used to following simple security guidelines, like keeping magnetic stripe cards at a safe place, keeping PIN codes secret etc. The explosion of Internet usage and the emergence of e commerce and e business introduced a whole new modus operandi regarding electronic transactions:

1. transacting entities do not belong in a closed, controlled group (like bank customers and banks),
2. transactions do not occur within private networks (such as the banking network or other private, corporate networks); instead, public, unsafe and uncontrolled networks are used,
3. a dynamic, constantly increasing, group of entities has the capability to offer services over these public networks; these entities do not always have the capability to perform detailed analyses of the security requirements concerning the services they provide,
4. users do not necessarily comprehend the underlying complexity of the transaction systems they use; most of them are not sensitive or cautious enough when it comes to security decisions [Spruit98].



Cardholder presents card at travel agent's EFT/POS

Acquirer Bank

Issuer Bank

Figure 1: Typical card transaction

Figure 1 : *Typical card transaction*

What banks have accomplished through Automatic Teller Machines and EFT/POS?? machines is to provide their clients with a communication medium the latter consider to be a *trusted path* of communication with their bank. The uniformity of ATMs globally and they fact that users are not required to comprehend the underlying security infrastructure probably contributed to them being so succesful.

Cardholders use their cards in merchants' EFT/POS bank provided machines, and the latter communicate with the Acquirer Bank which, in turn, communicates with the Issuer Bank in order to check revocation status of the card and authorise the transaction. The communication between EFT/POS and Acquirer Bank and the communication between the Acquirer Bank and the Issuer Bank is transparent to the merchant and the cardholder. The cardholder needs only to know its own PIN code, while the merchant needs to know nothing at all on the verification process. Therefore, the security measures protecting the transaction are *transparent* both to the cardholder and to the dependent entity, i.e. the merchant.

Service providers in the Internet have been using PKI lately, in more or less the same way. Certificates and CAs have been used as a means to construct a *trusted path* of communication between client and service provider. So far, it seems to go well. However, what has not been accomplished yet is to hide the complexity of the underlying security infrastructure put in place, in communication between the users of a service and the service provider.

Users –in the Internet– have to comprehend the mechanics of using digital certificates as a means to authenticate. In particular, dependent entities have to verify themselves the revocation status of certificates they are presented with. Mechanisms have been deployed in order to facilitate or automate verification of a certificate's status. Certificates themselves may include fields with URLs pointing to Certificate Revocation Lists (e.g. distributionPointsCrl???) or to Online Certificate Status Protocol responders [OCSP, ??]. Software that makes security decisions based on presented certificates has been developed to use those URLs in order to retrieve revocation information and do the verification by itself, if it is properly configured.

Entities transacting over the Internet, both dependent ones and certificate holders, are rarely security experts. One should not demand from them understanding of the security measures put in place in order to protect their transactions. Neither the certificate holders (authenticating entities or signers) nor the dependent entities should have to contribute manually to the location and validation of Certificate Status Information (CSI). That shouldn't be the job of certificate holders or dependent entities, as it is not the job of real world merchants to locate or validate credit card revocation information.

Certainly, one could argue that it should be the responsibility of certificate holders to provide dependent entities with the most current CSI [Rivest98]. Rivest also claims that it should be the responsibility of the dependent entities to set the requirements CSI they are presented with has to meet. Although entities that offer services over the Internet can very well be security aware, much more than the certificate holder on an average, that does not make them security experts. Furthermore, dependent entities as well as certificate holders are usually not so knowledgeable, motivated and perceptive [Spruit98] regarding security controls. They don't always follow good security practice. Therefore, certificate holders and dependent entities are bound to make security mistakes or overlooks, consciously or unconsciously, leading to direct or indirect failing. Human failing is in many cases the reason for the malfunction of a security mechanism, the incorrect use of a security service or a security breach [Spruit96]

Furthermore, PKI users sometimes have to make security decisions in order to facilitate certificate path validation. The number of questions has to be restricted and the way they are posed has to be studied carefully, otherwise security layer transparency will cease to exist and users may fall back to a position where they should strive to understand the mechanics of CSI mechanisms. This often leads computer users to direct or indirect failing [Spruit98] or 'authorisation fatigue'.

If we want to make sure PKI users receive complete and timely CSI concerning the transactions they perform, we have to assign the responsibility for CSI location and validation to Certification Authorities, in analogy to the banking sector paradigm.

PKI seems to be gaining users' confidence; however, PKI enabled services are not yet security transparent enough. Users do not always comprehend the complexities of security mechanisms, and they should not be obliged to, after all. *Transparency* of security mechanisms is a solution to these problems. However, a transparent mechanism for the dissemination of CSI must not lack the security features provided by the other non transparent mechanisms. We claim an agent based mechanism could meet those requirements. An agent based mechanism responsible for locating and validating CSI, leveraging this burden from dependent entities and certificate holders.

In this paper, we present the prototype of an alternative mechanism for disseminating certificate status information, which we call ADoCSI (Alternative mechanism for the Dissemination Of Certificate Status Information [Iliad99]. Our mechanism is based on Software Agents [??], and its use promotes security transparency in PKI enabled services over the Internet.

This paper is organized as follows: In section 2 we present a short overview of Software Agents. In section 3, we introduce our mechanism, ADoCSI. Section 4 contains our thoughts for future work on the mechanism, while in section 5 we draw our final conclusions.

2.Agents

Agents have recently been (and still are) a major research topic, both for the academia and for the industry, for various reasons [Petrie97]. The researchers, trying to define accurately their view and specifications of Agents, came up with a variety of alternative names or adjectives. A short review of those follows [Nwana96], [Chess95]:

1. *Static*. Agents that do not transport themselves to execution environments other than the one they were originally. They remain at that execution environment and use other methods in order to communicate with other agents, such as RPC,

2. *Mobile*. These agents can transport themselves to other execution environments in order to communicate locally with other agents and at the same time they can preserve the external state they had in the previous location,

3. *Deliberative*. Agents that contain internal reasoning and collaborate with other agents in order to achieve their goals,

4. *Autonomous.* Agents that have states and goals and do not need user feedback or interaction (except perhaps an initial user feedback) in order to carry out their tasks and reach their goals,

5. *Co-operative.* Agents that co operate with other agents in order to reach their goals,

6. *Interface* (also called User). Agents that sit between another agent and a user. The user communicates with the interface agent only, and the latter transforms the user's goals into formal statements and procedures and assigns those to the other agents; the interface agent launches agents on behalf of the user, co-ordinates them and provides the results of their work back to the user,

7. *Heterogeneous.* Agents that encompass several of the characteristics of the other agents and can talk to other agents through the use of a standard agent communication language, thus providing for interoperability between agents.

