Using CRL Push Delivery for Efficient Certificate Revocation Information Distribution in Grids
CESNET Technical Report X/2007

Daniel Kouřil, Luděk Matyska, Michal Procházka
CESNET z.s.p.o., Zikova 4, 160 00 Praha 6, and Czech Republic
{*kouril,ludek,michalp*}*@ics.muni.cz*

December 6, 2007

Abstract

Checking revocation information is necessary to prevent from using digital certificates whose contents become invalid. In current system either periodical retrieval of Certificate Revocation Lists (CRLs) or the Online Certificate Status Protocol (OCSP) are the most common mechanisms to access revocation information issued by the certification authorities. As both these approaches pose problems we propose a new method based on a Push model, which is based on the Grid Monitoring Architecture. Using this approach we guarantee the revocation information is distributed in a robust and timely manner. We also describe a pilot implementation of the service based on the proposed design.

1 Introduction

Reliable use of the PKI is based on credible certificates and trust in their authenticity. A revocation mechanism has been developed to inform relaying parties about stolen, invalid, or compromised certificates known to the issuing Certification authorities (CA). Information about canceled certificates is published by each CA or specialized service using a specific information channel as the only way how the information can reach the relying parties (end users and services) that rely on credibility of the certificates. Such an approach obviously requires to check the latest revocation information as part of the standard verification of the certificate because accepting a revoked certificate could potentially lead to unauthorized access or information leakage. CAs usually pay attention to proper management of revocation information and make sure revocation information is updated promptly and made available to the relying parties.

The most usual mechanism for distribution of revocation information nowadays is using a *certificate revocation list* (CRL) that the CA publishes regularly for the relying parties to download. While the certificate validity check is made local (using the downloaded copy of the revocation list), there may be too long delay between the time when a new CRL is published and when it is retrieved by all relying parties. Due to potentially large size of the CRLs they often cannot be downloaded more faster so the window cannot be reduced. This limitation was dealt with the *Online Certificate Status Protocol* (OCSP), which allows to make a direct connection to the CA during certificate verification to check if the certificate is still valid. However, the OCSP also suffers from few drawbacks, notably the dependency on a network connectivity to an OCSP responder, which makes the certificate verification last longer and be more fragile. Also, due to the high computational cost the OCSP server may become a bottleneck if too many requests are issued simultaneously.

There are also other solutions addressing the revocation problem. Despite some of them are very effective, support for such mechanisms is not implemented in the major tools used to build a

production environment, e.g. Grid, so architects of such systems usually must choose one of these two methods.

Both the OCSP and standard way of CRL retrieval represent a *Pull* model of revocation information distribution, where the relying parties are responsible for checking if it is valid or if a new piece of revocation information has been issued. In this report we present a new way of distributing CRLs, based on a push model, which ensures that they are delivered to the end systems almost immediately after their publishing by the CA. The relying parties thus do not need to contact the CA directly to consult the revocation information since the locally stored CRL is current.

The rest of the report is organized as follows. In the following section we provide an overview of current technologies used to check revocation information. Then we describe specifics of the Grid environment including comparison of OCSP and CRL performance. We then describe requirement for a new service for CRL distribution based on the Push model in Section 4 and implementation of the service in Section 5. We end the report with evaluation of the system and conclusions.

2 Related work

Effective handling of revocation information is crucial and has been subject of research activities yielding in several results. In this section we present approaches that are linked to our work. For more complete surveys summarizing current revocation techniques we refer to [2, 6, 23, 19]. We follow the classification used in [6] and categorize the methods into three classes according to the character of certificate status information (CSI) they provide.

2.1 Mechanisms providing negative CSI

These methods work as black-lists, specifying the certificates that have been revoked.

2.1.1 Certificate Revocation Lists

The most widely used method is to use the Certificate Revocation List (CRL), which is issued by the CA to mark previously signed certificates as invalid. Upon issuing the CRLs are made available from a public repository so that each relying party can download the current CRL on its system. Since the CRL is signed with the CA private key it is possible to distribute the CRLs among multiple repositories and no trusted service is needed to help distribute the CSI. The CRLs are assigned a validity timestamp and also contain information when a new CRL will be published, so the relaying party can always verify it uses the current CRL. Being standardized by the X.509 standard [22] and profiled by IETF [5] CRLs are supported by virtually all PKI systems. There are several approaches how the CRLs are actually retrieved, varying from regular downloading just before their expiration, through a periodic polling of the CRLs distribution points, to retrieval occurring just during the certificate verification (CRLs may or may not be put in a local cache in the last case).

Regardless the exact approach to CRL retrieval, large size of the CRLs yielding a low scalability and untimeliness are the key drawbacks of the CRL use. CRLs can be very long causing increased bandwidth requirements. CRLs also leave a long *window of vulnerability*, especially when they are updated only before expiration. Since CRLs are usually valid for a week or two, a certificate revoked just after issuing a fresh CRL can be accepted by the relying party for several days. Some CAs issue new CRLs upon each revocation however the relaying party must mention and download the new CRL to take the new CSI into consideration. Also such unscheduled CRL publishing limit the possibility of having the CRLs distributed among other untrusted repositories, since the relying party cannot verify the CRL is the latest one as published by the CA.

In order to mitigate the shortcoming of the standard CRL, several approaches have been proposed that are shortly described here.

2.1.2 Delta CRL

Delta CRL is an attempt to address the large size of the CRLs. A Delta CRL is a special CRL containing a list certificates that have been revoked since the last complete CRL (*base CRL*) was issued. A Delta CRL is supposed to be much smaller than the corresponding full CRL and thus it can be retrieved more often by the relying parties with less bandwidth consumption. The concept of Delta CRL is part of the X.509 standard [22].

2.1.3 Over-Issued CRL

This model of CRL management was introduced inčitecooper99 to decrease the peak request rate that can be observed when a CRL has expired and new one has just been published. The mechanism suggests to issue multiple CRLs that overlap in time so that more valid CRLs may exist at a given moment. Using multiple CRLs makes it possible to distribute the requests among the community thus limiting the peaks during CRL downloads.

2.1.4 CRL Distribution Points

The CRL Distribution Points mechanism also aims at reduction of the CRL size. The schema is also standardized by the X.509 standard [22] and allows to segment the CA certificate space into multiple smaller sets, each one having its own CRL available from its Distribution Point. When a suitable division rule is chosen and certificates are grouped together according to similar use, it is likely that a verifier will only need a few of the segments to validate all the certificates she communicates with.

