PKIX Working Group
INTERNET-DRAFT

M. Sakurai (NEC)
H. Kikuchi (Tokai Univ)
H. Hattori (Meiji Univ)
Y. Sameshima (ICAT)
H. Kumagai (ICAT)

expires in six months

Jan 31, 1999

Web-based Integrated CA services Protocol, ICAP

<draft-sakurai-pkix-icap-01.txt>

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Abstract

This document provides a sub set of specifications how to issue, publish X.509 certificates and certificate revocation lists (CRLs). It also provides the certificate validation service by online. In the proposed specifications, the World Wide Web (WWW) is used for secure distributing certificates across a firewall in both human and machine readable syntax. These specifications define not only the protocols between the PKI clients and a single CA, but also the protocols between the CAs. With the CA-CA communications, the PKI clients can retrieve any certificates and CRLs without specifying the location of the appropriate CA, by only asking to the neighbor CA.

Table of Contents

1. Introduction

This document defines a Web-based protocol to integrate typical CA services such as certificate issuing request and certificates/CRLs retrieval, based on a practical and compact CA model. In our model, a CA supports:

- o multiple application-specific certificate profiles
- o online certificate issuance based on password authentication
- o certificates and CRLs retrieval using CA-CA communications

First, [PKIX-PROF] defines the general X.509 Certificate and CRL profile, and applications compliant to this profile will increase. In fact, the certificate profile tends to be application-specific, for example, a certificate format for S/MIME and one for SSL is different in detail. Therefore it is practical that a CA should support multiple application-specific certificate profiles. In this model, a CA is required to be able to distinguish application types among certificate issuing requests. Then, we define certificate issuing request data including application types.

Second, in the practical point of view, determining an appropriate certificate issuing procedure is very important in PKI. Of course, there are several procedures appropriate for each authentication level. A typical procedure is as follows: get a temporary password after an authentication step, create key pairs, and have a certificate issued using the password in online. In the model, CA is

required to manage password database, and confirms account and password on each certificate issuing. We define certificate issuing request data including account and password information.

Third, to make CA based applications available in the global networks, it is required to retrieve certificates or CRLs not only from local repository but also from other repositories. Furthermore, it is convenient for applications that retrieval is done by throwing only one query to neighbor CA, without specifying each repository corresponding to the issuer CA of the certificate. In this model, a CA is required to manage other CA's repository access point information and communicate each other. We define certificate retrieval protocols on supposition of CA hierarchy.

Though the certificate management protocol proposed in [PKIX-CMP] does not specify unique transport protocol, HTTP is assumed in this document. Now, World Wide Web (WWW) is ubiquitous, and almost all internet users can use it even if the site they belong to has a firewall against intruders. The WWW provides some useful facilities for PKI; such as information caching at both proxy servers and client softwares, a secure transport layer service for confidentiality, a request forwarding which can be used for CA-CA communications, and easy manipulating message format.

