A Certificate Revocation Scheme for a Large-Scale Highly Replicated Distributed System

Bogdan C. Popescu
Vrije Universiteit
Amsterdam, The Netherlands
bpopescu@cs.vu.nl

Bruno Crispo
Vrije Universiteit
Amsterdam, The Netherlands
crispo@cs.vu.nl

Andrew S. Tanenbaum
Vrije Universiteit
Amsterdam, The Netherlands
ast@cs.vu.nl

Abstract

A common way to protect objects in distributed systems is to issue authorization certificates to users, which they present to gain access. In some situations a way is needed to revoke existing certificates. Current methods, such as having a master revocation list, have been designed to work efficiently with identity certificates, and do not take into account the delegation of certificate-issuing rights required when implementing complex administrative hierarchies for large distributed applications. In this paper we present a novel mechanism for revoking authorization certificates based on clustering users and servers, and present arguments showing that it is more efficient than other methods. We also discuss a way for probabilistically auditing the use of the revocation mechanism proposed to reduce the chances of any component behaving maliciously.

1 Introduction

A worldwide distributed system may contain various protected objects to which users may acquire access. These may include bank accounts, music files, and databases. In a small system, it may suffice to have a central database that keeps track of which users have access to which objects and in what way. However, in a large distributed system, centralized access control is impractical. An alternative scheme is to issue each user a cryptographically sealed authorization certificate storing the user's permissions with respect to a given object. To access that object, the user authenticates first and then presents the certificate, which is inspected for the necessary permissions before executing the request. Normally, such certificates include an expiration date after which they are invalid.

However, situations arise when the object owner needs to revoke access quickly and without warning, for example, if the certificate is stolen, or the user is busy obtaining information from an object and using it in violation of the law (e.g., posting copyrighted information on the Internet). Revoking authorization certificates is much more difficult than simply deleting the user from a central access control list, and this is made even more difficult when considering the possibility of delegation of access rights. In such a situation, it is not enough to only verify that a given certificate has not been revoked, we also need to ensure that the delegation chain that connects the owner of the object to the issuer of the certificate is still valid. In this paper we describe a scheme for revoking authorization certificates efficiently in a very large distributed system when taking into account the possible delegation of certificate-issuing rights.

To make this design concrete, we have applied it to the Globe distributed system [20], which we are developing. Globe has been designed to support a billion users and a trillion objects, many of which are expected to be highly replicated for performance and fault tolerance reasons.

The rest of the paper is organized as follows: section 2 gives a quick overview of the Globe system with an emphasis on the security architecture. In section 3 we introduce the framework model for describing our revocation mechanism. In section 4 we present our scheme, the trade-offs we make in order to guarantee performance, and the mechanisms we use to ensure that such trade-offs will not lead to less security. In section 5 we look at existing revocation mechanisms and compare them to our scheme, and in section 6 we conclude.

2 The Globe System

Globe is a distributed system based on replicated shared objects. While the idea of encapsulating functionality into objects is not new (systems like Corba [1], Legion [9] or DCOM [7] rely on this paradigm), what makes Globe unique is that objects not only can be used by a large numbers of users on different machines through remote procedure calls, but also can be physically replicated on many

hosts at the same time to improve performance.

The central construct in the Globe architecture is the distributed shared object (DSO). As shown in Figure 1 a DSO is built from a number of **replicas** that reside in a single address space and communicate with replicas in other address spaces. All the replicas that are part of a DSO work together to implement the functionality of that DSO. A replica consists of the code for the application (the code that implements the functionality of the DSO that replica is part of), the part of the DSO state the replica stores, and the replication mechanism.

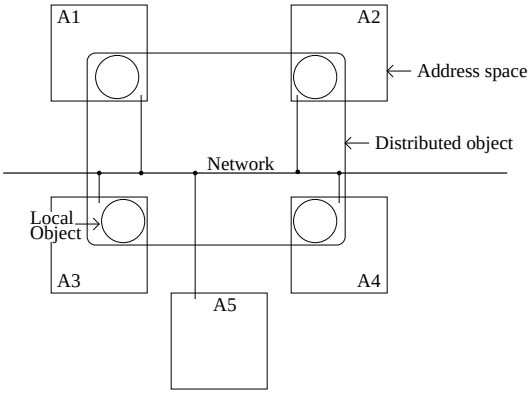

Figure 1. A Globe DSO replicated across four address spaces

In order to use a Globe DSO, a client has to find a replica part of that DSO, connect to it and then invoke one of the methods in the DSO's public interface. If the method invocation modifies the internal state of the replica, the replica will propagate the changes to other replicas according to the DSO's replication algorithm.