This wide spectrum of names result in semantic overlap and confusion. Furthermore, since there were no criteria for 'agenthood', some researchers came up with software constructs which they called Agents, even though they did not have any of the functionality that characterises Agents [Petrie97]. Genesereth [Gene94] argues that the criterion for 'agenthood' is a behavioural one; a software application is an agent if and only if it communicates with other software applications, using an Agent Communication Language (ACL). According to Genesereth, typical object oriented applications cannot be characterised as agents since the messages exchanged between the objects may have different semantics when exchanged between different objects. On the contrary, the semantics of a language construct in an ACL is always the same, no matter which two agents is it that this message is being exchanged between. There are many criteria one can use in order to develop a taxonomy of Agents. However, the lack of a widely accepted standard practice which could act as a reference point, renders these taxonomies rather inaccurate or at least not in accordance with other ones.

We should clarify at this point that in this paper we are not interested in any Artificial Intelligence properties Agents may have. We are only interested in the distributed functionality they offer, since we can use that in order to meet some of the CSI dissemination criteria [Iliad00b] that other CSI dissemination mechanisms do not meet.

The agents we describe in this document are mobile, deliberative, heterogeneous and (some of them) interface agents. However, since we are not interested in establishing a new name for the agents we discuss, we will be referring to them simply as Agents or CSI Agents. We would like to warn the reader that the term CSI Agents does not constitute a new flavour of Agents. We will use it as a reference name to the mechanics of our Agent construct throughout this document. The agents we will describe must use a standard way of talking to each other. One of the examples of languages used by agents to communicate is the Knowledge Query and Manipulation Language (KQML) [Lamb97], [KQML], which describes programmatic content and the Knowledge Interchange Format (KIF) [Gene92b], [Chess95] which can be used to form knowledge representations, express tasks and goals. Another example is the more recent Agent Communication Language (ACL) of the Foundation for Intelligent Physical Agents (FIPA) [FIPA98b???more recent??]. The advantage of using such a language for agent communication instead of messages with proprietary content format is that it will be possible for the agents involved in our scheme to be updated transparently by their owners or developers, without affecting the interoperability or agent inter-communication. CSI Agents could expire, and their owners or developers will have to (transparently) replace them with new ones at that time (see section 3.5). These actions are not going to affect the operation of our scheme, since the communication between agents will remain the same, being based on one of the aforementioned agent communication languages.

The agents we will present must meet the following requirements:

1. they must be able to suspend execution and resume it at another execution environment,
2. they must retain their state, when transporting themselves to other execution environments,
3. they must be able to create child agents and deploy them,
4. they must be able to select a network location, out of a list of locations, with the least network congestion,
5. they must be able to communicate the retrieved information back to their owner or to their owner's application that spawned the agent.

We should note here that we are not going to examine the mechanics of the implementations of Agent infrastructures. Implementations exist [Frost96], [Aglets], [Odyssey] that meet the requirements set in this section, as far as Agent design and operation is concerned.

3.ADoCSI

In this section we present the prototype of an alternative mechanism for disseminating CSI. We shall be calling this mechanism *ADoCSI* (Alternative Dissemination Of Certificate Status Information). The Agents described in this mechanism exchange CSI, in one (or more) of the formats described in [Iliad00a]. This mechanism makes extensive use of the properties and functionality of the CSI mechanisms presented in [Iliad00a], and provides at least one additional feature: the *transparency* in disseminating CSI [Iliad00b].

Let us consider the entities that take part in our mechanism (Figure 2??):

1. **CA**: The three CAs in Figure 2 offer the standard services a CA should offer. In addition to that, they develop the CA-CSI Agents and the User-CSI Agents, which are used throughout ADoCSI for the dissemination of CSI
2. **Agent Meeting Places (AMP)** [Chess95]: They are also called Agent Platforms (AP) [FIPA98a]. These provide the physical infrastructure where agents can be transported to and communicate with other agents that are already there. In ADoCSI, AMPs must also possess a X.509v3 certificate which will be used in order to authenticate the AMPs against the Agents that visit them. If CAs decide to have their CA-CSI Agents installed in more than one AMP, the agents sent by the dependent entities in order to retrieve CSI can select to visit a specific AMP if it serves the least number of CSI requests and if the communication lines to that AMP's direction are less congested to others. AMPs could be operated and controlled by the CAs themselves or they could be a separate Trusted Third Party.
3. **Directory Facilitator Agent (DF)**: These agents are resident (static) within the AMPs. They keep a complete and up to date register [Gene92], [FIPA98a] with the services offered by the Agents that are in the AMP and the methods other Agents should use in order to communicate with the former. In our mechanism, we assume that the DF will also be able to perform verification of the digital signatures on Agents. However, in an implementation it will most probably be another Agent inside the AMP that will do this job. We assume that the DF can undertake that task as well, in order to reduce the complexity of our ADoCSI specification.

4. **Dependent entity**: An entity that wishes to verify the validity of the certificate of another entity. The latter may have tried to authenticate (authenticating entity) against the dependent entity in order to access the services offered by the dependent entity, or the latter may have received offline, signed communication (e.g. e mail) from another entity (signer). In both cases, the dependent entity wishes to verify the validity of the certificate presented by the aforementioned entities (authenticating entity or signer).

5. **Authenticating Entity**: An entity that attempts to authenticate against the dependent entity in order to gain access to the services offered by the latter.

6. **Signer**: An entity that has sent to the dependent entity a signed piece of information.

7. **Certification Authority Certificate Status Information (CA-CSI) Agent**: This Agent is developed by the CA and it contains the latest available CSI, in one (or more) of the formats discussed in [Iliad00a]. CA-CSI Agents have short validity periods. Entities communicating with CA-CSI Agents can verify the validity period of such an Agent because the CA-CSI Agent contains a timestamp (time of CSI inclusion in the Agent) and a validity time period, and the Agent is signed with the CA private signing key; CAs issue new Agents if their validity period is close to the expiration date or if there is fresher CSI available at the CA, compared to the CSI stored in the Agent. Once a new Agent is issued, it is sent to the AMP. In case the Agents also serve CSI that is very frequently updated (e.g. Freshest CRLs, [??]), then this CSI can be stored externally, in the AMP, in a location where Agents can have access to. User-CSI Agents communicate with CA-CSI Agents at the AMPs in order to locate, validate and retrieve the CSI they are interested in. If CA-CSI Agents can deliver CSI in more than one ways or formats, then they negotiate with the requesting agents in order to deliver the CSI in a manner that meets best the CSI requirements set forth by the requesting agents, like freshness of information and maximum volume of information. In order to increase the availability of the system, CAs may decide to have their CA-CSI Agents installed in more than one AMP, thus allowing for load balancing of the communication traffic due to CSI requests.