2. Basic Definition, Requirements and Assumptions

2.1 CA model

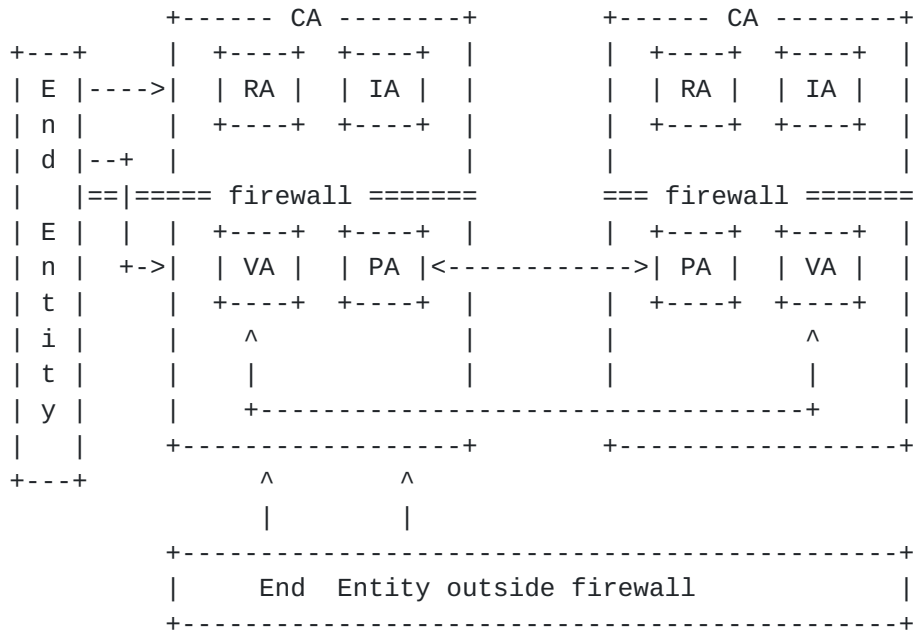

Figure 1: CA model

In this model, it is assumed that a CA consists of four authorities: Registration Authority (RA), Issuing Authority (IA), Publishing Authority (PA), and Validation Authority (VA). This document defines a protocol between RA and end entitity, a protocol between PA and end entity, and a protocol between VA and end entity. It also defines PA-PA protocols and a VA-VA protocol. But CA internal protocols, such as RA-IA, IA-PA, or IA-VA is out of scope in this document.

The End Entity in the Figure 1 is a PKI application program rather than a user.

2.1.1 Registration Authority

RA is the authority that confirms if the end entity requesting service such as certificate issuing, revoking, or updating is the real one and then decides to accept the request. For example, RA manages accounts and passwords database. Each account may be bound to a user or another RA. Passwords are desired to be disposable to prevent replay attack. How to distribute these accounts and passwords depends on the security level to be required.

If the certificate issuing request is acceptable, it is sent to IA. There may be multiple RAs in a CA.

2.1.2 Issuing Authority

IA is the authority that issues X.509 certificates conformed to some application-specific certificate profile and also issues X.509 CRLs. IA manages a private key for issuing certificates and CRLs, and certificate profile information.

For example, each application-specific certificate profile includes information below:

- application type name
- public key algorithm
- signature algorithm
- validity of certificate
- X.509 version
- mandatory attributes in subject of the certificate
- (in case of X.509 version3) extension fields to be used

After issuing a certificate, it is sent to PA. After issuing a CRL, it is sent to both PA and VA. Note that the certificate including the public key of IA is so called the "CA certificate".

2.1.3 Publishing Authority

PA is the authority that stores and distributes X.509 certificates and CRLs, that is, certificates and CRLs repository in PKI model defined in PKIX WG. PA manages certificates and CRLs database.

To retrieve certificates or CRLs issued by other CA, PA communicates with other PAs, and PA usually is placed outside organization's firewalls. Proposing web-based protocol enables end entities inside a firewall access to PA outside the firewall with existing web proxy.

To communicate with other PAs for certificate retrieval, simple restricted hierarchical certificate infrastructure is assumed. Figure 2 shows an example of hierarchy of PAs, where RootPA has two subordinate PAs, PA1 and PA2, having two certificates Cert1 and Cert2, respectively. Each of PA1 and PA2 has the database which at least consists of three fields; the own administrative realm, the RootPA's realm, and access point to the RootPA. The realm indicates e-mail domains and Distinguished Name patterns in the certificates managed by a PA. On the other hand, the RootPA has the database of the own realm, the subordinate PA's administrative realm and access point to each subordinate PA.

If an end entity throws a query to PA2 asking for Cert1 by some e-mail address, PA2 checks its own database and finds the domain of the e-mail address is not included in the own realm. PA2 then forwards the query to the parent PA, RootPA. RootPA checks its own database and finds the domain of the e-mail address in the query is included in the own realm. Next RootPA examines each realm of the two subordinate PAs, PA1 and PA2, and finds the realm of PA1 includes the domain of the e-mail address in the query. After that, there are two methods to get Cert1. One is that RootPA forwards the query to PA1, receives Cert1 from PA1, returns Cert1 to PA2, and PA2 returns Cert1 to the end entity. The other is that RootPA just returns the access point of PA1 to PA2, PA2 forwards the query to PA1, PA2 gets Cert1 from PA1, and PA2 returns Cert1 to the end entity. Thus the end entity can get Cert1 without knowing the access point to PA1. Section 3.6.4 describes the way the end entity gets Cert1 in detail.