The Globe security architecture [17] is based on public key cryptography. Each secure Globe object is required to have a public/private key pair; the **owner** of a DSO is the entity that has access to the object's private key and is in charge with setting the security policy for that DSO.

Globe users are also required to have public/private key pairs used for authentication and access control which is implemented through **user authorization certificates**. The structure of such a certificate is shown in Figure 2: it contains the user public key, some authorization information (for example which of the DSO's methods that user is allowed to invoke), the exact time when the certificate was created, the maximum validity time interval, and a **validation frequency** time interval which specifies the maximum time a validated certificate can be used before it has to be re-validated. As one can see, authorization certificates in Globe are very similar to SPKI/SDSI certificates [8]. The authorization information in the certificate can be expressed in different policy languages [4, 5]; however the exact syntax to be used is outside the scope of this paper.

User Authorization Certificate
User's Public Key
Authorization Information
Time of Issuing
Maximum Validity Time
Validation Frequency
Issuer's Signature

Figure 2. An authorization certificate in Globe

In the simplest case, user authorization certificates are digitally signed by the DSO's owner with the object's private key. However, in order to support more flexible security policies we allow the delegation of administrative rights, which is done through **administrative certificates**. The structure of a administrative certificate is similar to the one of the user authorization certificate, except that the authorization information also contains delegation rules. For example, a administrative certificate may specify that an administrative entity is allowed to issue user authorization certificates that only permit the invocation of certain methods of the DSO and also allowed further delegate this right to other administrative entities. Thus, in the general case, a user has a **user authorization certificate chain** which starts with an administrative certificate signed by the DSO's owner, and ends with the user authorization certificate containing the user's public key.

As we can see, each Globe DSO de-facto implements its own PKI (public key infrastructure) with the DSO public key as the global trust root. As opposed to existing PKIs which are mainly used for authentication, our per-DSO PKIs are used for authorization: before sending a method invocation request to a DSO replica, a Globe user is required to present its user authorization certificate chain and authenticate to the replica. The replica first ensures the certificate chain is a valid one and then consults the authorization information in the last certificate in the chain and determines whether the user is allowed to invoke the method.

The access control mechanism just outlined is well suited for a wide-area network environment because it allows users to carry tamper-proof copies of their access rights, and thus does not require a central access control list (ACL). However, there is a price to be paid for this, namely access rights revocation is now more difficult to carry out than in the ACL case; each administrative entity will have to participate in this revocation process, by periodically disseminating revocation status information regarding each of the user or administrative certificates it has issued. This re-

vocation information is then used by DSO replicas when verifying the validity of the certificate chains presented by users when invoking methods: the replica has to check each certificate in the chain against the revocation information published by the administrative entity that has issued the certificate.

It is clear that a DSO with a complex administrative hierarchy puts a significant extra workload on its replicas, which have to frequently validate long authorization chains. To alleviate this problem, we have designed a revocation infrastructure and mechanisms that makes use of the natural clustering of users and replicas for distributed Globe objects and significantly decreases the amount of validation work a replica has to perform.

So far, we have presented the motivation for our work. The next section will introduce a more formal model of our operational infrastructure which will then be used when presenting the proposed revocation mechanisms.

3 Operational Model

In this paper we consider an environment where digital certificates are used to implement access control policies for applications employing highly replicated servers connected by a wide area network. This environment fits the needs of Globe, but is also applicable to other types of applications, such as distributed databases or content delivery systems, which make use of replication to improve performance. As we have seen in the previous section, certificate revocation in such a setting introduces problems not present when revoking identity certificates. To handle these problems we propose a new revocation infrastructure where novel revocation mechanisms can be employed. This infrastructure is formalized in Figure (3) which shows the various entities in our model and the relationships among them:

- A number of certificate producers (CPs), each of them identified through a public/private key pair, which are organized in a hierarchical PKI. This PKI corresponds to the administrative hierarchy for a distributed application. The CPs are responsible with issuing authorization certificates for users and administrative certificates for their subordinate administrative entities. Each CP is also responsible with timely publishing revocation information regarding the certificates it has issued.

- The Users - non-trusted entities which use certificates to gain access to some distributed application through the service points provided by that application. In our model, users retrieve the certificates themselves and present them to the service points.

