

Regular Paper

**Notification of Certificate Revocation  
Status between Different Domains  
under a PKI System**

Yutaka Miyake (KDDI R&D Laboratories Inc.)  
[Regular Member#199301141]

Jonathan Millen (SRI International Inc.) [Non Member]

Grit Denker (SRI International Inc.) [Non Member]

Toshiaki Tanaka (KDDI R&D Laboratories Inc.)  
[Regular Member#198723548]

Koji Nakao (KDDI R&D Laboratories Inc.)  
[Regular Member#198209686]

Contact To:

Yutaka Miyake  
Network Security Laboratory  
KDDI R&D Laboratories Inc.  
2-1-15 Ohara, Kamifukuoka-shi, Saitama, 356-8502  
phone: (049)278-7367      facsimile: (049)278-7510  
email: miyake@kddlabs.co.jp

## **Abstract**

When public key certificates are used to control access by a client in one domain to a server in another domain, the certificate revocation status should be distributed to the server domain too. For security reasons, the distribution of information to other domains should be minimized, and external distribution points are subject to attack from third parties on the Internet. In this paper, we propose a mechanism to securely convey the current revocation status of certificates to other domains under a PKI (Public Key Infrastructure) system in the Web environment. Because our proposal does not need to modify standard browsers, we can introduce the proposed method into the current Web environment easily. We implemented a prototype system, and evaluated the system to prove its effectiveness.

## 1. Introduction

The combination of Web browsers and servers makes it convenient for anyone to access information on servers. Some servers provide private or sensitive information and require users to log in with password or smart cards. This form of access control is suitable if users must be distinguished from one another.

Recently, many organizations have been introducing PKI (Public Key Infrastructure) and distribute public key certificates for each member. The certificate includes user information, a public key for the user, and a signature by the CA (Certificate Authority). If the CA is trustworthy and it is proved that a user has a private key that corresponds to the public key in the certificate, the user can be identified and authenticated.

It is possible to use a PKI system for access control. In case of Web accessing, Web servers can identify the accessing clients by checking their certificates and possession of private keys. This method has several advantages compared to systems with passwords or smart cards. The main difference between the PKI and a password-based system is the procedure after the password or private key is compromised. In case of password-based access control, the client should change the password for every Web server. If the client accesses many Web servers with the same password, this procedure will be required for each Web server. Though the clients may assign different passwords for each Web server, this method necessitates troublesome password management. In the case of PKI, when the private key is revealed, the client revokes the certificate corresponding to the private key, and asks the CA to issue a new certificate. It is not necessary to inform every Web server that the private key and certificate are altered, because the Web servers obtain this revocation status information when they validate the certificate of accessing clients. Even if someone attempted to use the old certificate, when the revocation status of the certificate is checked, it is proved that the certificate is invalid.

In order to incorporate the PKI system into access control, the server has to have the capability to validate the certificates of clients. Normally, the server checks the revocation status of a certificate by retrieving a CRL (Certificate

Revocation List)<sup>1),2)</sup> or accessing an OCSP (Online Certificate Status Protocol)<sup>3)</sup> responder. If the client and server belong to the same domain and their certificates are issued by a CA in this domain, it is easy for the server to check the status of a client's certificate. However, if the client and server belong to different domains and their CAs are managed independently, a mechanism to inform the server in the other domain of the status of the certificate needs to be provided.

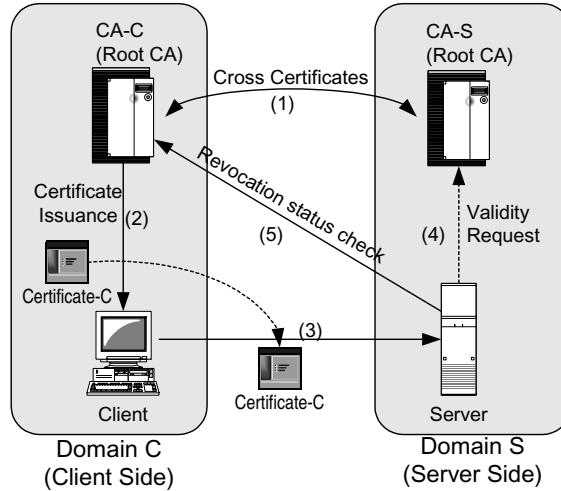
In this paper, we discuss a new notification method for conveying the status of certificates between different domains for the Web environment. In this situation, both domains rely on their own root CA and there may be some restrictions to access resource servers in the other domain. Therefore we propose an appropriate procedure to exchange the status of certificates safely. In the proposed procedure, when the client accesses the server, the client side sends the revocation status of the client certificate to the server domain. Therefore, the server domain can always have the up-to-date revocation status of clients. Moreover, the proposed procedure can be introduced to the Web environment easily without any modifications of browser software. We have implemented a prototype system of our proposal, and evaluated the system to confirm that the procedure is compatible with the current Web environment.