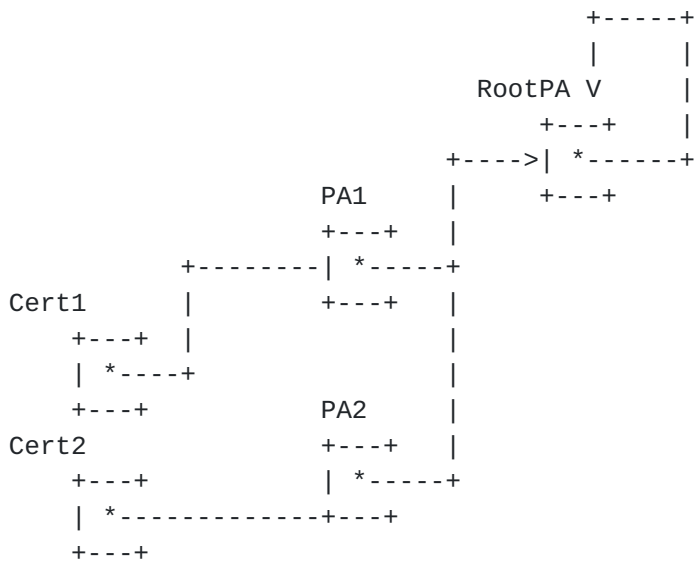

Figure 2: Example of PAs hierarchy

To communicate with PAs for CRL retrieval, it is assumed that an end entity already has some certificate to be examined and CRL distribution points are included in certificate contents. So, if the end entity throws the query to the PA2 for CRL of PA1 in Figure 2, PA2 gets CRL distribution point from the given certificate, and accesses to the point. In this example, the end entity may directly access to the PA1 to get CRL, but this document provides PA-PA communication for convenience.

2.1.4 Validation Authority

VA is the authority that decides if a specific certificate is valid or not, similar to the responder supporting Online Certificate Status Protocol (OCSP) [PKIX-OCSP]. VA is useful when it is anticipated that CRL size is too big and each application cannot scan the CRL quickly to examine validity of a certificate. VA has own CA's CRLs but does not have parent CA's CRLs. It is not VAs but end entities that are responsible for confirming hierarchical validation paths.

When VA replies to end entity's query without secure transport, it is basically required to sign the reply to prevent forgery. So, VA may have private key. To ask validity of a certificate issued by other CA, VA communicates with other VAs. Considering that a query to VA comes from other VAs or end entities outside organizations, VA is placed outside of a firewall. And considering that a reply should be returned quickly, the private key to sign replies is placed outside of a firewall and used automatically in online. If the private key of

VA is same with that of IA, security level of the CA depends on that of VA, no matter how the private key of IA is protected inside a firewall. Then it is preferable that private key of VA is distinct from that of IA.

It is assumed that end entity already has some certificate to be examined and VA access points are included in certificate contents. If the end entity throws the query to the neighbor VA, say VA1, to check the validity of certificate issued by other CA, the VA1 gets access point from the given certificate and forwards the query to the access point, VA2. In this case, the end entity has direct access to the VA2, but this document provides such a VA-VA communication for convenience.

2.2 Requirement for PKI Application

The PKI application typically needs access to CA in the following four cases;

1. certificate issuing request.
   As one of installation steps of application, a new certificate is required to be issued.

2. certificate retrieval.
   For example, a sender of secure message wants to retrieve the recipient's public key with which the message is to be encrypted. Further, the recipient of secure message wants to retrieve certificate of CA which issued the sender's certificate.

3. certificate verification.
   The recipient of secure message checks if the sender's certificate is not revoked by examining the corresponding CRL or asking the CA directly.

4. certificate and CRL publication.
   Soon after a certificate is issued by a CA, the new certificate shall be got access for anybody who wants it.

2.3 Requirement for X.509 Version 3 Certificate and Extensions

This proposal supposes that subset of X.509 Version 3 is used to form public key certificates. According to [PKIX-PROF], the subject and issuer names in X.509 (Version 1) may be an empty sequence and subjectAltName and issuerAltName extensions (Version 3) shall be specified instead of the field. Even if the subject and issuer names are specified, the subjectAltName shall be given as an identification of certificate retrieval.

Most PKI applications require many kinds of information about certificate including policy information, CRL, and VA information. In [PKIX-PROF], several access methods are defined for each kind of information. However, it is not so often case that a certain application has multiple access methods to PKI. Therefore, this document assumes that PKI application has an uniform access method of HTTP for the simplification of PKI protocol.

If this proposal is used, a standard certificate must specify

- authorityInfoAccess
- cRLDistributionPoints

shall specify

- subjectAltName,
- issuerAltName,

may specify

- authorityKeyIdentifier,
- subjectKeyIdentifier,
- keyUsage,
- privateKeyUsagePeriod,
- certificagtePolicies,
- basicConstraints,
- nameConstraints,
- policyConstraints,

etc.