- The Directories - these are much less trusted entities which serve as repositories for the revocation information. In our model we also require each directory to have a public key (certified by each CP that makes use of the directory) and to be able to digitally sign some of the information it sends to interested parties.

- The Servers - the 'interested parties'. They provide services to users under the access control policy set by the CPs. They check service requests against the access control policies stored in user certificates, and verify the validity of these certificates by getting revocation information from the directories. Each server is assigned to a directory; the way this is done is application specific and will depend on the replication strategy used; however, this is outside the scope of the paper. More important though is that a given server will only request revocation information from its assigned directory. All the servers assigned to a directory form the **dependent server set** for that directory.

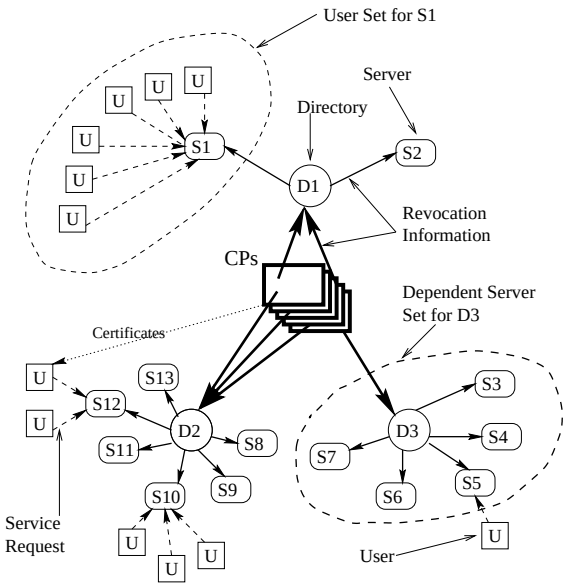

Figure 3. The Framework Model

As we can see, the model just introduced attempts to formalize what we described in Section 2 - how certificates are issued, used and revoked, the trust we place in each entity in the system, and how each of these entities acts in order to ensure timely revocation. Crucial in our revocation infrastructure are the on-line directories which, as we will see in the next section, play an active role in the revocation mechanism by taking away some of the verification burden from the servers. This comes in contrast to the vast majority of current revocation schemes which employ such directories

in a much more passive role - as simple (un-trusted) repositories for revocation information.

4 Proposed Revocation Mechanism

The standard mechanism for publishing revocation information is the **certificate revocation list (CRL)** [19]. A CRL is simply a list of certificates that have been revoked, time-stamped and signed by the revocation authority (in our case these are the CPs). The problem with this mechanism (we call it plain CRLs) is that verifying a certificate has not been revoked requires downloading the entire revocation list, which increases validation latency. Recently, a new class of "reduced-data structures" revocation mechanisms have been designed [16, 14, 11], and these mechanisms have the advantage that validating a single certificate requires retrieving only a small fraction of the revocation data.

The revocation mechanisms we propose in this paper combines some of this existing schemes and employs a new technique - probabilistic auditing - which allows us to take away some of verification burden from the servers and place it on the on-line directories, while at the same time limiting the amount of trust we need to put in these directories.

4.1 The General Idea

We first make the observation that for a highly replicated distributed application, the servers will cluster around areas with high density of users. After all, this is the very reason for replicating servers - to improve the performance on handling user requests. Where do we need many servers? In areas where we have lots of users - hence clustering. We claim that users sending requests to servers in one cluster are unlikely to also send requests to servers in a different cluster. After all, most users do not move great distances frequently, and taking your laptop from home to work is not likely to change the server you use.

Now, let us see the problem from the servers' point of view: most of the requests will come from users within a short network distance from them (a server located in California will get most of its requests from users on the West Coast, and very few requests from Holland, for example). We call this group of users that are in close network distance from the server and are very likely to use it, the **user set** for that server. Therefore, if we could validate the users in the user set fast, this would improve overall performance, even if validating users outside the set will be less efficient.

Let us now take the directories into account. The **user set** for a directory is the union of the user sets for all the servers that depend on it for revocation information. For the reasons we explained earlier, we expect the user set for a directory to be fairly stable over time.

CPs can then periodically push their CRLs to directories. For now, we require these CRLs to support on-line validation using a reduced data structure (proving that a certificate is valid or revoked does not require the entire list). This can be done using schemes like the ones described in [16], [14] or [11]. However, in section 4.3 we will describe a variant of our scheme that only uses plain CRLs.