8. **User Certificate Status Information (User-CSI) Agent**: This Agent is also developed by the CAs and it is given to the dependent entities, that is any entity that wishes to use the agent based scheme in order to retrieve CSI. This Agent is signed by the dependent entity every time the Agent is sent to locate and retrieve CSI. This signature is used throughout the Agent's life in order to verify its integrity and origin.

9. **Interface Agent**: The Interface Agent (IA) interacts with the User-CSI Agent and the applications that the dependent entity is using. IA generates and launches a new User-CSI Agent whenever the aforementioned applications need new CSI. Furthermore, it is the Interface Agent that installs the retrieved CSI at the local repository of the dependent entity, perform any certificate path validation needed and collaborate with the applications the dependent entity uses in order to inform the entity of any progress or changes concerning the requested or retrieved CSI. IAs that interact with specific software applications should be developed by the software companies that develop the aforementioned software applications.

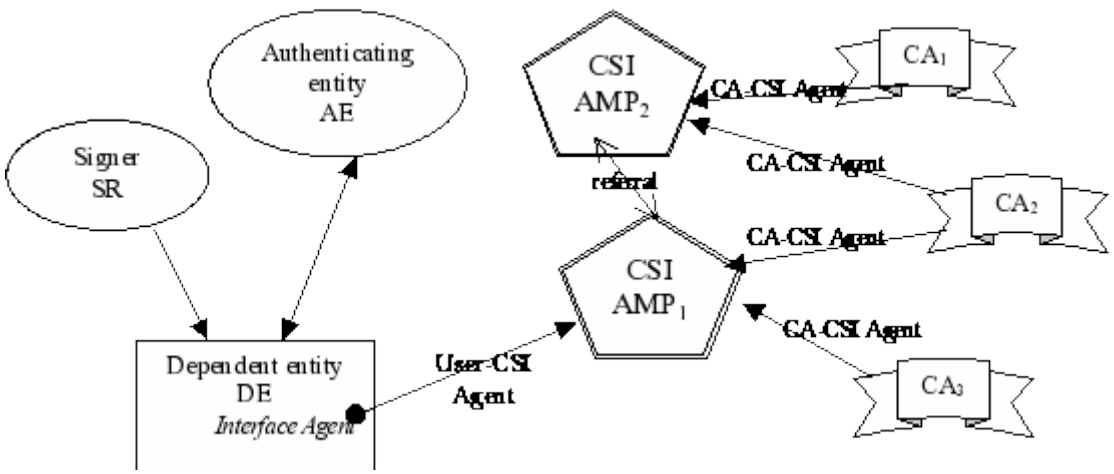


Figure 2: ADoCSI infrastructure

3.1AMP Location Function

The AMP location function can be implemented in the following ways:

1. location information is stored at the CA, and the User Agent will have to retrieve it before transporting itself to the AMP,

2. location information is stored in the CA-CSI Agent and it is given to the User-CSI Agent at the AMP,

3. location information is stored at the certificate of the authenticating entity (or signer) and the Interface Agent will have to retrieve it and deliver it to the User-CSI Agent,

4. location information is stored in the CA certificate and the Interface Agent will have to retrieve it and deliver it to the User-CSI Agent,

5. location information is stored in the User-CSI Agent.

The location of the AMP or AMPs that host the CSI Agent of a CA can be stored in a repository maintained by that CA, such as the Directory. The location information should be signed by the CA in order to enable the dependent entity to verify its integrity, unless the communication protocol supported by the CA repository includes integrity protection. If this is the way the location function is implemented, the User-CSI Agent will have to lookup the location information at the CA repository and verify the CA signature on that, before transporting itself to the AMP. A possible enhancement could be caching the location information in the local user environment, in order to avoid having the User-CSI Agent lookup the location information every time, before transporting itself to the AMP.

However, a CA may decide to stop trusting a specific AMP for hosting the CA-CSI Agent. In this case, the User CSI Agent who is using the cached location information will not know about it. This is one of the cases where the short lifetime of CA-CSI Agents could prove to be useful. If a CA decides to trust no longer an AMP for hosting the CA-CSI Agent, the CA will no longer update the CA-CSI Agent in that AMP, therefore even if a User-CSI Agent visits the specific AMP, it will not meet a non expired CA-CSI Agent.

A minor variation of the location function exists. This can be done in case the CA decides that online communication between the dependent entities and the CA should be avoided or at least should not be frequent. The location of the AMP or AMPs that host the CA-CSI Agent can be stored inside the CA-CSI Agent himself. Certainly, the dependent entity would have to retrieve this information for the first time from the CA, manually. Once this is done and the User-CSI Agent knows which AMP to visit for the first time, it will cache at the local storage area of the dependent entity that information and update it whenever the CA-CSI Agent gives the User-CSI Agent new location information.

AMP location information can also be stored in the X.509v3 certificate delivered by the authenticating entity (or signer) to the dependent entity. The Interface Agent can retrieve that information from the certificate and deliver it to the User-CSI Agent, before sending the latter to the AMPs in order to retrieve CSI on the aforementioned certificate. In this case, the User-CSI Agent does not have to find the AMP location from the CA or the CA-CSI Agent, therefore the communication and processing costs related to the actions of User-CSI Agent are decreased. However, the costs associated with distributing the certificates themselves are increased in this case, since the certificates contain additional information. In this case, the dependent entity is open to Denial of Service attacks, since the location information can only be changed (by the CA) whenever a new certificate is issued (or renewed). Therefore, if all AMPs mentioned in the certificate cease offering services, or if the CA stops trusting them and therefore stops sending updated CA-CSI Agents to those AMPs, the User-CSI Agent of the dependent entity will not be able to locate an AMP to retrieve CSI from.

A variation of the aforementioned location function is to store the location information in the CA certificate. In this case, the costs associated with the communication and processing done by the User-CSI Agent are also decreased, and costs associated with distributing the certificates are not that increased since it is only the CA certificate that contains additional information, not all the certificates issued by the CA. However, the dependent entity is more vulnerable to Denial o Service (DoS) attacks, because the validity period of a CA certificate is longer than that of any other certificate issued by that CA.

A way to deal with possible DoS attacks related with the last two types of location functions would be to have the first type of location function operational as well. If the third or fourth type of location function fails (DoS), the first type of location function can provide the dependent entity with the AMP location that is needed.

The CAs could develop User-CSI Agents in such a way that the Agents themselves decide which location function to use, every time they are called to retrieve CSI. This decision should be based on criteria like which location functions are available, which location function was used the last time the Agent was called to retrieve CSI, whether that function executed successfully or not, and user preferences regarding the location function to be used.

Finally, the location information could be stored inside the User-CSI Agent. The Agent would not need to locate and retrieve the AMP location information. His operation will be simpler and the size of the User-CSI mobile code application will be smaller, therefore communications costs related to the transport of the User-CSI Agent will reduce.