2.4 Requirement for X.509 CRL and Extensions

This proposal supposes that subset of X.509 Version 1 and Version 2 are used to form CRL.

If the X.509 Version 2 CRL is used, a standard CRL shall specify

- reasonCode

may specify

- CRL Number

etc.

3. Protocol Specification

3.1 transport protocol

This proposal assumes that either of a standard HTTP protocol or HTTPS protocol i.e. HTTP/SSL [SSL], is used as transport protocol. Thus, RA, IA, PA, and VA server can be implemented as a standard HTTP server which enables CGI facility.

3.2 request format

The request is sent with the POST method of the HTTP. Thus, the typical query is formatted as follows:

```
POST /cgi-bin/queryType HTTP/1.0
Content-length: xx

name1=value1&name2=value2&...&namen=valuen
```

where "queryType" is a type of query, and a pair of "name" and "value" are used to send PKI message. All pairs are encoded according to the rule defined in [HTTP].

(Note1) When the content length is computed, return code is treated as 2 length, which consists of CR and LF.

(Note2) When the value is encoded according to Base64 rule [Base64], end entity must substitute "+" code of Base64 with "%2b". And when the end entity sends the request on the telnet protocol, not sending directly from the socket, "=" code of Base64 must be also substituted with "%3d".

3.3 response format

In the response, application/x-pkixicap content-type is used and simply formatted as follows:

```
HTTP/1.0 200 OK
Date: Wednesday
MIME-version: 1.0
Content-type: application/x-pkixicap

queryType
statusCode statusMessage
INFORMATION
```

The "queryType" is the same as the type given in sending request and this shows the first line of response. Second line consists of "statusCode" and "statusMessage".

The "statusCode" indicates brief processing status category. The "statusCode" contains three digit codes. With at least one space,

the "statusMessage" follows and it contains the description of the "statusCode". "2xx" of "statusCode" indicates successful processing, and "3xx" of "statusCode" indicates error.

The "statusMessage" gives some hints on error handling, and it may have several message patterns for a "statusCode".

The third line, "INFORMATION" is interpreted in corresponding semantics. The syntax of "INFORMATION" depends on the query type, and will be defined in the later sections.

This simple response format is appropriate for application to interpret the response.

3.4 Type of Query

This document defines the following seven query types. Those names are used as "queryType" value in request and response format.

1. certreq	2. lookupreq	3. calookupreq
4. crlreq	5. verifyreq	6. updatereq
7. revokereq		

"certreq", "updatereq" and "revokereq" are requests to RA. "lookupreq", "calookupreq", and "crlreq" are requests to PA. And, "verifyreq" is a request to VA.

3.5 certificate issuing request type "certreq"

Type "certreq" is used for sending certificate issuing request to RA.

3.5.1 request

The certreq request shall be sent with the following pairs of name and value.

name	value
PKCS10	extended PKCS#10 [PKCS-10] data, Base64 encoded

The extended PKCS#10 format must include the following fields:

- Subject
- SubjectPublicKeyInfo
- account and password attributes for authentication of the end entity
- application type attribute to specify certificate profile

The extended PKCS#10 format may include the following field in optional:

- o extension fields attributes for the X.509 version3 extensions

  Extension fields attributes may be used if the value of the extension field should be specified by end entity itself, not CA policy.

Subject and SubjectPublicKeyInfo conform to standard PKCS#10. The rest four attributes, account, password, application type, and extension fields are defined as follows:

```
Account     ::= UnstructuredName
                - account
Password    ::= ChallengePassword
                - password
App         ::= PrintableString
                - application type
V3Extn      ::= Extensions
                - extension field
```

Note that attribute "UnstructuredName" and "ChallengePassword" are defined in PKCS#9 [PKCS-9]. Extensions are defined in [X.509] or also described in [PKIX-PROF]. Attribute "App" and "V3Extn" are originally defined and Appendix shows our local OID definitions for these two attributes.

3.5.2 response

The response consists of statusCode, statusMessage and INFORMATION. In "certreq" type, statusCode definition is as follows:

statusCode	meaning
200	successfully processed
301	request is incomplete
302	the certificate has already been issued
303	request is rejected
304	service is not available
305	(reserved)

If the statusCode is "200", the INFORMATION in response is the issued certificate encoded in Base64 format. Otherwise, INFORMATION is empty.

statusMessage is defined as follows:

statusCode	statusMessage
200	"accept your request"
301	"not PKCS10 format"
301	"signature verification failed"
301	"missing mandatory field (xxx)" where xxx is a name of missing field.
302	"detect duplicated DN"
303	"public key algorithm not supported "
303	"signature algorithm not supported "
303	"extension field (xxx) not supported " where xxx is OID, e.g. "551d11"
303	"application (xxx) not supported" where xxx is an application type name, e.g. "smime"
303	"authentication failed"
303	"can't issue cert anymore"
303	"access denied"
304	"service not available"

3.6 certificate retrieval type "lookupreq"

The "lookupreq" type is used for retrieving and searching certificate from PA.