The rest of this paper is organized as follows: In the next section, we describe the notification procedure to convey the certificate revocation status between different domains. In Section 3, we propose the necessary adaptations to implement our notification procedure in the Web environment. In Section 4, we explain the implementation of the prototype system. Section 5 discusses the proposed procedure, and conclusions are presented in Section 6.

## **2. Notification of certificate revocation status between different domains**

### **2.1 Connection of two independent PKI domains by cross certificates**

In order to construct a certificate chain from a client certificate to a CA in the server domain, both the CA that issues the client certificate and the CA



**Fig. 1** Certificate validation using cross certificates

in the server domain would normally have the same root CA at the top of the CA hierarchy. Generally, the certificate chain connection is made via the root CA. However, many domains manage their own root CA independently. In this case, another method is needed to establish a certificate chain between different domains.

Cross certificates<sup>4)</sup> have been introduced to resolve this problem. **Figure 1** shows the mechanism of cross certificates. In this figure, there are two domains, Domain C for the client side and Domain S for the server side. Domain C and Domain S have their own root CA, CA-C and CA-S respectively. In the environment of cross certificates, CA-C and CA-S exchange their public key certificates and sign the received certificates. This means that the server that relies on CA-S can accept certificates signed by CA-C, because there is a certificate for CA-C signed by CA-S. In this situation, the server in Domain S can validate the certificate of the client (Certificate-C) with the following procedure.

- (1) The root CAs in Domain C (CA-C) and Domain S (CA-S) exchange their public key certificates. The received certificates are signed by each root CA.
- (2) The CA-C in Domain C issues a public key certificate for a client (Certificate-C). This certificate is signed by the CA-C.
- (3) When the client accesses a server in Domain S, it sends its public key certificate (Certificate-C) to the server.

- (4) The server checks the validity period and signature in the certificate. If the signature of this certificate is signed by the CA-C, the server needs to retrieve the certificate for CA-C. Therefore the server retrieves the cross certificate for CA-C, and checks its signature. Because the cross certificate for CA-C is signed by its own CA (CA-S), the server can rely on it. If the server can validate the certificate (Certificate-C) by using the cross certificate, it is deemed conditionally valid.

## **2.2 Check of revocation status for certificates between different domains**

The sever can validate certificates issued by CAs in other domains by using the cross certificates. After validating the signature in client certificate, the server should check the revocation status of the client certificate ((5) in Fig. 1). If the client certificate is revoked before it expires, the certificate becomes unavailable.

In order to get the revocation information for certificates, the server should retrieve a CRL<sup>1),2)</sup>, which is a list of revoked certificates, or should access an online verification server, such as an OCSP responder<sup>3)</sup>. In case of the CRL, the server requires the CRL that was issued by the same CA that issued the client certificate. The online verification server also needs the information from the CA that issued the client certificate. Therefore the server has to access another domain to retrieve the revocation information for the client certificate if the CA in another domain issued it.

When the server retrieves the CRL from another domain or accesses the online verification server in another domain, there are some problems.

- Each domain must prepare an access point that provides certificate status information outside of the firewall for other domains. Because the access point can be accessed from other domains, maximum security protection is required for its data.
- Access to the access point should be restricted to known, registered domains. The access point should not send information about its domain to unrelated domains.
- The access point should provide only the information that is required to access other domains. Outflow of unnecessary information may reveal the

organizational structure in its domain, and it may become a security hole.

These problems also increase the management cost for each domain, and increase the risk of intrusion from other domains by placing the access point outside of the firewall.

### 2.3 Push mechanism for notification of revoked status

Because it is not safe to make an external access point for other domains, we use a push mechanism for notification of revocation status. This means that the client domain sends the status information directly to the other domains that need this information.

We rejected the approach in which each domain distributes CRLs to other registered domains periodically. This method has the following disadvantages.