3.2CSI Location, Validation and Retrieval

Once the User-CSI Agent is in the AMP, it contacts the Directory Facilitator in order to locate the CA-CSI Agent that holds the CSI he is interested in. The User-CSI Agent has to form a query in a predetermined Agent Communication Language and send it to the Directory Facilitator, who will reply to the User-CSI Agent using the same ACL as well. If the CA-CSI Agent that contains the information the User-CSI Agent is interested in, does not reside in that AMP then the DF should refer the User-CSI Agent to another AMP.

DF maintains a database with all the CA-CSI Agents that reside in the AMP and the respective CSI they hold. Whenever a CA-CSI Agent is transported to the AMP by the respective CA, it has to contact the DF in order to inform the latter of the CSI it holds. The DF should also update the aforementioned database by polling the resident CA-CSI Agents. This ensures that even if the DF was unavailable at the time a new or updated CA-CSI Agent tried to contact the DF or if a CA-CSI Agent was removed abruptly (e.g. accidental cease of operation, perhaps due to a software bug) from the AMP, the DF will become aware of that and will have the database updated with the relevant information. A CA-CSI Agent may have to be removed from the AMP because the CSI it contains is no longer valid or because the Agent has expired.

Having located the CA-CSI Agents they seek, User-CSI Agents communicate with them and perform a series of queries, using a predetermined ACL. Since a CA-CSI Agent could deliver CSI in a number of different formats [Iliad00a], User-CSI Agents will negotiate with the CA-CSI Agents in order to have the latter deliver the CSI in a way that meets best the requirements set by the dependent entity. The negotiation parameters that can be used, include:

1. *Freshness of CSI*. The freshness of the CSI received by a User-CSI Agent may vary, depending on the mechanism the CA-CSI Agent used to deliver that information,

2. *Length of CSI*. The length of the CSI received by a User-CSI Agent may vary, depending on the mechanism the CA-CSI Agent used to deliver that information. There are mechanisms (e.g. CRL) where the communicated CSI contains always more information than necessary (i.e. CSI on certificates that the dependent entity is not interested in),

3. *Estimated time of CSI delivery to the dependent entity*. The time that is estimated to take for the CSI to be delivered to the dependent entity varies, depending on the length of the information and the availability and bandwidth of the network that connects the AMP and the dependent entity. The User-CSI Agent should be able to evaluate the bandwidth and availability of the aforementioned network and decide on the time it will take to carry the information back to the dependent entity. Furthermore, the User-CSI Agent should request from the CA-CSI Agent for CSI that meets specific length restrictions, in order to be able to transport it back to the dependent entity within the specified time period,

4. *Delegation of certificate path validation*. This validation can be performed either by the CA-CSI Agent or the Interface Agent, depending on the mechanism used for delivering the CSI to the User-CSI Agent,

5. *Bounded revocation*. The User-CSI Agent may choose to prefer one CSI dissemination mechanism to another because the former supports bounded revocation [Iliad00b].

Once the User-CSI Agent has at his disposal CSI, it must validate it. The Agent must verify that CSI it obtained, concerns the certificate it has been instructed to find CSI on. If it is so, the Agent must transport itself back to the dependent entity. If the latter is not connected to the network at that point of time, the User-CSI Agent should be allowed to remain at the AMP for a predetermined maximum amount of time, waiting for the dependent entity to connect back to the network.

3.3Certificate Path Validation

Validation of a certificate path can be performed by the Interface Agent or it can be delegated to the CA-CSI Agent. In any case, the dependent entity should be able to verify the authenticity of the received CSI. If the certificate path has been validated by the CA-CSI Agent, the dependent entity should be able to verify that the received, processed, CSI is authentic; if the certificate path is to be validated by the Interface Agent, the dependent entity should be able to verify the authenticity of the certificate path information received.

3.4Interface Agent

The role of the Interface Agent (IA) is to provide a link between the User-CSI Agent and the PKI aware applications the dependent entity uses. The IA has access to the certificate and CSI repository of the dependent entity (e.g. the computer filesystem), which it updates as soon as new CSI becomes available. Furthermore, the PKI aware applications that reside at the site of the dependent entity communicate with the IA in order to request CSI on specific certificates. It is the IA that will then form a CSI query, using a predetermined ACL, describing the CSI that has to be retrieved and the retrieval negotiation parameters. Also, the IA may inform the User-CSI Agent on the location of the AMP to visit, if that information is stored in the certificate of the authenticating entity or signer, or in the respective CA certificate. Consequently, the IA spawns a User-CSI Agent and assigns to it the task to perform the ACL query. Once the User-CSI Agent returns to the dependent entity location, it communicates with the IA in order to deliver the CSI that has been retrieved. The IA must be able to identify the User-CSI Agent, once it comes back, verify his integrity, recall the query it was assigned with and examine whether the User-CSI Agent did return with the requested CSI. If certificate path validation has not been delegated to the CA-CSI Agent, the IA will perform the certificate path validation or leave this task for the application itself, if the application supports certificate path validation. The IA will install the new CSI in the certificate and CSI repository of the dependent entity and inform the application of the new CSI that is available.

In case the dependent entity wishes to locate CSI in order to verify a certificate of a signer who sent a signed document to the dependent entity, then there is no online communication between the certificate holder and the dependent entity. Therefore, there is no need for online location, retrieval and validation of CSI. If this is the case, the Interface Agent should send the User-CSI Agent to search for CSI on that certificate and notify the dependent entity (the human operator) that he may logoff the network if he wishes so. The User-CSI Agent must be checking frequently (polling) for the existence of the dependent entity in the network and return to it when the dependent entity connects back to the network. In this case, the User-CSI Agent may return to the dependent entity by other communication means as well (e.g. e-mail). It is the duty of the IA then, to check frequently the mailbox of the dependent entity and locate any User-CSI Agents there and retrieve them.

3.5ADoCSI Security

We have presented in the previous sections an alternative mechanism for the dissemination of certificate status information, which we call ADoCSI. In this section, we present the threats this mechanism faces, and the respective security services and mechanisms that have to be in place in order to deal with these threats. The threats ADoCSI has to deal with are the following [Gritz97]:

3.5.1Unauthorised modification or replacement of Agents

An entity other than the CA could manage to modify a CA-CSI Agent before it is installed in the AMP, or install one that has been developed by that entity itself, in order to have that Agent give incorrect or false CSI to the User-CSI Agents. To deal with this threat, Agents should be distributed in a secure manner [Zhang97]; the integrity and origin of the CA-CSI Agents should be verifiable both by the AMPs and the User-CSI Agents. This can be done if the CA-CSI Agents are signed by the CA that sent them to the AMP. Both the AMP and the User-CSI Agents must have a copy of the CA certificate. The DF of that AMP and the User-CSI Agent must use the respective CA public key and verify the signature on the CA-CSI Agent. The AMP must contain the CA certificates of the CAs it receives CA-CSI Agents from, in a repository local to the AMP. The User-CSI Agent must also contain the CA certificates that will be needed to verify the digital signatures on the CA-CSI Agents the User-CSI Agent will communicate with.