Notice that the CRL Distribution Points schema does not help to reduce the bandwidth in scenarios where all the CRLs are retrieved in advance and cached on the local system.

2.1.5 Redirect CRL

The CRL Distribution Points schema introduced above does not address the scenario when a CRL of one of the fragments starts growing faster then the other CRLs. The Redirect CRL concept [1] allows for dynamically repartitioning of the space which makes it possible to control size of the fragment CRLs. They achieve this by extending the CRL with a new extension that can be used to redirect the user to a proper CRL Distribution Point that contains CSI for a given certificate serial number range. Using the Redirect CRL schema it is possible for the CA to limit the size of CRL however the cost of processing such CSI flow increases.

2.1.6 Windowed Certificate Revocation

This schema [11] was inspired by an environment where a full certificate are retrieved from a trusted repository when the relying party is verifying the certificate. In this *implicit* model, the revocation check is delegated to the repository that never returns a certificate that has been revoked. The client may not mount any revocation checks then. The Windowed Certificate Revocation schema strives to combine this implicit model with the *explicit* mechanism of CRL. Clients retrieve certificates from the repository and store them in a local cache. Validity of the certificates stored in the cache is verified using the CRLs acquired from the repository. Unlike standard use of CRLs revoked certificates are included in the CRL only for limited period—*revocation windows*. Using this approach the authors the authors optimized the cost as compared with standard CRL schema.

2.2 Mechanisms providing positive CSI

R. Rivest suggested in [18] not to use revocation as a mechanism to deal with compromised private keys and recommended to use separate evidence that the corresponding private has not been compromised. A new service ,,suicide bureau" (SB) was introduced, which is responsible for issuing *certificates of health* stating that a given certificate has not been compromised.

2.3 Mechanisms providing complete CSI

Mechanisms described in this section provide a complete information if a given certificate has been revoked or not. Some of the mechanisms may provide users with a succinct proof, which can be piggybacked to other peers along with the users' credentials.

2.3.1 Online Certificate Status Protocol

OCSP [15] is an IETF protocol that brings the ability to the end services to ask for validity of the certificate online. OCSP is a request/response protocol based on binary ASN.1 encoded messages usually transferred over HTTP. An OCSP service consists of a responder and a client, with the responder holding a list of certificate status information and the client querying the current status for a given certificate.

The OCSP responder is either operated by the CA or a third-party provider, its location can be either embedded directly in the certificate or specified by configuration of the end service (acting as the OCSP client).

One of the major limitations of OCSP is the requirement on having a generally trusted responder, which is operated online. The necessity to trust the responder(s) makes the relying parties deployment and operation more difficult. It is also worth mentioning that the responder statements can be as accurate as information maintained by the responder back-end. Therefore it is inevitable for a CA using an OCSP responder to ensure that revocation information is available for the responder immediately upon its publication. Such an arrangement can become a issue in a large environment with multiple responder.

Since OCSP responds are digitally signed using public-key cryptography their construction is quite expensive in terms of CPU time. It is clear that end services have to ask the responder for every certificates they handle. Such requests may easily overload the responder therefore caching of responses on the end services was proposed. When the end service obtains a response from the responder it saves the statement and can use it for subsequent validation for a defined period of time. This time should be set up on the basis of information from the response.

2.3.2 Certificate Revocation Status

CRS was introduced in [12] and later improved in [13]. Unlike CRL, in the CRS schema the CAs issue statements on both valid and revoked certificates, which are made available to the relaying parties. These statements are issued individually for each certificate. CRS is based on the same principle as the *Lamport schema* [9] and uses one-way functions to construct light-wight signatures on the statements. The CA issues a new statements each day and publishes them in the repository. The relying parties can retrieve and verify them very easily. One possible drawback of this method in large deployment can be the fixed number of revocations that must be specified on certificate generation.

2.3.3 Certificate Revocation Trees

The CRT schema [7] is based on Merkle hash trees, where leaves bear information about revoked certificates. The root of the tree is digitally signed. The trees are usually being maintained by responders that are used by the relying parties requesting current status of the certificates they are just verifying. The responder returns the leaf with information about the given certificate and all the nodes of the tree that are needed to re-construct the path from the leaf to the root. Using such a structure the responder is not required to sign every respond that is sent to the clients.

Serious shortcoming of the CRT method is a high computational cost of updates of the trees. Naor and Nissim suggested a more efficient method based on use a 2-3 tree [16], which reduced the complexity and the computing demands necessary for updates.

The idea of using hash trees for CSI management was further developed in [14] where the overall performance of the system was improved again.

3 Revocations handling in Grid

Consulting revocation information is very often neglected in current deployments of PKI, as they need additional tools to be installed and maintained. However, revocation checks are inevitable in large distributed environments where PKI is widely used, such as contemporary Grids. For example, utilizing the PKI, the EU EGEE project [4] joins together thousands users from all the world and provides them with access to grid facilities. Obviously, performing revocation checks is a must within such an environment. Presently, CRLs are used to distribute revocation information inside EGEE and they are the only means generally supported by the EGEE middleware—gLite [10]. A CRL retrieval script is shipped as a standard part of the gLite distribution, which is used for automatic download of the CRLs for all CAs installed on a given system. The script is quite simple and can be used in other PKI installations as well. It is usually started four times a day, which leaves quite a long window in distribution of revocation information. Moreover, given the huge number of relying parties, the CA servers are often overloaded and the CRL retrieval attempts often fail, making the CRL distribution be significantly delayed. For example, according to the information published by the Italian INFN CA, their CRL server handled a huge number of CRL retrieval request, ranging from 800 thousand in May 2005 to 1.5 million requests in April 2006. Curiously number of certificates revoked by the INFN CA in this period by was quite small, totaling just 67 certificates. However, basing on assumptions given for another large environment [17], we expect that the revocation rate will grow. As the number of certificates used in Grid is growing we believe current methods of revocation checks may become unfeasible. Therefore we look for a solution that could provide more scalable solution for the Grid environment.

Several solution were described in previous section that are suitable for various environments. Unfortunately we lack implementations of most of the methods that would be available in the middleware used currently. Therefore most of the Grid systems rely on use the basic version of CRL, which ensures interoperability and compatibility of the systems. There are also efforts ongoing to utilize OCSP to check revocation in the Grid environments since it is supported by basic middleware tools such as openssl.