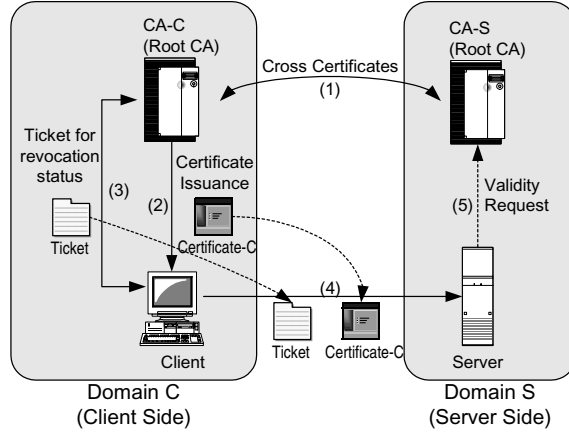
**Lack of scalability:** The CRL information that should be sent to other domains increases along with the number of registered domains. If there are many registered domains that need CRLs, this method requires a high speed network.

**Useless information:** The CRL lists the serial number of all revoked certificates. However most of this information may not be used at the other domains. It may mean that useless information is sent to many domains periodically.

**Information leak:** Since the CRL includes every serial number of revoked certificates, a change in this information may suggest management policy of the PKI, organizational restructuring, etc., in that domain. Therefore each domain should keep to a minimum the information sent to other domains.

In order to resolve these problems, we propose the mechanism shown in **Fig. 2**. In this mechanism, the client sends a ticket that has the status of the client's certificate. The procedure for this mechanism is as follows.

- (1) Root CAs in Domain C (CA-C) and Domain S (CA-S) exchange their public key certificates. The received certificates are signed by each root CA.
- (2) The CA-C in Domain C issues a public key certificate for a client (Certificate-C). This certificate is signed by the CA-C.
- (3) Just before accessing a server in Domain S, the client requests a ticket



**Fig. 2** Transfer of revoked status information with ticket

that certifies the non-revoked status of client’s certificate by the CA-C. The validity period of this ticket is short.

- (4) When the client accesses the server in Domain S, it sends both its public key certificate (Certificate-C) and the ticket to the server.
- (5) The server checks the conditional validity of the certificate as before. It also checks the signature of the received ticket with the same process. If the ticket is not expired and it indicates that the certificate has not been revoked, the certificate is accepted as valid.

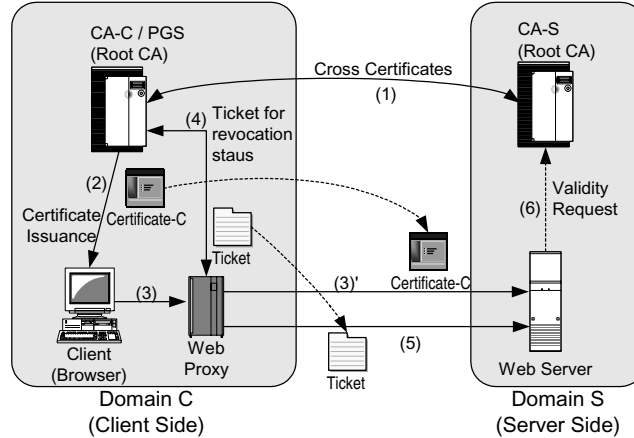
Only the minimum amount of information is sent to the another domain in this mechanism. Since the ticket that includes the status information is sent with the certificate, the client can also know the time and domain where the status of the client certificate is sent.

### 3. Adaptation to Web Environment

In order to apply our proposed approach to a practical Web environment, some modification and additional elements are required. In this section, we describe a system architecture that incorporates our approach into the Web environment with PKI.

#### 3.1 Assumptions

Currently, browsers and servers for the Web are used widely. Therefore, it is desirable that the elements of current system, such as browsers, Web servers, CA for PKI, can continue to be used. Accordingly, we designed the system



**Fig. 3** System architecture

architecture in consideration of the following points.

- There are several kinds of browsers. We would not like to modify each browser one by one. Moreover, we would not like to distribute special browser software to each client for this system. Therefore, we do not want to customize the client browser for the envisioned system.
- Since it is not easy to add new functions to a CA, we decided to make a separate software module called PGS (Privilege Granting Server). The function of this software is to issue the ticket that indicates the status information of a certificate. Since the ticket is signed with the private key of the CA, we assume that the PGS is executed on the same computer on which the CA is running.
- SSL (Secure Socket Layer)<sup>5),6)</sup> has the capability that the server can authorize the client by using the client's certificate (SSL Handshake Protocol). Because many browsers and Web servers support SSL, we decided to use this protocol for our system.