An entity other than the dependent entity may attempt to modify or replace the User-CSI Agent while the Agent is distributed to the dependent entities by the CA that developed it, while the Agent is transported to an AMP or while the Agent is returning to the dependent entity. The CA that developed the User-CSI Agent should sign that Agent as well, in order to deal with the aforementioned threats. If the signature of that CA is on the User-CSI Agent, the dependent entity and the DF will be able to verify the integrity of the Agent once they receive it.

3.5.2Unauthorised modification of information contained in the Agents

The CA-CSI Agents transport the CSI from the CAs to the AMPs. The integrity and origin of the CSI contained in the CA-CSI Agents can also be verified, by verifying the CA digital signature on the CA-CSI Agent.

This signature might prove to be redundant for some cases of CSI (e.g. CRLs) where there is already a mechanism for verifying the integrity of the CSI (e.g. in the case of CRLs it is, again, the CA signature on the CRL).

The User-CSI Agents also carry information. When the User-CSI Agent leaves the dependent entity to be transported to an AMP, it carries information it will use in order to form CSI queries it will direct to the CA-CSI Agent. A malicious entity might modify that information while the Agent is transported in order to prevent the dependent entity retrieve correct or complete CSI. This can be prevented if the dependent entity signs the Agent as well, before transporting it to the AMP. The AMP must verify both signatures (CA signature and signature of dependent entity) before allowing the execution of the User-CSI Agent.

If an AMP does not contain all the CA-CSI Agents a User-CSI Agent needs to contact, the AMP will refer the User-CSI Agent to another AMP in order to find the rest of the CSI it is looking for. The User-CSI Agent may be modified by an entity (without authorisation) while it is transported to that other AMP. In order to prevent that from happening, the AMP that refers the Agent must sign the Agent as well, in order to provide integrity protection for the CSI the Agent has retrieved from this AMP.

Finally, when the User-CSI Agent has gathered all the CSI it needs, it transports itself back to the dependent entity, either immediately or whenever the latter is connected again to the network. The Agent carries at that point some CSI for which the dependent entity cannot verify the integrity. This is the CSI the Agent retrieved from the AMP before transporting itself back to the dependent entity. The User-CSI Agent could generate at that time a symmetric key, encrypt that key with the public key of the dependent entity (the certificate of the dependent entity is contained in the User-CSI Agent) and store it along with the other information it carries. Before leaving the AMP, the User-CSI Agent should generate a Message Authentication Code (MAC) [Ford94], using that symmetric key; that MAC should be applied on the Agent itself and all the information it carries. Using that MAC, the dependent entity (the Interface Agent) can verify the integrity of the User-CSI Agent and of the information it retrieved for the dependent entity.

Verification of the origin of the CSI a dependent entity receives may be possible or not, depending on the CSI mechanism that the User-CSI and CA-CSI Agents decided to use in order to exchange the CSI that the former was looking for (see [Iliad00b], for the evaluation of those mechanisms concerning their capability for verification of the origin of CSI).

3.5.3AMP masquerade

An entity may attempt to masquerade as an AMP and attract User-CSI Agents. If a User Agent communicates with a (valid) CA-CSI Agent at an AMP that is controlled by an unauthorised, untrusted and possibly malicious entity, then that entity may manage to take control of the communication between the User-CSI Agent and the CA-CSI Agent and provide the former with false, inaccurate or partial CSI. Even if the two aforementioned Agents employ integrity mechanisms for the protection of their communication, while they are in the AMP, the AMP can always observe the Agents' actions, monitor the data they generate (e.g. symmetric keys that will be used for the protection of the integrity of the communication between these two Agents) and manipulate the communication between them. In order to prevent AMP masquerade, both the CAs and the dependent entities should encrypt their Agents with the public key of the AMP they intend to send their Agents to, before transporting them. Thus, the AMP will be able to decrypt and execute the Agents only if it is the AMP that the CAs or the dependent entities intended to send their Agents to.

3.5.4Denial of Service

Denial of Service attacks could be launched against the AMPs. In order to deal with this threat, the CA-CSI Agents should be sent by the respective CAs to more than one AMPs. Therefore even if an AMP cannot deliver the CSI dissemination service with the expected quality (e.g. the communication line that leads to the AMP is congested due to the DoS attack, or the User-CSI Agents that reside at the AMP perform too many CSI requests to the CA-CSI Agents, thus consuming much of the processing power of the AMP), a User-CSI Agent may select another AMP to retrieve CSI from.

Another countermeasure that can be used in order to deal with DoS attacks would be to minimise the Central Processing Unit (CPU) time quota that is available to every User-CSI Agent that resides in the AMP, thus reducing the possibility that a malicious User-CSI Agent could use too many system resources. Research is being performed on limiting the CPU time that Java mobile code can consume [Gorri98].

3.5.5Replay attacks

A malicious entity might attempt to capture a User-CSI Agent when it is transported back to the dependent entity and send another User-CSI Agent in its place, an Agent that was sent at an earlier date by the same dependent entity. If this is achieved, the dependent entity might be led into believing that the Agent it received is authentic, as well as the information it carries. However, that CSI will be old and possibly invalid. If that CSI has not been

generated with a mechanism that supports bounded revocation, then the dependent entity will consider the invalid CSI it has received as valid. In order to avoid that, the dependent entities should include a nonce (Number used ONCE) [Schn96] in their User-CSI Agents in order to prevent such Agent replays.

A malicious entity could attempt to capture the CA-CSI Agents on their way to an AMP and replace them with valid, but old, CA-CSI Agents of the same CA. If this replay attack was successful, then the User-CSI Agents could retrieve old, and possibly invalid, CSI. However, the CA-CSI Agents are timestamped and they also contain a validity period, which is short. If an unauthorised entity attempts to replay an old CA-CSI Agent to an AMP then the AMP will dismiss that CA-CSI Agent as expired.

If an entity had taken control of an AMP it could attempt to store in that AMP only old CA-CSI Agents, with the purpose of disseminating old, and possibly invalid, CSI. The User-CSI Agents would not retrieve CSI from those expired CA-CSI Agents because the former would realise that the latter are expired (by verifying the timestamp and validity period they contain) Agents and therefore they should not trust the CSI they carry.

3.5.6Malicious Agents

Signing the Agents with the private keys of the Agent developer or the entity that is sending the Agent does not guarantee that the Agent will not act in a malicious way either towards the AMP or towards other Agents. This measure provides only accountability for the actions that the Agent will perform, on behalf of the entity that sent the Agent. Protecting the AMP and Agents, from other Agents has been a topic of research both in academia [Necu98], [Necu96], [Wil98], [Wil99] and in the commercial world [Gong98], [Gong97].