In the first step we performed a simple test of these two mechanisms to evaluate their demands. The OCSP faces possible huge on-line traffic of OCSP requests and computational complexity so it does not seem as a perfect solution to the revocation management problem as it faces possible huge traffic of OCSP requests. Also, OCSP adds another point of failure into the already complex infrastructure. And the last but not least problem is substantial increase of the delay of requests to the end services with the certificates due to the necessity to contact responder. In the next section we discuss this problem in more detail.

3.1 Comparison of OCSP and CRL

We set up a testbed to study the difference in latency when validating the certificate with OCSP and with CRL distribution. The testbed consisted of an Apache server acting as an end service performing certificate validation, two OpenCA OCSP responders providing revocation information about certificates, and an `wget` client application. Every service ran on a separate machine. Being based on standard services, the testing environment represents a real-world service running e.g. a web service container.

For testing purposes 100 CAs and 500 client certificates per CA were generated and also 100 randomly chosen certificates from each CA were revoked and corresponding CRLs generated. The certificates contained a standard X.509 extension that points to appropriate OCSP responder. All the CRLs were copied to the Apache web server, CRLs from the first 45 CAs were copied to the first OCSP responder and CRLs from the last 55 CAs were copied to the second OCSP responder.

The goal of the test was to measure differences in time that were needed to serve an HTTP GET request sent over an SSL connection. We studied performance of three different scenarios: checking using CRLs stored in local files, checking using OCSP and disabled checking at all. Using the `wget` client we initiated several connections to the web server, each using a randomly selected certificate of a CA for client authentication. The test represented sequential access of 10, 50, 100,

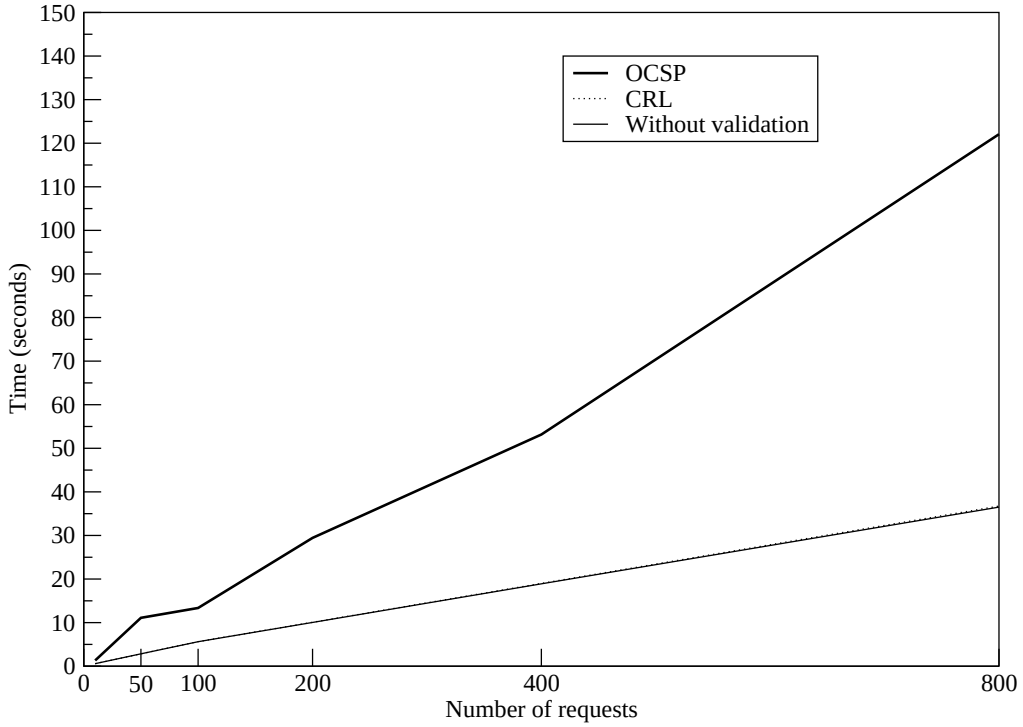

Figure 1: OCSP vs. CRL

200, 400 and 800 users from 100 different CAs. We choose sequential access to avoid parallel optimization of Apache web server.

The results of the test can be seen in Figure 1. The time in graph represents total run time of the script. Each number is a mean value of ten repeated tests. The graph shows big difference between the OCSP and CRL validation time. Difference between CRL and without validation is negligible. This simple test thus demonstrated the huge overhead associated with the use on on-line service for each certificate validation, confirming the hypothesis that the tests must be done on a locally available data.

Having analyzed the cost of the solutions we concluded the CRL overhead is reasonably small in terms of CPU time required for each verification. Unfortunately the bandwidth requirements needed to retrieve CRL are huge, which prevents the CRLs from being updated timely. One of the possibility we considered was to adapt current Grid middleware to support one of the revocation schemes described in Section 2. Such an approach would require large changes to the current infrastructure and also influence the operation of the systems so it would be very hard and error-prone process. Instead of trying to adapt current Grid middleware to support a new revocation mechanism we decided to change the way how CRLs are delivered to the relying party. This approach allowed us to use the same middleware.

4 Using the Push model for CRL distribution

As explained in the previous section, the OCSP introduces a significant performance penalty while the current concept of CRL does not ensure timely distribution. We present an alternative

schema for revocation data distribution based on the *Push* model, where the relaying parties do not actively poll the CA seeking fresh CRLs but instead subscribe for this information and rely on the messaging infrastructure to deliver to them a new CRL immediately after it has been published by the CA. Using the Push model is a novel approach to the CRL distribution and is not supported by current CAs, therefore we propose a *CRL distribution service* (CRL DS) that collects revocation data from one or more CAs and distributes it to the clients. The schema of the service is depicted in Figure 2.

