Recommended Paper

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

Yutaka Miyake,† Jonathan K. Millen,†† Grit Denker,††
Toshiaki Tanaka† and Koji Nakao†

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 convey the current revocation status of certificates to other domains securely under PKI (Public Key Infrastructure) system in WWW environment. Because our proposal does not need to modify the standard browsers, we can introduce the proposed method to the current WWW environment easily. We implemented prototype system of our proposal, and evaluated the system to prove the effectiveness of this system.

1. Introduction

The combination of WWW (World Wide 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 (Certification 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 the PKI system for access control. In case of WWW accessing, Web servers can identify the accessing clients by checking their certificates and possession of private keys. This method has several advantages compared with systems with passwords or smart cards. The main difference between the PKI and 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 revoked 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) [1),2)] or accessing an OCSP (Online Certificate Status Protocol) [3)] responder. If the client and server belong to same domain and their certificates are issued by a CA in this domain, it is easy for the server to check the status of client's certificate. However if the client and server belong to different domains and their CAs are managed inde-

† KDDI R&D Laboratories Inc.
†† SRI International Inc.

The initial version of this paper was presented at CSEC Workshop on July 2001, which was sponsored by SIGCSEC. This paper was recommended to be submitted to the Journal of IPSJ by the chairperson of SIGCSEC.

pendently, a mechanism to inform the server in the other domain of the status of the certificate should be provided.

In this paper, we discuss the notification method for conveying the status of certificates between different domains for WWW environment. In this situation, both domains rely on their own root CA and there may be some restrictions to access resource servers on 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 client certificate to the server domain. Therefore, the server domain can have revocation status of clients as recent as possible in access always. Moreover, the proposed procedure can be introduced to the WWW environment easily without any modifications of browser software. We have implemented prototype system of our proposal, and evaluated the system to confirm that the procedure is compatible with the current WWW environment.

The rest of this paper is organized as follows: In the next section, we describe the procedure of notification to inform the revoked certificate status between different domains. In Section 3, we propose the procedure of notification between different domains for the WWW environment. In Section 4, we explain the implementation of prototype system. Section 5 discuss the proposed procedure, and conclusions are presented in Section 6.

2. Notification of Revoked Certificate Status between Different Domains

2.1 Connect of Two Independent PKI Domains by Cross Certificates

In order to construct a certificate chain from a client certificate to a CA in server domain, both the CA that issues the client certificate and the CA in server domain would normally have a same root CA at the top of the CA hierarchy. Generally, the certificate chain 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 [4] 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 client side and Domain S for server side. Domain C and

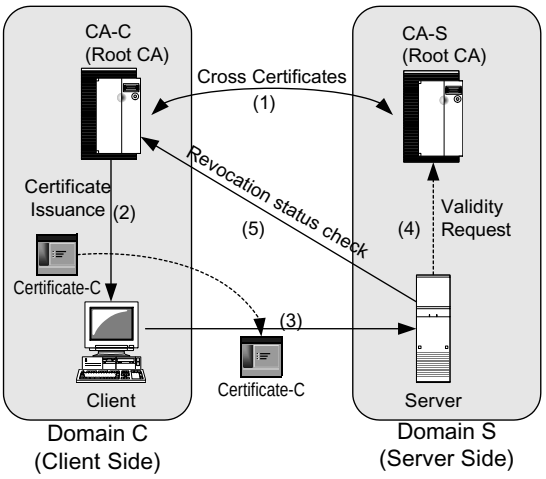

Fig. 1 Certificate validation using cross certificates.

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 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 the reliable CA (CA-S), the server can rely on it. If the certificate (Certificate-C) is signed by a public key in the certificate for CA-C, 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 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 [1),2)] that is a list of revoked certificates, or should access an online verification server, such as an OCSP responder [3)]. 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 revoked status. This means that the client domain sends the status information directly to the other domains that need this information directly.

As a notification method of the revoked certificate status, 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).

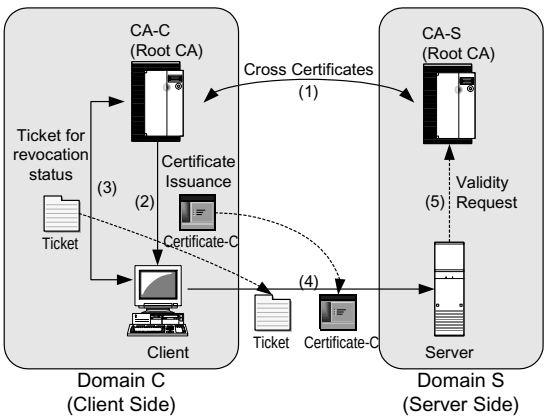

Fig. 2 Transfer of revoked status information with ticket.

This certificate is signed by the CA-C.