Solutions to this problem include the use of tamper resistant hardware [Wil98], [Wil99] and the use of language semantics in order to verify the safety of a specific application [Gong97], [Necu98] either dynamically, at run time, or statically. Furthermore, access control mechanisms can be employed in order to restrict access to the resources or services offered by an Agent [Gong98]. Limiting the resources of an AMP that an Agent can use, in order to carry out certain tasks, is a subject that is also under research [Gorri98].

Besides these, confining the use and transportation of Agents inside network domains [Gritz98], and enforcing security policies for their operation could contribute to a controlled execution environment for these Agents. Finally, the integration of application development [Jacl97] platforms that can be used for the development of Agents could lead into an integration of the respective security functionality. Such an integration could give to the developers more tools to address the security issues that arise from the operation of Agents [Gritz98].

4.Future work

Protecting Agents from AMPs is a still a subject of research [Wil98], [Wil99]. Perhaps the security countermeasures that have been used in order to protect the AMPs from the Agents' actions can be used here as well. If the CAs could verify against specific safety properties [Necu96], [Necu98] the code of the DF Agent and the static Agents of the AMP, or if the CAs themselves could check for these properties [Gong97], then the CAs could sign the aforementioned Agents. Therefore, CAs could protect Agents from malicious AMPs, since the actions of an AMP would be predictable.

A dependent entity Agent might not want to let a CA-CSI Agent or an AMP know for which CSI the User-CSI Agent is looking for, unless the CA-CSI Agent does contain that CSI. A dependent entity Agent might want that, in order to protect his privacy (at least from the AMPs or CA-CSI Agents that do not have the CSI that the dependent entity is looking for). If that is achieved, then the risk of having a compromised AMP or CA-CSI Agent giving false CSI to User-CSI Agents could be reduced as well, because the AMP or CA-CSI Agent will not know what CSI the User Agent is looking for, unless the AMP contains such CSI. One way to achieve that [Riord98] would be to form the CSI query in the following way:

Query: H(N, CertID, CaCertID),

where H() is a hash function, and CertID is an identifier for the certificate the dependent entity is looking CSI for. It could be the public key of that certificate, the hash of that public key, the hash of the serial number of the certificate and the respective Distinguished Name [X.500] contained in the certificate, or any other construct that can identify uniquely a certificate issued by a specific CA. Furthermore, the query should include another certificate identifier, identifying the CA that issued the certificate, the dependent entity is looking CSI for. Finally, in order to avoid dictionary attacks from an AMP that would want to find out for which certificate is the dependent entity looking CSI for, a nonce (N) should be included as well. This nonce could be the nonce included in the Agent in order to protect the entities that receive the Agent from replay attacks.

The query is formed from the hash of the aforementioned information. The User-CSI Agent will be retrieving the respective certificate identifier values from the CA-CSI Agent and will be calculating the respective query values until he finds one that matches. If this is the case, the User-CSI Agent can retrieve CSI from the CA-CSI

Agent at that time. If the CA-CSI Agent or all the CA-CSI Agents that reside in the AMP, do not contain the CSI that the User-CSI is looking for, then the latter will communicate with another CA-CSI Agent, or will transport itself to another AMP. However, neither the CA-CSI Agents that the User-CSI Agent communicated with, nor the AMP the User-CSI Agent was residing in, will be able to know for which certificate was the Agent looking for.

Neither the CSI mechanisms we have reviewed in [Iliad00a], nor ADoCSI, take into consideration the reasons for the revocation of a certificate when performing certificate path validation. Furthermore, during a certificate path validation, none of the aforementioned mechanisms takes into consideration certificates that contain the same public key but belong to other certification paths, and they have been revoked. Obviously, the aforementioned certificates should belong to the same entity, but it could also be the case of two or more entities that had -on purpose- the same key pair, certified by different CAs.

Assume that certificates C_k and C'_k belong to the respective certificate chains C and C', depicted in the table below, and contain the same subject public key. Should C_k be revoked, that does not mean that C'_k will be revoked as well, as it belongs to another certificate chain. The fact that C_k has been revoked should affect C'_k in a different way, depending on the reason for the revocation of C_k [Fox98]. Currently, PKI does not support the revocation, or any other action upon a subject's certificate, if another certificate containing the same public key is revoked. Let us examine the various reasons for the revocation of a certificate. These reasons can be included, optionally, as an extension to the CRL [ISO9594], [Hous99]. This extension is called CRLReason [REFERENCE??].

$C_1, C_2, \ldots C_{k-1}, C_k, \ldots C_n,$
$C'_1, C'_2, \ldots C'_{k-1}, C'_k, \ldots C'_n$
Table 1: Certificate chains

C_k could be revoked because of the following reasons [ISO11770], [Fox98]:

1. The public key contained in Ck should not be trusted anymore because it has been compromised. The CRL entry should contain the reason 'keyCompromise'. In this case, C'k should be revoked as well, because it contains the same, compromised public key.

2. The binding between the public key contained in Ck and the subject name contained in Ck should no longer be trusted, because the affiliation of the certificate holder has changed, there is a new certificate that supersedes Ck or because the certificate holder will not be using anymore Ck. The respective reason codes that should be used in the CRL entry are: 'affiliationChanged', 'superseded' and 'CessationOfOperation'. If C'k contains information that bind the person denoted by the subject name in C'k to the same affiliation as is the case with Ck then C'k should be revoked as well. If Ck was revoked because it was superseded then C'k might not need to be revoked. Finally, if Ck was revoked because the certificate holder will not be using that certificate anymore, then C'k should be revoked or should not be revoked depending on the reason why the certificate holder will not be using C'k anymore.

3. Ck 1 (certificate holder k-1 is the issuer of certificate Ck) is no longer a position to vouch for the validity of Ck. This would be due to a compromise of the key contained in Ck 1. The reason code that should be used in the CRL entry is: 'cACompromise'. In this case, C'k should not be revoked, because C'k-1 (and every certificate higher in the hierarchy of C') is not affected because of the compromise of Ck-1.