Certificate is identified by either of the following name forms;

a. email address
b. Distinguished Name
c. Issuer and Serial Number

All name forms must be fully specified because a substring matching rule might violate a privacy issue when the PA is the outside of firewall.

The request and response format are described in the next clauses.

3.6.1 "lookupreq" with email address

3.6.1.1 request

The lookup query is sent with the following pairs of name and value.

name	value
EmailAddress	e-mail address string
TimePeriod	(optional) months and years string conforming to the following syntax YYYYMM[HH|HHMM|HHMMSS]
Latest	(optional) "1"
KeyUsage	(optional) one of the following strings "digitalSignature","nonRepudiation", "keyEncipherment","dataEncipherment", "keyAgreement",etc.

The "EmailAddress" pair is a mandatory for this type of request. "TimePeriod" may be used when the end entity wants some expired or revoked certificates. If "TimePeriod" is not specified, expired or revoked certificates are not searched. "Latest" may be used when the end entity wants the latest certificate even if the multiple certificates are hit. "KeyUsage" may be used if the end entity wants to get some certificate for a specific purpose.

3.6.1.2 response

The response consists of statusCode, statusMessage and INFORMATION. In "lookupreq" type, statusCode definition is as follows:

statusCode	meaning
200	successfully processed and uniquely retrieved.
201	successfully processed but got multiple hits
301	request is incomplete
303	request is rejected
304	service is not available

If the statusCode is "200", the INFORMATION in response format is the certificate encoded in Base64 format. If the statusCode is "201", the INFORMATION in response format is DN lists originally

defined in this document. Otherwise, INFORMATION is empty.

statusCode	statusMessage
200	"accept your request"
201	"found several certs"
301	"existed, but revoked"
303	"existed, but expired"
303	"no match"
303	"unknown CA"
303	"unknown mail domain"
303	"not correct input"
303	"hit too many cert" note that it depends on CA policy whether this message is used or not.
304	"service not available"

The first line of DN list shows the number of hits (more than 2), and the remain lines' syntax is defined as follows:

IssuerAndSerialNumberData + ":" + DistinguishedNameData Note that this syntax is represented in a single line.

IssuerAndSerialNumberData is IssuerAndSerialNumber data defined in PKCS#7 [PKCS-7], encoded in Base64 format. DistinguishedNameData is a string representation of Distinguished Names defined in [RFC1779].

The DN list example is shown here:

```
2
MIIDSzCCAw0CAQAwggEZMRAwDgYDVQQGEwcbJEJGfEtcMREwDw:
CN=Christian Huitema, O=INRIA, C=FR
ME8wSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgEE:CN=ALPHA Tom, EM=alpha
@abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div., O=NEC
Corporation, C=JP
```

3.6.2 "lookupreq" with Distinguished Name

3.6.2.1 request

The lookup query is sent with the following pairs of name and value.

name	value
c	Country
o	Organization
cn	Common name
ou1	(optional) Organizational unit
ou2	(optional) Organizational unit 2
ou3	(optional) Organizational unit 3
ou4	(optional) Organizational unit 4
sopn	(optional) StateOrProvinceName
Locality	(optional) Locality
StrAddr	(optional) StreetAddress
pcode	(optional) postal code
phnum	(optional) phone number
title	(optional) title
TimePeriod	(optional) months and years conforming to the following syntax YYYYMM[HH|HHMM|HHMMSS]

The pairs of "c", "o", "cn" are mandatory. When the specification is incomplete, the PA may reject it for privacy issue or accept it as substring matching.

3.6.2.2 response

The response is same with the one defined in the previous clause 3.6.1.2 except that "303 unknown mail domain" is replaced below:

statusCode	statusMessage
303	"unknown DN"

3.6.3 "lookupreq" with Issuer and Serial Number

3.6.3.1 request

The lookup query is sent with the following pairs of name and value.

name	value
IASN	IssuerAndSerialNumber data defined in PKCS#7 [PKCS-7], encoded in Base64 format

3.6.3.2 response

The response is same with the one defined in clause 3.6.1.2 except "201" response, because IASN should specify unique certificate.