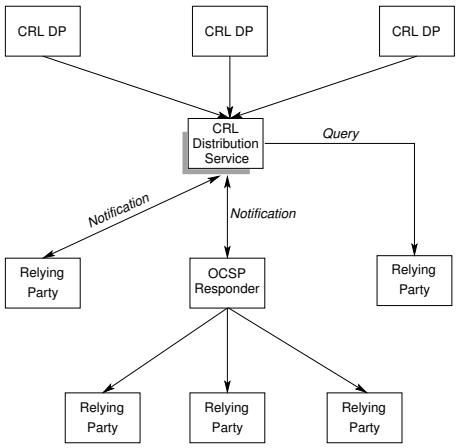

Figure 2: Schema of the CRL Distribution service

The service will be responsible for maintaining current CRLs, which are collected from the CA CRL distributed points. The CRLs have to be retrieved by a regular polling of the CAs, which is a standard and supported way for the operation. As only few instances of the CRL DS are expected to operate compared with the number of relying parties, the interval between the downloads can be very short. After retrieving a new CRL, the CRL DS will store it in its database and trigger sending a notification message containing the new CRL to the subscribed clients. The message will be put into a notification infrastructure which ensures its delivery to the relying parties. The infrastructure must be scalable enough to serve all the clients and also robust to cope with various errors that may occur during transmissions. For example, if a a particular client is temporarily unavailable the messaging infrastructure must ensure it will be contacted immediately after it re-connects again. After receiving the notification message, the client will decode it and store the obtained CRL in a local system replacing the old one.

While the notification messages are the primary way of distribution, the CRL DS will also provide an interface to fetch a current list of all CRLs as maintained by the service. The interface should allow the clients to specify a subset of the CAs they are interested in or additional criterion to be used for selection of the CRLs. This interface is expected to be used by clients without a permanent network connectivity, e. g. mobile users. Such users are very often forced to use a slow or unreliable network so the messages exchanged with the CRL DS should be as small as possible. The CRLs returned by the service should be always sent in a single message.

CRLs are always signed with the CA private key, which ensures their reliability and also provides integrity protection. If they were tampered with during the transmission, the end system would immediately detects the change and refuse to install the received CRL. The notification infrastructure therefore need not to provide any additional protection of the messages.

Once the CRL is delivered to the local system (regardless if notifications or queries were used) it replaces the current CRL stored locally and can be immediately used by the application running on the system. Thus the mechanisms does not require any change in the application code or configuration provided the applications are ready to work with locally stored CRLs. The CRL DS can even be combined with additional components, e. g. it can be used as a source of CRL data for an OCSP responder serving OCSP enabled applications.

Since the CRL may become quite large the service can save the bandwidth using a binary diff algorithm (e. g. [21]) and transporting only differences that can be applied to the previous CRL producing the latest one. In order to avoid potential problems with authenticity and ordering of the differences it is necessary that the messaging protocol be reliable so that no message can be lost. After completing the CRL the client must verify the signature of the new CRL, which also ensures the CRL has not been tampered with during transmission.

The design and requirements outlined above are very similar to the concepts used by the Grid monitoring architecture (GMA)[20], which defines a general framework for monitoring in highly distributed environment. Having gained a lot of experience with monitoring in Grids we decided to base the CRL Distribution Service on an implementation of a GMA and to utilize the results we achieved in this area.

5 CRL Distribution using the Logging and Bookkeeping service

This section describes our pilot implementation of the proposed schema for CRL distribution. The implementation is based upon the L&B architecture that provides a GMA and which has proven to be flexible enough to provide all necessary functionality.

5.1 Logging and Bookkeeping Service

The Logging and Bookkeeping (L&B) service [8, 3] is developed within the EU EGEE project as a real time monitoring system for tracking jobs on large Grids. It is a standard part of the gLite software stack produced by the project and has been in production for many years now. The L&B architecture consists of two parts: the L&B server processing data about the jobs, and the L&B messaging infrastructure which is responsible for reliable and fault-tolerant delivery of monitoring information from the Grid components to the L&B server.

The L&B approach requires that each grid component that processes a job moving through the grid environment is instrumented to send to the L&B server a message about all important events in the job lifetime. In this way various internal components of the Grid scheduler, and eventually the end computing resource are instructed to send L&B messages about the jobs they process. The messages are carried as *L&B events* using a dedicated L&B messaging infrastructure allowing the L&B server to collect effectively all events for the jobs. The infrastructure is based upon mediators that provide entry points to easy insert a message and ensure the message is delivered to the proper L&B server. The mediators are installed closely to the components generating the L&B messages and use their disk as a persistent storage of events that cannot be temporarily delivered. After receiving an L&B event, the L&B server stores it in the database and updates the job information including current state of the job. The state is computed by the L&B state machine based on the received events. The state machine reflects a life of the job in the grid environment, corresponding state diagram can be seen in Figure 3. A classical sequence of states for a job starts with a SUBMITTED state, meaning that the Grid scheduler accepted the job to process. This state is followed by WAITING meaning the job is waiting to be scheduler for a particular computing resource, then READY (job can be sent to the selected resource), SCHEDULED (job was accepted by the computing resource and is waiting in a local queue), RUNNING (job was actually started), DONE(OK) (job finished successfully), CLEARED (the output was retrieved by the user).

Apart from the standard events generated by the middleware, the user may also append additional information to their jobs using a special event delivering a UserTag. This way jobs can be marked with an arbitrary label, which can be used to distinguish one group of jobs from another.

The L&B server exposes a powerful consumer interface that can be used by the clients to query for state of their active jobs as maintained by the server. The queries may specify a number of additional condition, such as various time constraints or appended user tags. In additions to the the query interface, the L&B server also allows the users to subscribe for notifications that are sent by the server on changes, according to the criterion specified by the client. In this way the