We have demonstrated that there are cases when the revocation of a certificate belonging to one certificate chain may provoke the revocation of a certificate that belongs to another certificate chain but bears the same public key. However, the current, supported certificate path validation mechanisms and CSI location mechanisms do not support such actions. Even if such actions were supported buy the aforementioned mechanisms, there would still be questions to be answered regarding the revocation of one certificate on another certificate that contains the same public key. One of the issues that would have to be tackled with, is what would happen in case a certificate needs to be revoked for a reason that is not included in the list of reasons proposed by [ISO9594], [Hous99]. It could be the case that a CA (the holder of C_{k-1}) has realised that an entity (the holder of C_k) has been misidentified, either because of a procedural registration error or because the entity has provided false identification [ISO11770] to the

CA during the registration phase. In this case, none of the existing, proposed reasons to be included in the CRL entry do not concern the real reason for the revocation. Another issue that would have to be tackled with, is the actions to be taken with respect to C'_k in case C_k appears in a CRL with the value certificateHold in the entry extension reasonCode. A certificate could be suspended (certificateHold) for many, different reasons and it would not be clear in this case what actions should be taken either by the entities that own the certificates in the certification chain C' or by the dependent entities that wish to use certificates (e.g. C'_k, C_k) that belong to these chains.

ADoCSI could possibly provide the means to deal with the aforementioned problems, since the Agents involved in ADoCSI do have access to a common pool of CSI resources (the set of all AMPs and CA-CSI Agents). User-CSI Agents could seek the necessary information in this pool in order to verify the validity of a certificate, even if the same public key is contained in more than one certification paths. In addition to that, the dependent entity could customise the User-CSI Agents (through a graphical user interface provided by the Interface Agent) in order to let them know the preferences of the dependent entity regarding the revocation of certificates, depending on the reasons other -related- certificates have been revoked. Clearly, this subject has to be studied further.

5.Conclusions

We presented the prototype of an alternative mechanism for disseminating certificate status information (ADoCSI), which presents some enhancements, compared to the other mechanisms that handle CSI dissemination [Iliad00a], [Iliad00b]. One of the most important enhancements is that this mechanism meets the *transparency* criterion [Iliad00b]. Entities using PKI are not always knowledgeable, motivated and perceptive towards security procedures. ADoCSI could be of use, because it relieves these entities from the burden of carrying out security related tasks in order to locate and retrieve Certificate Status Information. In addition to that, new features can be implemented in the operation of this CSI dissemination mechanism in a transparent way. New features can be developed and integrated in the mechanism (e.g. in the CA-CSI Agents and the User-CSI Agents), that possibly serve the needs of the dependent entities in a better way. However, the dependent entities themselves need not know about the mechanics of these new features.

ADoCSI presents other advantages as well. The network traffic that results from CSI queries can be reduced, since the queries take place on a local level, inside the AMPs. Furthermore, ADoCSI is resilient to unreliable networks, because it can operate both in an online and offline manner.

Most of the existing CSI dissemination mechanisms restrict the Certification Authorities in the way they must operate. CAs cannot re partition the CSI space, or transport their CSI repositories elsewhere, because dependent entities will not be able to track those changes easily. ADoCSI lifts that restriction. CAs can re partition their CSI space, transport the CSI repositories and also maintain CSI in a number of different formats. Such actions would result in confusion of the dependent entities and would create problems in the certification process, if a CSI mechanism out of those presented in [Iliad00a] was used. On the contrary, if ADoCSI is used as a CSI dissemination mechanism, such an action on behalf of the CAs would result in even more efficient CSI dissemination.

ADoCSI provides for adjustable CSI. The dependent entities can retrieve CSI that meets their own requirements, without understanding the mechanics of CSI dissemination process and without the dependent entities having to act themselves or contribute in any other way to the aforementioned process. The only other mechanism offering fine grained adjustability in CSI location and retrieval is the Freshness constrained Revocation [REFERENCE??], but at the cost of loosing economy and scalability.

ADoCSI is a more efficient CSI dissemination mechanism compared to the ones we have presented in [Iliad00a]. Although the set of features it supports is larger than the respective sets of other mechanisms [Iliad00b], it is still an economic and scalable mechanism. Furthermore, it provides for timely dissemination of fresh CSI. Most of the other mechanisms that offer fresh CSI in a timely manner are either not economic or not scalable.

A disadvantage of ADoCSI is that it requires that the Trusted Computing Base (TCB) [Goll99] of the dependent entities and CAs grows in size. The User-CSI Agent and the Interface Agent are part of the TCB of the dependent entity. These Agents are locating, retrieving and possibly validating CSI on behalf of the dependent entity. CA-CSI Agents also form part of the new TCB of CAs. CA-CSI Agents are the ones that disseminate the available CSI, not the CAs, anymore. Taking into consideration that security issues related with the use of Agents still exist, such a growth of the Trusted Computing Bases requires that these problems have to be solved, in an economic way, before using Agents in the CSI process.

Generation and dissemination of Certificate Status Information has to be studied further. There are still problems that need to be solved, regarding CSI mechanisms. Solution to these problems could require the sharing of CSI among a wide group of CAs and the processing of that CSI before communicating it to the dependent entities. We believe that Agent based CSI mechanisms could possibly provide the means to deal with the

aforementioned problems since the Agents involved in such mechanisms have access to a common pool of CSI resources, the set of all collaborating AMPs and the respective, resident CA-CSI Agents.

A final note on ADoCSI is that it assumes there will be a sufficient number of collaborating AMPs and that each AMP contains a sufficient number of CA-CSI Agents. Furthermore, ADoCSI assumes that all Agents involved in the scheme are using the same Agent Communication Language. Although these assumptions do not pose any problem at all, on a technical level, there might be other obstacles in implementing such a CSI scheme. There might be commercial CAs that do will refuse to have their CA-CSI Agents in the same AMP as another CA, or in the AMP developed by an organisation that is somehow related with another CA. Furthermore, if a CA has a large user base (entities that have been issued certificates from that CA) that CA may refuse to serve CSI requests from User-CSI Agents developed by other CAs. In general, real life implementation of ADoCSI requires that CAs will develop multi level collaboration, in order to deliver CSI.

Taking into consideration the above, a degenerate version of ADoCSI might prove to be the most economic and efficient one. We refer to the case where each CA is operating its own AMP, hosting only its own CA CSI Agent, while the Directory Facilitator at that AMP knows the location of the AMPs of all other CAs. This is a very close analogy to the private network of the banking system. A dependent entity would have to send its User CSI Agent to an AMP it already knows the location of (Acquirer AMP/CA), and the latter would refer it to the AMP that contains the appropriate CSI (Issuer AMP/CA).

We have pointed out the need for transparency of PKI security services and in specific the need for transparency in the location and validation of CSI. We presented the prototype of an alternative mechanism for disseminating CSI that meets this need. There is clearly much work to be done yet, as pointed out at least in section 4. However, we consider we brought forward some new requirements regarding CSI mechanisms and sketched a prototype framework for meeting these requirements.