Each directory then issues **local certificates** for the users in its user set. A local certificate has the same structure as a user authorization certificate, and will contain the exact same authorization information, the only difference being that it is signed by the directory instead of a CP. Once it issues a local certificate, the directory is required to constantly monitor the validity of the corresponding user authorization certificate, and promptly inform its dependent servers when it is revoked. In order to do this, the directory stores the entire authorization certificate chain for the certificates it monitors, and constantly retrieves the global CRLs corresponding to the administrative certificates in this chain. If any of the certificates in this chain is revoked, the directory will revoke the local certificate by adding it to a local CRL. In order to handle timing constraints properly, the directory is required to issue local CRLs with a frequency equal to the smallest validation frequency of any of the certificates in the certificate chains it monitors.

Whenever a user not in the user set for the directory invokes a method on one of the servers in the directory's server set, that server forwards the global user authorization certificate chain to the directory. The directory retrieves (if necessary) and checks all the global CRLs corresponding to the certificates in the chain, and if they are all valid issues a local certificate for that user. The local certificate is then returned to the server, which returns it to the user together with the result of the method invocation. As we can see, this step (registering a new user to a directory) is rather expensive because a number of CRLs need to be retrieved and verified, but this is something the server would have to do anyway if it wouldn't be done by the directory.

This rather high initial overhead is well amortized if the user stays in the user set of that directory for a longer period of time. For a user who already has a local certificate, validation is much faster: the user only has to show its local certificate to a server, which only needs to check one CRL - the local one issued by the directory - in order to ensure the certificate is valid.

However, the price we have to pay for reducing the validation workload on the servers is the need to put some trust in the on-line directories. In the scheme so far described, servers in a given directory domain do not deal with the certificates and CRLs issued by the CPs, but with the local certificates issued by their directory. Things work fine as long as the authorization information in the local certificate exactly matches the one in the global user certificate,

and directories promptly report when global certificates are revoked, but there is always the threat of local directories abusing their authority. In the following section we will describe a technique - probabilistic auditing - which can be used to effectively counter the threat of local directories acting maliciously.

4.2 Probabilistic Auditing

In any revocation scheme it is assumed the directories are much less trusted than the CPs publishing the CRLs. In the scheme we described in the previous section, directories are allowed to sign local certificates and local CRLs, and this gives them the potential power to decide which users are allowed to access the servers in their domain. By abusing this power, they can perform the following types of malicious acts:

- Allow unauthorized users to use the directory's servers - this can be done by giving those users a local certificate even when they do not have a global one, or giving them more permissions in the local certificate than in the global one.
- Allow revoked users to use the directory's servers - by not adding these users to the local CRL.
- Deny legitimate users the access to the directory's servers - by putting these users on the local CRL, even if they are not on the global one, or by simply refusing to issue them a local certificate.

In order to prevent a directory from performing such malicious acts, its dependent servers perform **probabilistic auditing** of the directory's actions. Whatever the directory does (signing a local certificate, publishing a local CRL) is subject to verification by some of its servers. Such verification occurs only with a certain probability - small enough, so it is not a major performance hit, but large enough so that it is a powerful deterrent against directory misbehavior. The probability of an audit can be fine-tuned on a per-directory basis. There are several types of audit actions that can be performed by a depending server on its directory:

First, a server can compare the local certificate against the global authorization certificate chain one to make sure they match. This is the simplest form of verification, it only involves extra public key signature verification operations, so the probability of such auditing can be set quite high (most likely to 1 - always verify) without negatively affecting performance. This type of audit protects against the most serious threat in our scheme - the local directory injecting unauthorized users in the local environment.

Second, a server can ask its directory for a proof that a local certificate (that has been verified to be identical to the global one) is indeed still valid. This is a more expensive operation since it involves one directory-to-server network round-trip time to send the audit request and get the proofs, plus the time needed by the server to validate each certificate in the global authorization chain. This type of audit prevents the directory from keeping revoked users in the local environment.

Finally, a server can request proof that a user that has been revoked in the local environment (either by putting her on the local CRL, or by the directory refusing to issue a local certificate for that user) has indeed been revoked by the CPs. This operation is again rather expensive, involving the same steps as the previous type of audit. This type of audit prevents the directory from denying access to legitimate users to the local servers.