(3) Just before accessing a server in Domain S, the client requests a 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 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 WWW Environment

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

3.1 Assumptions

Currently, browsers and servers for WWW are used widely. Therefore, it is desirable that the current system (browsers, Web servers, CA for PKI) can be used continuously. Accordingly, we designed the system 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 software to each client for this system. Therefore, we do not customize the client browser for this system.
- It is not easy to add new functions to a CA. Therefore, we decide to make a separate software module called a 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 CA, we assume that the PGS is executed in same computer on which the CA is running.

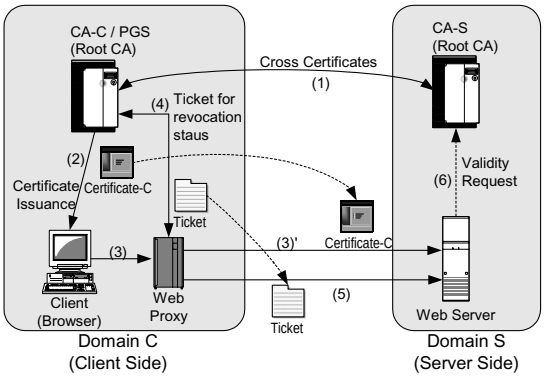

Fig. 3 System architecture.

- SSL (Secure Socket Layer)[5),6)] 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 function for our system.

3.2 System Architecture

Figure 3 shows the system architecture that incorporates our method. We adopted 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 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

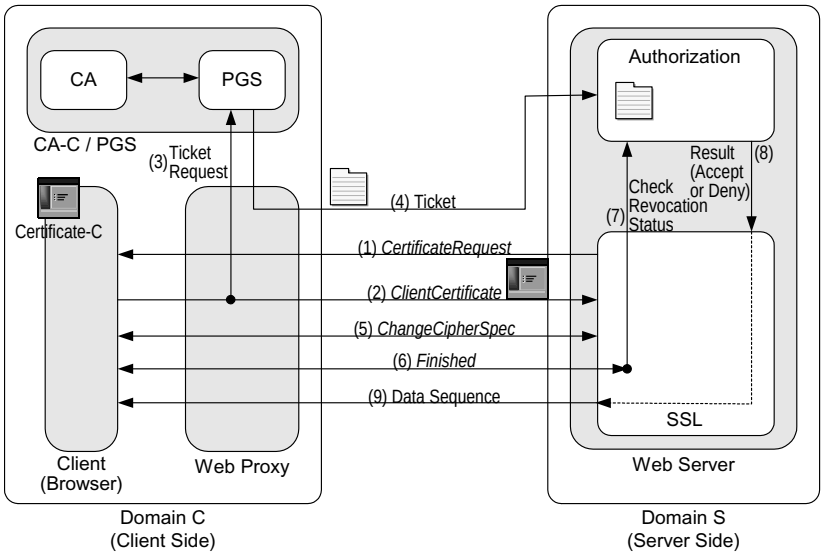

Fig. 4 Protocol for client authorization.

the Web server are 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 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 control client access.

(8) The SSL module in the Web server receives the result (Accept or Deny) of revocation status from the authorization module.

(9) If the result is Accept, data on accessing 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 [3]. 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 prototype system. Three components (Web server, PGS and proxy) are developed for this system. In this section, we describe the outline of these components.

4.1 Design Policy

In order to realize easy adaptation to current WWW environment, we have considered the following issues to design each component.

- As described in Section 3.2, our proposal needs to modify the Web server. Because the program of Web server may be upgraded for correcting security holes and adding new functions, the modification for proposed mechanism should be embedded easily even if the Web server would be upgraded.
- Because almost browsers can specify the proxy server by protocol, the Web proxy does not need to support all protocols used by the browsers. So we have decided to implement `https` protocol (HTTP with SSL) only for the proxy. In this case, because other protocols, such as `http` and `ftp`, does not pass the proxy, we do not need to support these protocols in the proxy.

4.2 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 OpenSSL toolkit that supports the SSL and TLS protocols [5),6)].

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. Because this service is implemented as servelet, we can adapt this component to the program easily even if the program of Web server is upgraded.

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 and ACL (Access Control List) entry (in `.xdaaccess` file) specifying the privilege in the client certificate, and permitting access.