6.References

[Aglets] IBM Aglet Workbench, available at http://www.trl.ibm.co.jp/aglets

[Chess95] Chess D., Grosof B., Harison C., Levine D., Parris C., , IBM Research Division, T.J. Watson Research Center, New York, & Tsudik G., IBM Zurich Research Laboratory, Itinerant Agents for Mobile Computing, Research Report 20010, 1995

[FIPA98a] Agent Management, Part 1, Foundation for Intelligent Physical Agents Specification, Geneva, October 1998

[FIPA98b] Agent Communication Language Version 2.0, Part 2, Foundation for Intelligent Physical Agents Specification, Geneva, October 1998

[FIPA98c] Agent Security Management Version 1.0, Part 10, Foundation for Intelligent Physical Agents Specification, Geneva, October 1998

[Ford94] Ford W.,Computer Communications Security, Prentice-Hall, 1994

[Fox98] Fox B., LaMacchia B., Certificate Revocation: Mechanics and Meaning, In Proceedings of Financial Cryptography 98, LNCS 1465, New York, Springer Verlag

[Frost96] Frost H. R., Cutkosky M. R., Design for Manufacturability via Agent Interaction, In Proceedings of the ASME Computers in Engineering Conference, Irvine, CA, August 1996

[Gene92] Genesereth M. R., An Agent-Based Approach to Software Interoperability, In Proceedings of the DARPA Software Technology Conference, 1992

[Gene92b] Genesereth M., Fikes R., et al., Knowledge Interchange Format, version 3.0, reference manual, Technical Report, Computer Science Department, Stanford University, 1992

[Gene94] Genesereth M. R., Ketchpel S. P., Software Agents, In Communications of the ACM 37 (7), 1994

[Goll99] Gollman D., Computer Security, John Wiley & Sons, 1999

[Gong97] Gong L., Mueller M., Prafullchandra H., Schemers R., Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2, In Proceedings of the USENIX Symposium on Internet Technologies, 1997

[Gong98] Gong L., Signing, Sealing and Guarding Java Objects, Lecture Notes in Computer Science (LNCS), Vol.1419, Springer-Verlag, June 1998

[Gorri98] Gorrieri R., Marchetti R., Applet Watch-Dog: A Monitor Controlling the Execution of Java Applets, Proceedings of the IFIP SEC'98, published by Austrian Computer Society, 1998

[Gritz97] Gritzalis, S., Spinellis, D. Addressing Threats and Security Issues in World Wide Web Technology, In Proceedings of the 3rd IFIP International Conference on Communications and Multimedia Security, Chapman & Hall, 1997

[Gritz98] Gritzalis S., Iliadis J., Addressing security issues in programming languages for mobile code, In Proceedings of the DEXA '98 9th Workshop of Database and Expert Systems Applications, IEEE Computer Society Press, August 1998

[He98] He Q., Sycara P., Finin T. W., Personal Security Agent: KQML Based PKI, In Proceedings of the ACM Conference on Autonomous Agents (Agents'98), May 1998

[Hous99] Housley R., Ford W., Polk W., Solo D., Internet X.509 Public Key Infrastructure Certificate and CRL Profile, , IETF Network Working Group, Request for Comments 2459 (Category: Standards Track), January 1999, available at http://www.ietf.org/rfc/rfc2459.txt

[Iliad00a] Ioannis S. Iliadis, Diomidis Spinellis, Sokratis Katsikas and Bart Preneel, A Taxonomy of Certificate Status Information Mechanisms, In Information Security Solutions Europe ISSE 2000, Barcelona, Spain, September 2000. European Forum for Electronic Business.

[Iliad00b] J. Iliadis, D. Spinellis, S. Katsikas, D. Gritzalis, B. Preneel., Evaluating Certificate Status Information Mechanisms. In Proceedinds of the 7th ACM Conference on Computer and Communication Security: CCS '2000, pages 1-8. ACM Press, November 2000.

[Iliad99] Ioannis S. Iliadis, On the Dissemination of Certificate Status Information, MSc Dissertation, Dept. of Mathematics, Royal Holloway College, University of London

[ISO11770] ISO/IEC JTC 1/SC 27/WG 1 11770-1.2, Information Technology - Security techniques - Key management - Part 1: Framework, 1996

[ISO9594] ISO/IEC 9594-8 (1994), Open Systems Interconnection - The Directory: Authentication Framework. The 1994 edition of this document has been amended by the Draft Amendments [Dram96] and a Technical Corrigendum [Cor95]

[Jacl97] Jacl and Tcl Blend, (1997) available at http://sunscript.sun.com/java/

[KQML] ARPA Knowledge Sharing Initiative, External Interfaces Working Group, Specification of the KQML agent-communication language, working paper, July 1993

[Lamb97] Lambrou Y., Finin T., A Proposal for a new KQML Specification, Technical Report CS-97-03, University of Maryland, Baltimore, US, 1997

[Necu96] Necula G.C., Lee P., Safe Kernel extensions without run-time checking, Proceedings of the Second Symposium on Operating Systems Design and Implementation, 1996

[Necu98] Necula G. C., Compiling with Proofs, PhD Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, September 1998

[Nwana96] Nwana H. S., Software Agents: An Overview, In Knowledge Engineering Review, Cambridge University Press, 1996

[Odyssey] General Magic Odyssey, available at http://www.genmagic.com/agents

[Petrie97] Petrie C. J., What's an agent… and what's so intelligent about it?, IEEE Internet Computing Vol. 1, Number 4, July-August 1997

[Riord98] Riordan J, Schneier B., Environmental Key Generation towards Clueless Agents, In Proceedings of Mobile Agents and Security, LNCS, Springer-Verlag, 1998

[Rivest98] Rivest R., Can We Eliminate Revocation Lists?, In Proceedings of Financial Cryptography 1998, available at http://theory.lcs.mit.edu/~rivest/revocation.ps

[Schn96] Schneier B., Applied Cryptography, John Wiley & Sons, 1996

[Spruit96] Spruit M. E. M., Looijen M., IT security in Dutch practice, Computers & Security, 15-2, 157, 1996

[Spruit98] Spruit M. E. M., Competing against human failing, In Proceedings of the XV IFIP World Computer Congress: 14th IFIPSEC '98 International Information Security Conference, September 1998

[Thiru95] Thirunavukkarasu C., Finin T., Mayfield J., Secret Agents - A Security Architecture for the KQML Agent Communication Language, In Proceedings of CIKM'95 Intelligent Information Agents Workshop, December 1995

[Wil98] Wilhelm U. G., Staamann S., On the Problem of Trust in Mobile Agent Systems, In Symposium on Network and Distributed System Security, Internet Society, March 1998

[Wil99] A Technical Approach to Privacy Based on Mobile Agents Protected by Tamper-resistant Hardware, PhD Thesis, Wilhelm U. G., Department of Informatics, EPFL, Lausanne, 1999

[X.500] CCITT Recommendations X.500-X.521, Data Communication Networks Directory, CCITT, November 1988

[Zhang97] X.N.Zhang, Secure Code Distribution, IEEE Computer, June 1997