### 3.2 System Architecture

**Figure 3** shows the system architecture that incorporates our method. We adopted a Web proxy design for ticket handling. When the client browser uses the Web proxy to access the Web server, the Web proxy obtains a ticket and transfers it to the server. The procedure for this architecture is as follows.

- (1) Root CAs in Domain C (CA-C) and Domain S (CA-S) exchange their

public key certificates. The received certificates are signed by each root CA.

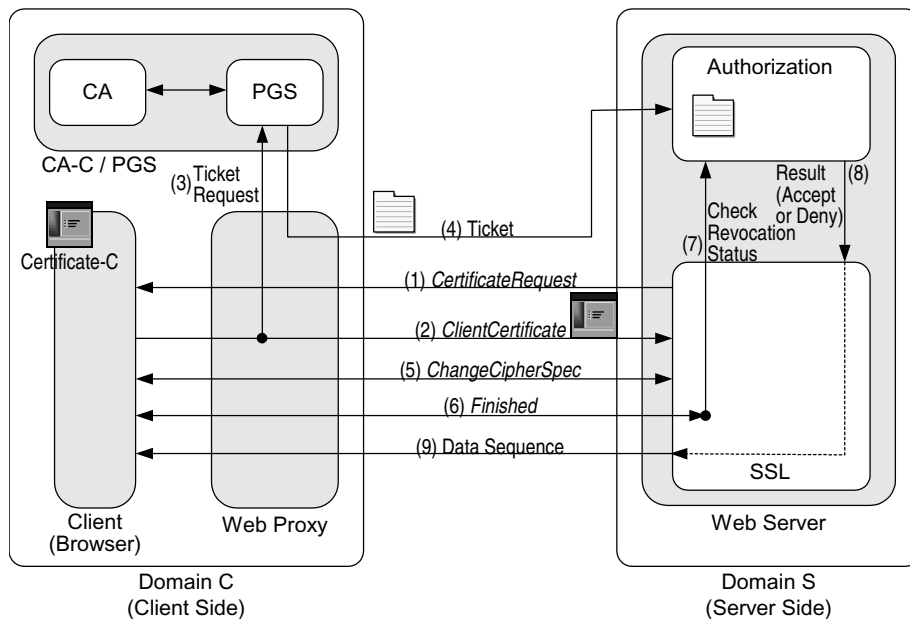
- (2) CA-C in Domain C issues a public key certificate for a client (Certificate-C). This certificate is signed by CA-C.
- (3) The client browser tries to access the Web server in Domain S via the Web proxy in Domain C. The Web proxy relays communication between the browser and Web server.
- (4) During the SSL handshake, the Web proxy extracts the certificate information sent by the client. The Web proxy requests the status of the certificate from the CA-C / PGS, and receives the ticket.
- (5) The Web proxy sends the ticket to the Web server in Domain S. This action is independent of communication between the browser and Web server.
- (6) The server checks the validity periods and signatures in the certificate and ticket as before.

A new Web proxy and some modification of the Web server is required in this architecture. However, there is no modification of the client browser because the Web proxy executes the additional procedure instead of the client. The Web proxy might run on the client workstation, or it might run on a gateway so that it can be shared by all the clients on a local network.

### 3.3 Protocol for Authorization and Ticket

**Figure 4** shows the protocol for client authorization. (The steps that do not relate to client authorization are omitted in this figure.) SSL handshake protocol messages are shown in italics.

- (1) In the SSL handshake protocol, the Web server requests the public key certificate of the client by using SSL *CertificateRequest* message.
- (2) The client returns the certificate by using the SSL *ClientCertificate* message.
- (3) The Web proxy in Domain C extracts the certificate from the message *ClientCertificate*. It asks the PGS to issue the ticket for this certificate.
- (4) After the Web proxy receives the ticket from the PGS, it transfers the ticket to the Web server in Domain S. The authorization module in the



**Fig. 4** Protocol for client authorization

Web server receives this ticket. This communication is independent from the SSL handshake protocol. The authorization module is a new function added to the web server to validate and read the ticket.

- (5) Both the browser and Web server send the *ChangeCipherSpec* message. After this message, all messages are encrypted.
- (6) Both the browser and Web server send the *Finished* message. This message completes the SSL handshake protocol.
- (7) When the Web server receives the *Finished* message from the browser, it checks the revoked status of certificate by contacting the authorization module. This check takes the place of other activities like password checking that are performed in other servers to authenticate the client.
- (8) If the result is “Accept”, and access is permitted to this client, data on accessed pages are transferred to the browser. In case that result is “Deny”, messages that indicate access denial are sent to the client.