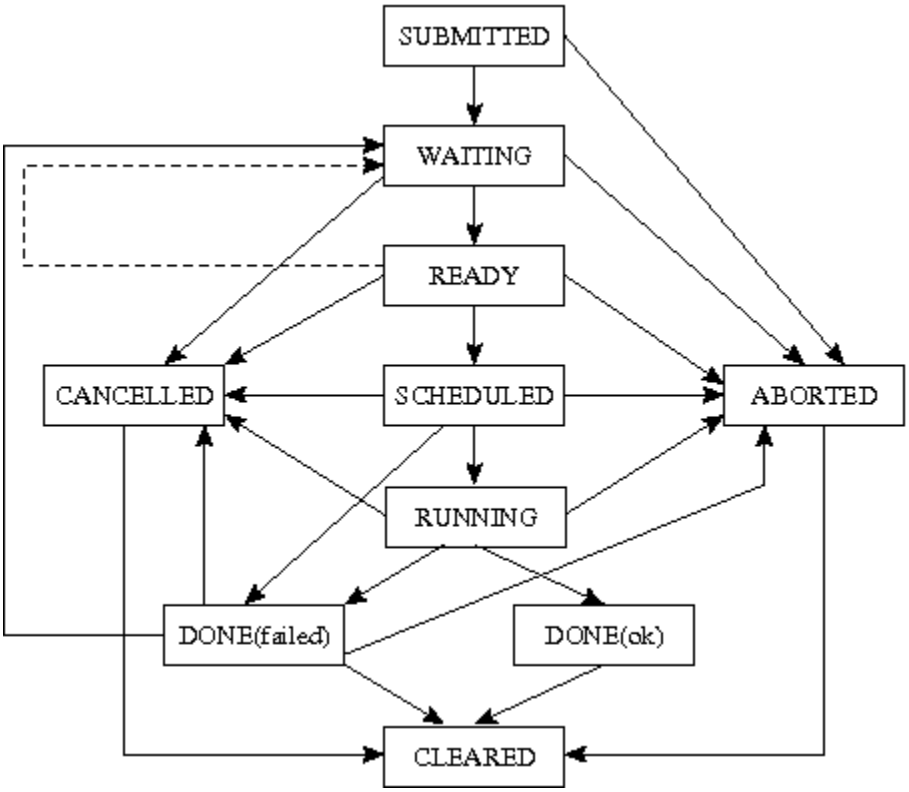

Figure 3: L&B job state diagram

users can wait for an asynchronous message instead of periodically polling the L&B service to check if the state of their job(s) changes. The notification messages are also sent via the L&B infrastructure ensuring a reliable delivery even when the client is temporarily inaccessible at the time the notification is being generated and sent. An overall schema of accessing the L&B server data is given in Figure 4.

The L&B architecture was designed with security in mind, thus all communication channels are protected using the TLS protocol and all components must have X.509 certificates to authenticate. Access to the job information is only allowed by the job owner by default. However, the L&B server allows the users to specify access control lists for individual jobs to allow other users to share the job data.

5.2 Using L&B to distribute CRLs

Having examined the revocation information flow, we noticed the life cycle of a CRL can be represented using a simple state diagram, which reflects important points in its lifetime. Based on this observation we decided to treat the revocation information issuance as a single job, which can be handled by the L&B service and its state machine. Subsequently, the L&B's implementation of the GMA makes it possible to distribute the CRLs across the community in a fast and robust manner. Especially, using the notification functionality the relying parties are notified immediately upon updating the CRL without any additional overhead. Also, the powerful query interface can be used by clients to ease the CRL maintenance and the bootstrapping process. With all these features grouped together, the L&B server could act as an instance of the CRL Distribution Service as described in Section 4.

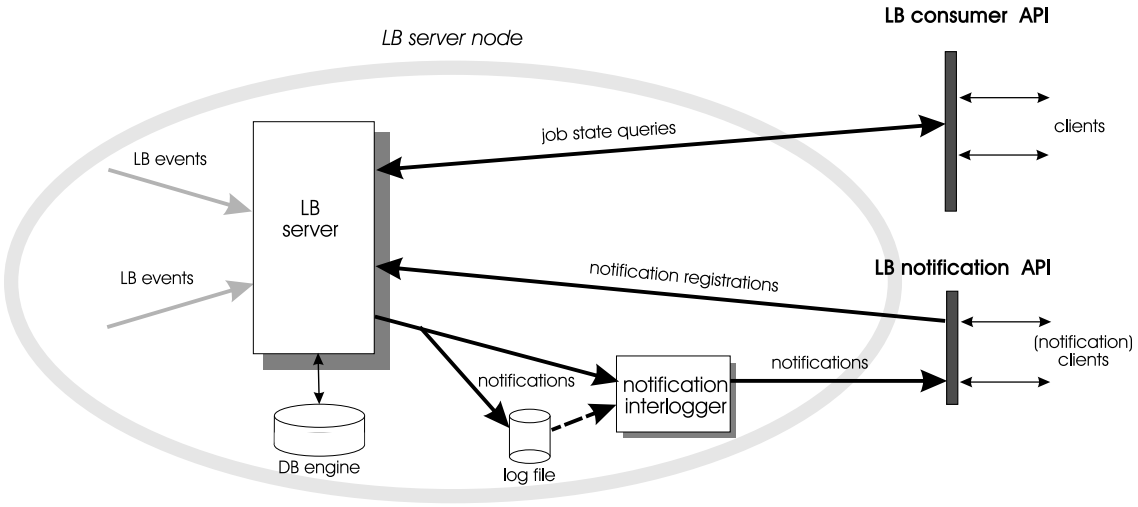

Figure 4: L&B queries and notifications

5.2.1 Storing CRLs to the L&B

The CRL lifetime can be described using three basic states, starting with an initial state (REGISTERED), which refers to establishment of the CA before the very first CRL is issued or entered to the L&B database. After that, the (PUBLISHED) state is entered repetitively in a loop, marking each publication of a new version of the CRL. If the CA ceases operation and stops issuing the CRL, the lifetime of CRL concludes in a final state (TERMINATED). To better use the potential of the L&B job state machine we added a new state (UPDATING) to split the loop over the (PUBLISHED) state into two steps. The state diagram can is presented in Figure 5.

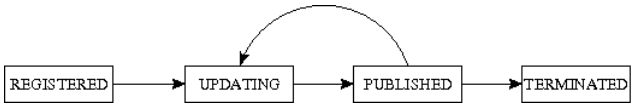

Figure 5: CRL life state diagram

This CRL life cycle can be simply mapped to a subgraph of the standard L&B job state diagram and thus can be handled by the standard L&B processing.

We model each CRL as a standard L&B job record in the L&B database. In order to register a new CA and a corresponding CRL, a new job entry is established, which will carry all the necessary information. Namely, the job record contains the CRL (if available) and also additional information, such as timestamps referring to transitions between states etc. Each job entry is assigned a globally unique identifier, which is usually generated by the L&B server on registering the job with L&B. However, when a new CA is being registered, a specially crafted job identifier is requested by the registration client, which consists of a network identification of the L&B server (triplet of the protocol, hostname, and port), common prefix labeling the CRL jobs, and a hash value computed from the public key certificate of the CA. This way the job identifier can be constructed by any client solely using the CA certificate (provided the L&B machine is known).