From a performance point of view it is important to note that some of these audit actions do not need to be done during the time the server is handling user requests. For example, a server can have a background task that verifies the local CRL, by randomly selecting entries in that CRL and asking the directory to prove these entries have indeed been revoked. Another possibility is for servers to keep a log of users that have been served but only their local credentials have been verified. Whenever a server is idle, it can randomly select users from the log and request the directory to prove these users have indeed not been revoked.

By using such background checking techniques, we can set the audit probability to be quite high without degrading the server's response time to user requests. It is important to notice that background auditing does not prevent individual security violations (since the check will most likely happen after the violation has occurred). Instead, auditing works as a **deterrent**, and we claim that in the environment we described this can be efficient in making directory misbehavior very unlikely.

Another interesting fact is that our scheme allows servers to choose the amount of trust they want to put in the local directory: for less important transactions, they can immediately trust the local certificate, and rely on the probabilistic audit mechanism to catch any eventual misbehavior on the directory's part. For important transactions, the servers can do an on-line complete verification. Such a verification is always possible in our revocation scheme, and involves the following steps:

- request the user's global certificate chain and verify it
- ask the directory to prove the user has not been revoked

Finally, it is important to understand that in a wide-area environment, it is always possible to experience link failures that would prevent directories from receiving fresh revocation information. When a directory gets to the point where it is impossible to get a fresh CRL from the CP, it has two options: it can either stop answering validation requests, or it

can go into a "degraded" operational mode, when it explicitly warns its users that the responses provided are based on non-fresh data. Which one of these two options is more appropriate is dependent on the application scenario.

4.3 Discussion

When describing our revocation mechanism, we required the global CRLs generated by the CPs to support on-line verification using reduced data structures (as described in [16], [14] or [11]). However, these algorithms are rather complex, not that widely used and rely on heavy cryptographic computations which may put too much workload on the CPs and servers. The good news is that our revocation mechanism can be adapted to work even if CPs produce just plain simple CRLs.

This works as follows: local directories keep issuing local certificate and local CRLs as in the original scheme. The first type of audit (local certificates compared to the global ones) also works in the same way as in the original scheme. However, the last two types of audit need to be modified, since now verifying that a local certificate has indeed not been revoked requires a server retrieving the global CRLs corresponding to all the certificates in the global certificate chain, and if most of the servers end up retrieving global CRLs we give away all the performance benefits gained by only distributing local CRLs. We need to ensure the global CRLs will only be sent to a small number of servers. This can be done by setting a small probability that a server requests the global CRL after it receives a fresh local CRL. For example, if this probability is 0.05 and we have 100 servers depending on a directory, then, on average, only five of them will ask for the global CRLs. Thus, the majority of servers will only use the local CRL, while a few servers will have both the local and global CRLs. What happens then is that servers holding both local and global CRLs will check local user certificates against the local CRL, and their corresponding global chains against the global CRLs. A directory is caught misbehaving if a local certificate is not revoked in the local CRL, while its corresponding global chain is no longer valid, or if a local certificate is revoked while its global chain is still valid. The local certificate and local CRL are signed and time-stamped by the directory, so the vigilant server can simply forward them to the root CP as an undeniable proof that the directory has been corrupted. The CP can then take coercive action.

One thing we have not explained so far is what happens when a directory is caught 'red handed'. This is very much dependent on the relation between the directory and the CPs/servers for which it facilitates the distribution of revocation information. One option would be that the root CP, servers and directories are all part of the same administrative realm. In this case, directories are inherently trusted, except that because they are on line all the time they are more exposed to attacks. In this case, the probabilistic auditing can be used to determine when a directory starts acting malicious because it has been hacked, and once such a directory has been identified, administrative action can be take to fix the problem (for example, rebooting the machine, re-installing software on it, changing account passwords).

Yet another possibility is that the servers and directories are independent entities, all collaborating as parts of a peer-to-peer system. In this case, a reputation scheme can be employed, with the entities with a high reputation being elected to serve as directories. When directories are caught red-handed, their reputation will drop, and eventually they will be removed from the directory role.

5 Related Work

The simplest revocation mechanism is no revocation at all, simply use arbitrary short lived certificates. In this scheme users would have to contact the certificate producers very often to obtain new certificates. Each new certificate requires the issuing CP to generate a digital signature - an expensive public key operation. Also, since users ask for new certificates frequently, this means CPs need to be on-line all the time, which is a security hazard.