The Web server does not alter the communication sequence between the server and client. It observes the SSL message passively to obtain the client certificate. It does not know any private key or decrypt any encrypted messages. Therefore the secret or authenticated communication between client and server

is protected from the Web proxy after the SSL handshake is established.

The ticket uses the OCSP Response Message format<sup>3)</sup>. Minimally the following information has to be included in this ticket.

- CA Name
- Serial Number of Client Certificate
- Revocation Status (Revoked, Not Revoked, Unknown)
- Validity Period of This Ticket
- Signature by CA

## 4. Implementation of Prototype System

We have implemented the proposed architecture as a prototype system. Three components (Web server, PGS and proxy) were developed for this system. In this section, we describe the outline of these components.

### 4.1 Web Server

For the server side, we chose the Apache HTTP server software and Tomcat as an engine to enable the use of servlets and Java Server Pages (JSPs) with Apache. Apache is built and installed simultaneously with `mod_ssl`, the Apache interfaces to the OpenSSL toolkit that supports the SSL and TLS protocols<sup>5),6)</sup>.

The authorization server is just another service of the Web server, implemented as a Java servlet. The servlets are associated with protected Web pages. Each one is an instance of a Java class that performs an access-control check when the page is requested through the Web server. The authorization server accepts tickets from the proxy, and it also handles requests from a servlet to confirm that a client certificate has not been revoked.

The servlets control access to the protected Web pages by making three checks:

- the client certificate is properly signed and not expired;
- there is a ticket providing that the client certificate has not been revoked;  
and
- there is an ACL (Access Control List) entry (in `.xdaaccess` file, see next section) permitting access.

## 4.2 ACL

Access control and authorization at the server are based on the contents of the client certificate, using some form of ACL. While access could be controlled on the basis of the user or subject name, the client CA can also create client certificates containing a more general “privilege” field. The privilege field is implemented as an X.509 extension field.

The ACL is a file called `.xdaaccess` (analogous to the `.htaccess` file used by Apache). Each `.xdaaccess` file applies to Web pages in its directory and all subdirectories. It contains entries that explicitly allow or deny access on the basis of each privilege string. For example, if “Manager” and “Officer” are privileges, an `.xdaaccess` file might contain the lines:

```
# Deny everyone except for
# Manager and Officer
deny all
allow Manager
allow Officer
```

Privileges may be considered to belong to a hierarchy, with implications for inheritance of access rights. Possible extensions to tickets and ACL processing to support a privilege hierarchy in the context of RBAC (role-based access control) have been suggested in another paper<sup>7</sup>.

## 4.3 PGS

The PGS is implemented as an OCSP responder for X.509 certificates. This responder is compliant with RFC2560<sup>3</sup>). The OCSP status information for certificates is received from an LDAP server specifying the certificate database.

The PGS is implemented as a Java servlet executed by Tomcat. The configuration files for the PGS servlet contain initialization parameters for the servlet and set up a secure area used by the servlet.

## 4.4 Proxy

The proxy is an SSL proxy written in Java. It assumes that the server will request a client certificate for secure SSL connections. SSL versions 3.0 and 3.1 are supported in our experimental implementation. Since there are no other constraints on the machine on which the proxy runs, it can be run on the end-

user's machine so that there is one proxy per user, or it can be executed as a client-side proxy, in which case many users are supported by one proxy.

The proxy listens for a client certificate passed from the client to a server. Once detected, the proxy forwards the client certificate to the OCSP responder (PGS). The proxy receives the OCSP response and forwards it to an authorization server in form of an `HTTP POST` message. The authorization server can be a static server, or the proxy constructs its URL from the SSL server's IP address, a default port and a directory path defined in the proxy configuration file. The proxy assumes that the OCSP response is compliant with RFC2560<sup>3</sup>).

The proxy can contact the PGS synchronously or asynchronously. If the proxy works in a synchronous fashion, it is guaranteed that the authorization server gets the ticket before the SSL handshake between server and client is completed, whereas in the asynchronous mode the handshake may complete without the ticket having arrived at the authorization server. In the latter case, an access request may be denied, in spite of the fact that the client certificate is valid and has privileges that would allow access, but it can then be retrieved.