After a new CRL job is registered with L&B, it is necessary to ensure it is updated properly upon each change of the CRL. In order to update the job record in L&B few events must be generated and sent to the L&B. This causes the CRL job to switch to the state UPDATING, assign the new CRL, and finally switch back to the PUBLISHED state. Each such transition between states can trigger a notification to be send if any client subscribed to receive it. The contents of the CRL is stored along with the job as a custom UserTag attribute. Thus it is available later if the job information is queried from the L&B server. Timestamps of each state

change are automatically added to the L&B record, which allows users to query e. g. only for CRLs that changed from a given point in time.

We want the service to be as general as possible handling multiple independent CAs. Therefore we do not use any direct link to the CA served, but utilize the standard CRL monitoring tools instead. We adapted the EGEE default retrieval script to use the L&B command-line tools and send appropriate events to the L&B server whenever it detects a new CRL. We configured the script to be invoked every five minutes to aggregate CRLs of all supported CAs and check for changes in them. Such a short interval allows to detect a new CRL very quickly.

5.2.2 Retrieving CRLs

As described before the users have two ways of getting the CRLs—notifications and queries (see also Figure 4. Notifications provides the most convenient way as they do not require any special activity on the client side and ensure the new CRL is delivered to the clients almost immediately after its issuance. On the other hand queries provide complementary functionality and are useful when notifications cannot be used by the client or if additional operations with the CRLs are needed.

In order to receive the notification messages the clients must subscribe first to their receiving and start a client on the local machine, which is able to handle the notification messages. In the subscription request the client specifies CAs whose CRL they want to be notified about. Job identifiers used to specify the CAs are derived from the CA public key certificate as described in previous section. The notification messages are generated by the L&B server upon each state change of a registered CA. After such a change, a notification message is sent to all subscribed clients, which contain the full state information, included the new CRL.

Notification messages are delivered using the L&B messaging infrastructure, which guarantees their reliable delivery. It is able to cope with clients temporarily unavailable and finish the delivery as soon as the clients connect again. It is even possible for the clients to change their location without having to unsubscribe the notification and subscribe a new one. It is sufficient if the client reconnects to the L&B server after change and announce its new address. This functionality is targeted to mobile users who can roam between different networks and still keep their CRLs current.

Each notification registration is assigned a lifetime and must be renewed before it expires. Length of the lifetime is configurable on the server side and is usually couple of days, in our pilot installation we configured the notification lifetime to one year. The notification must be renewed by the clients before it expires.

An issue concerning the notification mechanism may be the necessity for the client application to be reachable from outside. This requirement cannot be met if the client machine is behind a firewall or NAT. Users limited with such an environment can still use the queries to keep the CRL refreshed. This approach represents a roll back to the classical CRLs download, as the client must periodically poll the L&B server. Advantage of the L&B based solution is that all the CRLs can be queried and downloaded in a single step. The queries can ask either for a full set of current CRLs or only for those CRLs that changed since a particular point of time (incremental updates). As the L&B queries can contain several conditions that specify the set of CRLs, the clients can ask queries such as: *Give me CRLs by CAs A,B,C that changed since last Monday.* The resulting set of CRLs is returned in a single message, which significantly decreases demands on network communication compared with the standard way of retrieving CRLs.

A couple of ways is available to query the L&B server. Simple query for a particular CRL can be performed using the standard gLite commands, for more complex queries a specialized client implementation must be used. The L&B provides a C/C++ API, which can be used to create queries and getting responses. The L&B server also offers a Web Service interface that can be easily accessed by a range of various clients.

We developed a pilot implementation of a client that is able to use both notifications and queries. In the former case it is run as a daemon on the client system, which subscribe to notifications for a locally installed CAs and waits for the messages from the L&B. The later mode

can be used e. g. by mobile clients to fast update their CRLs, supporting both incremental and full updates. Regardless the way of retrieving CRL from the DS, the CRLs delivered to the client machine are stored on a disk and made available to the middleware and applications in the same way as if they were retrieved directly from the CAs.

The L&B messaging is secured using TLS, which provides general security but also generates a chicken and egg problem in the deployment described in this report. Since the CRLs are consulted whenever a new TLS connection is being initialized, the L&B based DS will fail to deliver a new CRL to a client whose CRL for the L&B certificate already expired. This applies for both notification and synchronous queries. There are a few workarounds to cope with the problem. One can use a specialized CA used only to issue a single certificate (for the L&B DS). In the future the L&B infrastructure could also be adopted not to use the TLS protocol. Such a change would not open any serious problem since the CRL are already signed.

6 Evaluation

In the next three figures we give an evaluation of the bandwidth cost of the push model compared with OCSP and the standard pull model. For completeness's sake we also added a push model delivering only CRL differences instead of full CRLs. The test case consists of 100 CAs each issuing 3 000 certificates, 100 services that perform verification of the certificates. Frequency of the verification is 10 requests per second. We have calculated with the 5 % rate of the certificate revocation. In the CRL Pull model we have set up four updates of the CRL per day. Time period is one year. Variables are explained in the table 1.

Table 1: Defined variables of the test case

Variable	Value	Description
ca_num	100	Number of CAs
c	3000	Number of issued certificates by one CA
p	0.05	Rate of revoked certificates (%)
q	315360000	Number of verification per year
q_day	864000	Number of verification per day
l_sig	0.0025	Size of the signature (MB)
l_sn	0.00002	Size of one item in the CRL list (MB)
diff_p	0.1	Reducing factor of the CRL list
f	4	Frequency of the CRL update per day for CRL Pull

In graph 6 there is shown influence of the number of clients in the infrastructure on the distribution of the revocation data. The OCSP method has a constant value because this method only depends on the number of verifications. Three other methods depend on the number of clients because we set up 5 % rate of the certificate revocation from all issued certificates. It is clear that the size of the CRL list grows and has to be distributed to all services. Functions used for the calculation are explained in the table 2.