Besides this simplistic solution, revocation has been widely studied but almost all the previous work has been focussed on the efficient implementation of the revocation data structure, in order to reduce bandwidth and storage requirements at the client side [18, 13, 10, 6, 14, 11, 16]. All these mechanisms assume that the revocation information (in most cases a revocation list) is signed by the CP (or some other authorized party) and is stored on some un-secure on-line server. A different approach is taken by the designers of the On-Line Certificate Status Protocol (OCSP) [15] and the Data Certification Server Protocol [2]. These mechanisms require that before accepting any certificate, an application server engages in some on-line protocol with a trusted entity and verifies the certificate has not been revoked. The obvious drawback of this approach is that it needs a trusted on-line entity that does certificate validation. The other problem with on-line verification protocols is that they do not make use of the natural geographical clustering of validation requests.

One mechanism that makes use of this natural geographical clustering is Segmented CRL/CRL Distribution Points ([10, 6]). However, this is mostly dealing with identity certificates, and does not consider the problems posed by by long delegation chains that are specific to authorization certificates.

The only work we are aware of that explicitly considers the revocation of authorization certificates is [12]. However, that paper focuses mostly on a high-level discussion on dif-

ferent scenarios where authorization certificate revocation may be needed, and what are the operational requirements in each of these situations. In our paper, we focus more on system design, and we describe a revocation infrastructure that makes use of semi-trusted on-line entities.

6 Conclusion

In this paper we tackle the problem of authorization certificate revocation, in a wide-area network distributed environment. We explicitly take into account the possibility of authorization delegation; this leads to (potentially long) authorization certificate chains which, when validated, put an unacceptably high workload on servers. We are able to alleviate this problem by shifting some of this validation burden to a number of semi-trusted revocation directories. A novel technique - probabilistic auditing - is then used to ensure these directories cannot go malicious without being detected.

As for future work, we plan to implement the revocation mechanism presented in this paper in conjunction with a large, highly replicated, Globe application (the Globe Distribution Network [3], for example). This would allow us to verify the user clustering assumption, and identify refinements that could make our scheme more efficient.

References

[1] The Common Object Request Broker: Architecture and Specification. www.omg.org, Oct 2000. OMG Document formal/01-12-01.

[2] C. Adams and R. Zuccherato. Data Certification Server Protocols. Internet Draft (expired), June 1998.

[3] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, I. van der Wijk, M. van Steen, and A. Tanenbaum. The Globe Distribution Network. In *Proc. 2000 USENIX Ann. Conf. (FREENIX Track)*, June 2000.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust-Management System, Version 2. RFC 2704, September 1999.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of IEEE Conference on Privacy and Security*, 1996.

[6] D. Cooper. A Model of Certificate Revocation. In *15th Annual Computer Security Applications Conf.*, 1999.

[7] G. Eddon and H. Eddon. *Inside Distibuted COM.* Microsoft Press, Redmond, WA, 1998.

[8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, September 1999.

[9] A. Grimsaw and W. Wulf. Legion - a view from 50000 feet. In *Fifth IEEE Int'l Symp. on High Performance Distr. Computing*. IEEE Computer Society Press, Aug 1996.

[10] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure: Certificate and CRL Profile. RFC 2459, http://www.ietf.org/rfc/rfc2459.txt, January 1999.

[11] P. Kocher. A Quick Introduction to Certificate Revocation Trees (CRTs). http://www.valicert.com/company/crt.html.

[12] Y. Kortesniemi. Validity Management in SPKI. In *Proc. of 1st Annual PKI Research Workshop*, 2002.

[13] P. McDaniel and A. Rubin. A response to "Can we eliminate certificate revocation lists?". In *Financial Cryptography*, 2000.

[14] S. Micali. Efficient Certificate Revocation. Technical report, MIT/LCS, 1996.

[15] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. IETF RFC 2560, June 1999.

[16] M. Naor and K. Nissim. Certificate Revocation and Certificate Update. In *7th USENIX Security Symp.*, pages 217–228, January 1998.

[17] B. Popescu, M. van Steen, and A. Tanenbaum. A Security Architecture for Object-Based Distributed Systems. In *Proc. 18th Annual Computer Security Applications Conference*, Dec 2002.

[18] R. Rivest. Can we eliminate certificate revocation lists? In *Financial Cryptography*, LNCS1465, pages 178–183, 1998.

[19] U.S. National Institute of Standards and Technology. A Public Key Infrastructure for U.S. Government Unclassified but Sensitive Applications. Federal Information Processing Standards Publication 180, 1993.

[20] M. van Steen, P. Homburg, and A. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, pages 70–78, January-March 1999.