Each `.xdaaccess` file applies to Web pages in its directory and all subdirectories. It contains ACL entries that explicitly allow or deny access on the basis of a privilege. 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
```

4.3 PGS

The PGS is implemented as an OCSP responder for X.509 certificates. This responder is compliant with RFC2560 [3)]. 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 constrains 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 associate to 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 sever can be a static server, or the proxy construct 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 [3].

This proxy can define whether the proxy is contacting 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 being 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.

5. Discussion

The proposed scheme is reminiscent of Kerberos [7] 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 [8]. 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 so much. Because a client certificate was issued within few minutes ago and will expire within short minutes, servers do not need to validate the received certificate [9),10]. However, as described in Ref. 9), this short-lived-certificate framework is not appropriate for Internet applications, where there is little control over both clients and servers. In case of short-lived-certificate, both servers and clients need to handle the certificate smartly. For example, the servers need to distinguish the short-lived-certificate and long-lived-certificate. When the servers receive the short-lived-certificate, they should not check the revocation status of it. Therefore, we need to modify the servers to distinguish the short-lived-certificate. The clients also need modification to handle the short-lived-certificate. Today, most Web browsers assume that certificates are long-lived. For example, these browsers do not automatically removed expired certificates and their associated key pairs. Because the Web browsers must obtain a new certificate whenever starting or expiring a previous certificate under short-lived-certificate framework, management of certificates annoys the users of browsers without modification. Since our approach does not need to update the certificate frequently, it is said that this approach is appropriate for general Web browsers.

In Section 2.3, we avoid to use of the CRL. Though one of the reason is lack of scalability, delta-CRL may resolve this problem [2]. Because difference between the previous CRL and current CRL is sent under the delta-CRL framework, communication overhead for sending the CRL may decrease with this mechanism. However the size of revocation information is unpredictable under the delta-CRL mechanism. Moreover, several problems described in Section 2.3 are not resolved by using the delta-CRL. Since the server sends minimum revocation information on appropriate timing in our approach, the proposed scheme is superior to the delta-CRL under multiple domain environment.

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, the server requires retrieving a CRL or accessing to an online verification server, such as an OCSP responder, to check a revocation status of the client certificate in normal PKI system. However, from the viewpoint on 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 the mechanism to attach a ticket that includes the revoked status of certificate with the sending client certificate. With this mechanism, only the minimum status information is sent, it is sent unforgeably, and only when access is requested. The external access point to distribute status information to other domains is not required.

In order to implement this architecture easily into the WWW 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 can be supported without change to the architecture as described.

References

1) *ITU-T Recommendation X.509: Information Technology —Open Systems Interconnection— The Directory: Public-Key and Attribute Certificate Frameworks* (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. (1999).
6) Dierks, T. and Allen, C.: The TLS Protocol – Version 1.0, RFC 2246 (1999).
7) Kohl, J. and Neuman, C.: The Kerberos Network Authentication Service (V5), RFC 1510 (1993).
8) Sirbu, M. and Chuang, J.: Distributed Authentication in Kerberos Using Public Key Cryptography, *Internet Society 1997 Symposium on Network and Distributed System Security* (1997).
9) Hsu, Y. and Seymour, S.: An Intranet Security Framework Based on Short-Lived Certificates, *IEEE Internet Computing*, Vol.2, No.2, pp.73–79 (1998).
10) Rivest, R.: Can We Eliminate Certificate Revocations Lists?, *Financial Cryptography '98*, *LNCS 1465*, pp.178–183 (1998).

(Received September 5, 2002)
(Accepted March 4, 2003)

Editer's Recommendation

The authors propose a unique method for the very important topic of the distribution of certificate revocation information in PKI.

(Chairman of SIGCSEC Eiji Okamoto)

Yutaka Miyake received the B.E. and M.E. degrees of Electrical Engineering from Keio University, Japan, in 1988 and 1990, respectively. He joined KDD (now KDDI) in 1990, and has been engaged in the research on high speed communication protocol, secure communicating and privacy protection system for network applications. During a year of 2000 to 2001, he was a visiting researcher at SRI International and University of California, Berkeley. He is currently a senior research engineer of Network Security Lab. in KDDI R&D Laboratories Inc. He received IPSJ Convention Award in 1995. He is a member of IEICE.

Jonathan K. Millen is a Senior Computer Scientist in the Computer Science Laboratory, SRI International, working in the area of dependable systems and cryptographic protocol verification. From 1969 to 1997 he worked at The MITRE Corporation. He is co-Editor-in-Chief of the Journal of Computer Security, an associate editor of the ACM Transactions on Information and System Security, and the founder of the annual IEEE Computer Security Foundations Workshop. He holds a Ph.D. in Mathematics from Rensselaer Polytechnic Institute, an M.S. from Stanford, and an A.B. from Harvard.

Grit Denker is a Computer Scientist in the Computer Science Laboratory at SRI International in California. From 1995 to 1996 she worked as an assistant professor at the Technical University of Braunschweig in Germany. Her main areas of interest are formal specification and verification of cryptographic security protocols, security for the Semantic Web, and logic-based approaches for distributed system analysis. She holds a Ph.D. in Computer Science and an M.S. in Mathematics from the Technical University of Braunschweig.

Toshiaki Tanaka received the B.E. and M.E. degrees of Communication Engineering from Osaka University, Japan, in 1984 and 1986 respectively. He joined KDD (now KDDI) and has been engaged in the research on network security, cryptographic protocol, mobile security, digital rights management, and intrusion detection techniques. He is currently a senior manager of Network Security Lab. in KDDI R&D Laboratories Inc. He is a member of IEICE.

Koji Nakao received the B.E. degree of Mathematics from Waseda University, Japan, in 1979. He joined KDD (now KDDI) and has been engaged in the research on multimedia communications, communication protocol, secure communicating system and information security technology for the telecommunications network. He received IPSJ Research Award in 1992. He is a member of IEICE. He has been a part-time instructor in Waseda University and The University of Electro-Communications since 2002.