## 5. Discussion

The proposed scheme is reminiscent of Kerberos<sup>8</sup>) because the PGS performs a function similar to the Ticket Generating Server (TGS) in Kerberos, to create tickets for use with an application server. The original Kerberos had an Authentication Server to which the user would log in with a password, but currently there are extensions to Kerberos to permit the use of public key certificate for user identification. Our approach has a very different notion of the purpose and structure of a ticket, however, due to its use of public keys and access control in dealings with the application servers.

The PKDA (Public Key Distributed Authentication) approach is another outgrowth of Kerberos that uses public key certificates<sup>9</sup>). Its objective is to distribute PGS functions to "PKDA-enabled" servers, and the tickets it generates are essentially Kerberos tickets, containing a symmetric session key to be used with a specific server. A ticket in our proposal is different because it is a status response associated with a standard X.509 certificate, which conveys a public

key that can be used with available protocols like SSL or TLS to set up authentication or data encryption as desired, and it establishes group membership in a way that is not confined to a specific server.

If certificates have short validity periods, we may not need the revocation information. If a client certificate was issued only a few minutes ago, servers do not need to validate the received certificate. However, this increases the load on the CA because new certificates must be sent more frequently. A two-level staged expiration scheme has been suggested<sup>10)</sup>, but this more complex system still requires a “suicide bureau” to maintain revocations due to key compromise<sup>11),12)</sup>.

## 6. Conclusions

We have proposed a PKI mechanism to support access control of a client in one domain to a server in another. When the server and client belong to different PKI domains, and the server needs to check the revocation status of the client certificate, a normal PKI system must retrieve a CRL or access an online verification server, such as an OCSP responder. However, from the viewpoint of security, it is not desirable for the client domain to provide an access point outside of an Internet firewall and to distribute unnecessary information. Therefore we have proposed a mechanism to send a ticket that includes the revocation status of certificate along with the client certificate. With this mechanism, only the minimum status information is sent, it is sent unforgeably, and only when access is requested. An external access point to distribute status information to other domains is not required.

In order to integrate this architecture easily into the Web environment, we designed a system architecture that does not require modification to Web browsers. It uses a Web proxy that observes an SSL connection passively, a PGS module to support ticket generation in conjunction with a client CA, and an authorization module in the Web server to check access using the certificate and ticket.

Access control and authorization at the server are based on the contents of the client certificate, using some form of ACL. While access could be controlled on the basis of the user or subject name, the client CA can also create client

certificates containing a more general privilege field. Access control by privilege is more efficient when the server has many individual clients, and is willing to trust the client CA to assign server privileges to clients in its domain. This approach is supported by a certificate extension field and the `.xdaaccess` file as described.

## References

- 1) ITU-T Recommendation X.509: Information Technology – Open Systems Interconnection– The Directory: Public-Key and Attribute Certificate Frameworks, ITU-T (2000).
- 2) Housley, R., Ford, W., Polk, W. and Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC-2459 (1999).
- 3) Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, RFC-2560 (1999).
- 4) Adams, C. and Farrell, S.: Internet X.509 Public Key Infrastructure Certificate Management Protocols, RFC-2510 (1999).
- 5) Freier, A., Karlton, P. and Kocher, P.: The SSL Protocol: Version 3.0, Netscape Communications Corp. (1996).
- 6) Dierks, T. and Allen, C.: The TLS Protocol: Version 1.0, RFC-2246 (1999).
- 7) Denker, G., Millen, J. and Miyake, Y.: Cross-Domain Access Control via PKI, Third International Workshop on Policies for Distributed Systems and Networks, pp. 202–205 (2002).
- 8) Kohl, J. and Neuman, C.: The Kerberos Network Authentication Service (V5), RFC-1510 (1993).
- 9) Sirbu, A. and Chuang, J.: Distributed Authentication in Kerberos Using Public Key Cryptography, Internet Society 1997 Symposium on Network and Distributed System Security (1997).
- 10) Rivest, R.: Can We Eliminate Certificate Revocation Lists?, Proc. Financial Cryptography '98, LNCS 1465, pp. 178–183 (1998).
- 11) Wright, R., Lincoln, P. and Millen, J.: Efficient Fault-Tolerant Certificate Revocation, ACM Conference on Computer and Communications Security,

pp. 19–24 (2000).

- 12) McDaniel, P. and Rubin, A.: A Response to “Can We Eliminate Certificate Revocation Lists?”, Proc. Financial Cryptography 2000 (2000).

## Acknowledgments