Table 2: Definitions of the functions used in graph 6

Function	Description
crl_size(k) = k * p * l_sn + l_sig	Size of the CRL list from one CA
crl_pull(x) = f * 365 * ca_num * crl_size(x)	Total size of the transfered CRL in CRL Pull model
crl_push(x) = p * ca_num * x * crl_size(x)	Total size of the transfered CRL in CRL Push model
crl_push_diff(x) = crl_push(x) * diff_p	Total size of the transfered CRL in CRL Push Diff model
ocsp(x) = q * l_sig	Total size of all OCSP requests

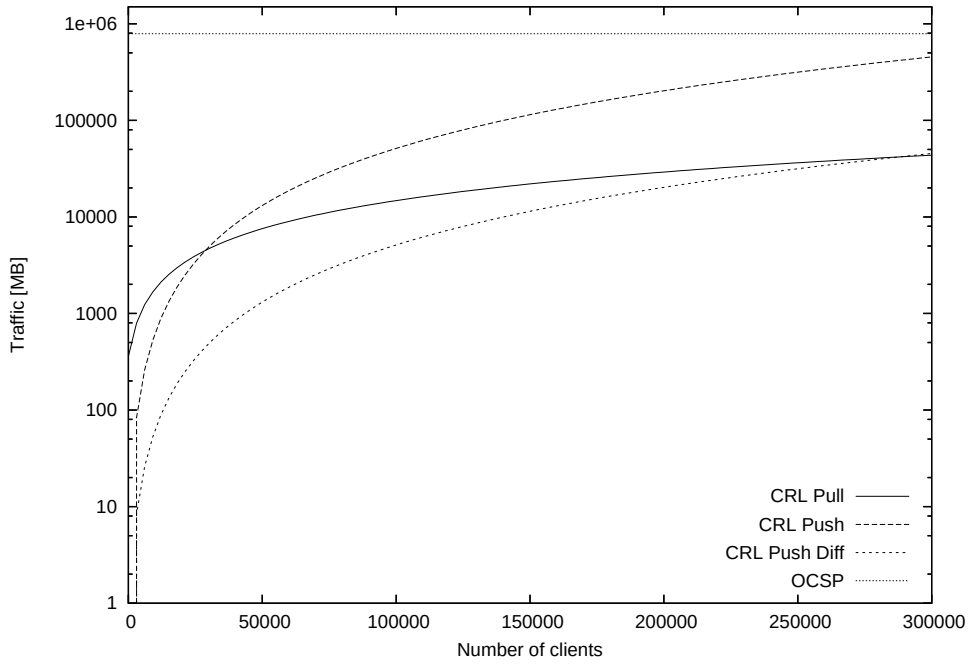

Figure 6: Influence of the number of clients on the distribution of the revocation data

Graph 7 shows a changing number of the requests which has influence only on the OCSP because verification with the CRL does not require any on-line network communication. The value for CRLs is aggregated amount of data needed for transmit CRLs to the services over one year. The CRL Pull model has better performance then CRL Push due to the update interval being set to four times per day, while the CRL Push method updates CRL at the services after each change of the CRL.

Table 3: Definitions of the functions used in graph 7

Function	Description
crl_size(k) = k * p * l_sn + l_sig	Size of the CRL list from one CA
crl_pull(x) = f * 365 * ca_num * crl_size(c)	Total size of the transfered CRL in CRL Pull model
crl_push(x) = p * ca_num * c * crl_size(c)	Total size of the transfered CRL in CRL Push model
crl_push_diff(x) = crl_push(x) * diff_p	Total size of the transfered CRL in CRL Push Diff model
ocsp(x) = x * l_sig	Total size of all OCSP requests

As we can see in graph 8 if the revocation rate in our test environment exceeded 28 % the OCSP method has better performance then the CRL Push model but not better then the CRL Push Diff model. This is due to growing size of the CRL revocation list and growing frequency of revocation. The CRL Pull model has the best performance but the CRL data are not really up to date. This graph represents one day.

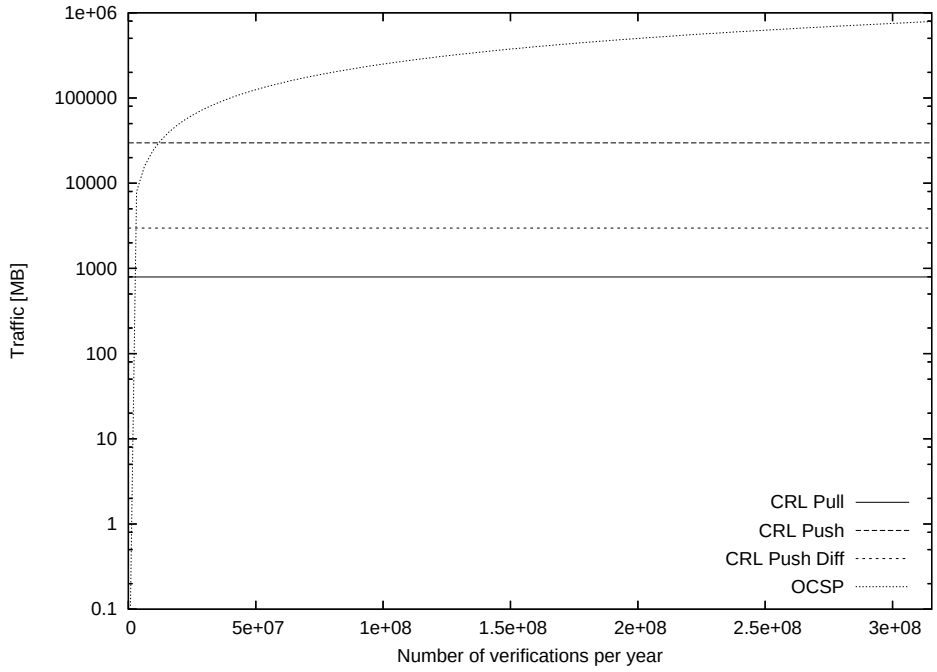

Figure 7: Influence of the number of verifications on the distribution of the revocation data

7 Conclusion

We presented a novel way of distributing revocation information using the grid monitoring architecture and its messaging features. The mechanism described ensures a fast and reliable delivery of CRLs to end services, which need not to contact any other network services to get current revocation information. We proposed a concept of the CRL Distribution Service that utilizes the GMA functionality and presented a pilot implementation of the service. The implementation is based on the gLite L&B service, which was primarily designed to job monitoring, however we demonstrated the service can be used even for more general tasks.

The pilot implementation primarily supports sending notification messages that contain new CRLs. In addition to this Push model, a query interface can be used, which offers an easy way of aggregating current CRLs, allowing for both full and incremental updates.

Table 4: Definitions of the functions used in graph 8

Function	Description
crl_size(k) = c * k/100 * l_sn + l_sig	Size of the CRL list from one CA
crl_pull(x) = f * 365 * ca_num * crl_size(x)	Total size of the transfered CRL in CRL Pull model
crl_push(x) = (x/100 * ca_num * c) * crl_size(x)	Total size of the transferred CRL in CRL Push model
crl_push_diff(x) = crl_push(x) * diff_p	Total size of the transferred CRL in CRL Push Diff model
ocsp(x) = q_day * l_sig	Total size of all OCSP requests

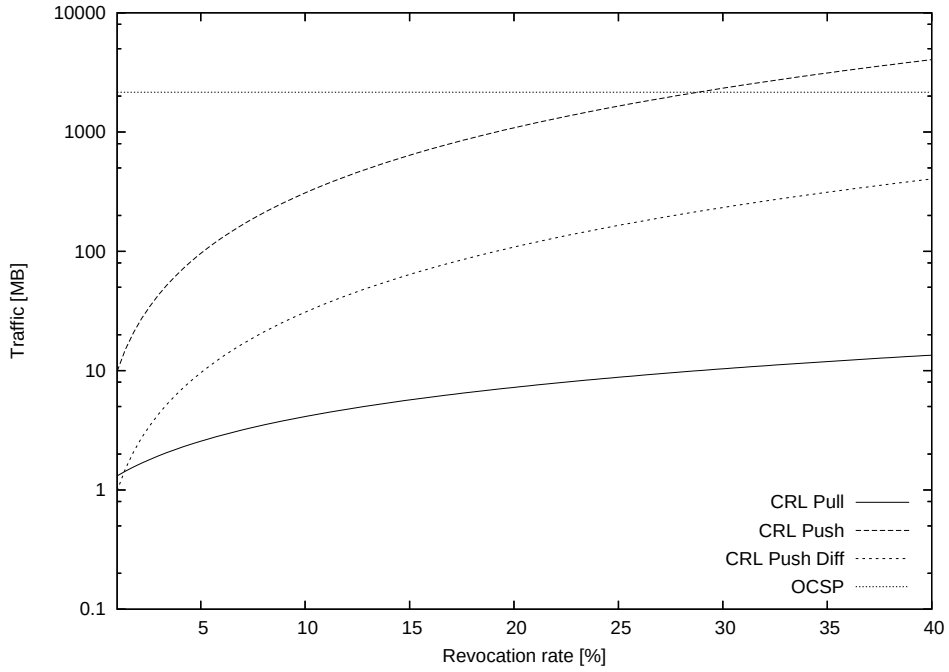

Figure 8: Influence of the rate of revocation on the distribution of the revocation data

Acknowledgment

The work has been supported by the research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201) and EGEE. EGEE is a project funded by the European Union under contract INFSO-RI-508833. We also acknowledge the national funding agencies participating in EGEE for their support of this work. We also thank all the designers and developers of the gLite L&B service.

References

[1] C. Adams and R. Zuccherato. A general, flexible approach to certificate revocation. Technical report, Entrust, 1998.

[2] A. Årnes, M. Just, S. J. Knapskog, S. Lloyd, and H. Meijer. Selecting Revocation Solutions for PKI. In *NORDSEC 2000 Fifth Nordic Workshop on Secure IT Systems*, 2000.

[3] F. Dvořák, D. Kouřil, A. Křenek, L. Matyska, M. Mulač, J.Pospíšil, M. Ruda, Z. Salvet, J. Sitera, J. Škrabal, and M. Voců. Services for Tracking and Archival of Grid Job Information. In *Proceeding of Cracow Grid Workshop*, 2005.

[4] EGEE. Enabling Grids for E-sciencE (EGEE) Project. `http://www.eu-egee.org/`.

[5] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280, April 2002.

[6] J. Iliadis, S. Gritzalis, D. Spinellis, D. de Cock, B. Preneel, and D. Gritzalis. Towards a Framework for Evaluating Certificate Status Information Mechanisms. *Computer Communications*, 26(16):1839–1850, Oct. 2003.

[7] P. C. Kocher. On certificate revocation and validation. In *Financial Cryptography*, 1998.

[8] D. Kouřil, A. Křenek, L. Matyska, M. Mulač, J. Pospíšil, M. Ruda, Z. Salvet, J. Sitera, J. Škrabal, and M. Voců. Distributed Tracking, Storage, and Re-use of Job State Information on the Grid. In *Computing in High Energy and Nuclear Physics (CHEP04)*, 2004.

[9] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11), Nov. 1981.

[10] E. Laure, F. Hemmer, F. Prelz, S. Beco, S. Fisher, M. Livny, L. Guy, M. Barroso, P. Buncic, P. Kunszt, A. Di Meglio, A. Aimar, A. Edlund, D. Groep, F. Pacini, M. Sgaravatto, and O. Mulmo. Middleware for the next generation Grid infrastructure. In *Computing in High Energy Physics and Nuclear Physics (CHEP 2004)*, 2004.

[11] P. McDaniel and S. Jamin. Windowed Certificate Revocation. In *INFOCOM (3)*, 2000.

[12] S. Micali. Efficient Certificate Revocation. Technical Report MIT/LCS/TM-542b, MIT, 1996.

[13] S. Micali. NOVOMODO: Scalable Certificate Validation And Simplified PKI Management. In *1st Annual PKI Research Workshop*, Apr. 2002.

[14] J. L. Muñoz, J. Forné, O. Esparza, and M. Rey. Efficient Certificate Revocation System Implementation: Huffman Merkle Hash Tree (HuffMHT). In *Trust, Privacy and Security in Digital Business*, 2005.

[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 *Proceedings 7th USENIX Security Symposium (San Antonio, Texas)*, Jan 1998.

[17] N. I. of Standards and Tables. Public Key Infrastructure Study. Final Report, Apr. 1994.

[18] R. L. Rivest. Can We Eliminate Certificate Revocations Lists? In *Financial Cryptography*, 1998.

[19] A. Slagell, R. Bonilla, and W. Yurcik. A survey of PKI Components and Scalability Issues. In *IEEE International Performance, Computing, and Communications Conference*, 2006.

[20] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A Grid Monitoring Service Architecture. Global Grid Forum Performance Working Group, 2001.

[21] A. Tridgell. *Efficient Algorithms for Sorting and Synchronization.* PhD thesis, The Australian National University, 2000.

[22] ITU Recommendation X.509 (03/00). Information technology – Open systems interconnection – The directory: Authentication framework, 2000.

[23] P. Zheng. Tradeoffs in Certificate Revocation Schemes. *SIGCOMM Computer Communication Review*, 33(2), 2003.