3.6.4 PA-PA protocol in "lookupreq"

A PA server may forward a request to another PA server when it does not have sufficient information to response to the request. If a PA which does not support PA-PA protocol should response the statusCode "303" with the statusMessage "unknown CA".

Suppose that there are PA1, PA2 and RootPA, and PA1 has a request for retrieving a certificate from PA2. The PA1 and the PA2 does not have their access point but the access point to RootPA. There are two possibilities for PA1 to get access to PA2 (Figure 3).

- Model 1. [referral] PA1 sends the request to RootPA (1), which then replies to PA1 with the access point to PA2 (2). PA1 forwards the request to PA2 again (3), and finally PA1 gets the information from PA2.

- Model 2. [chaining] PA1 sends the request to RootPA (1), which redirects it to PA2 on behalf of PA1 (2). PA2 answers to Root PA (3), which forwards it to PA1 (4).

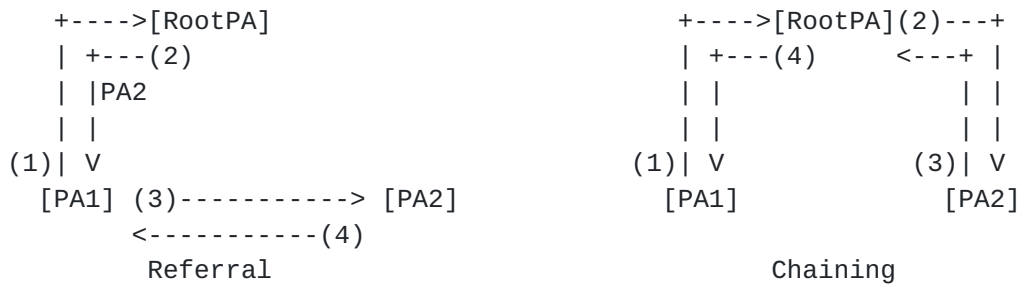

Figure 3. PA-PA models

To implement these two models, each PA must have information about its own realm in order to determine if the "lookupreq" request should be forwarded. Since the request includes an e-mail address or part of a Distinguished Name in a certificate, if the realm is represented with e-mail address domain lists or Distinguished Name patterns, PA can easily decide to answer directly or to forward.

Addition to its own realm information, PA1 and PA2 must know the URL of RootPA server to forward the request. In general, there may be multiple root PAs for a PA. RootPA, which does not necessarily correspond to the root of a CA hierarchy, must know the URL of PA1 and PA2 as subordinate PAs to forward the request (in the chaining model) or return the location information (in the referral model).

In the Figure 3, the depth of the virtual PA hierarchy is only one and there are only a root PA and leaf PAs. However, there could be deeper PA hierarchies and mediate PAs can exist. A mediate PA acts like a router, i.e. it forwards a request to a parent PA or a child PA according to its own realm information. Then each PA must know which topology type it belongs to; top, leaf or mediate.

In this example, RootPA manages the database in the following form:

```
Me:TOP:RootPA.lll.or.jp:aaa.bb.co.jp, ccc.bb.co.jp, foo.ff.co.jp,
   bar.ff.co.jp
Child:LEAF:PA1.bb.co.jp:aaa.bb.co.jp, ccc.bb.co.jp
Child:LEAF:PA2.ff.co.jp:foo.ff.co.jp, bar.ff.co.jp
```

Each line in the database consists of ":" separated fields.

If the first field of each line is "Me", then the line contains the information about the owner PA of the database. If the first field of each line is "Child", then the line contains the information about a child PA for the owner PA of the database. If the first field of each line is "Parent", then the line contains the information about a parent PA for the owner PA of the database.

If the second field of each line is "TOP", then the PA is the root in a PA hierarchy. If the second field of each line is "MEDIATE", then the PA is mediate in a PA hierarchy. If the second field of each line is "LEAF", then the PA is a leaf in a PA hierarchy.

The third field of each line shows the PA server's name and the

fourth field of each line shows a realm with e-mail domain lists, separated with commmas. For simplicity, the DN pattern lists are omitted in this database.

In the example above, the first line indicates that RootPA is a TOP in the PA tree, the hostname is "RootPA.lll.or.jp", and the realm includes the four e-mail domains; "aaa.bb.co.jp", "ccc.bb.co.jp", "foo.ff.co.jp" and "bar.ff.co.jp". The next two lines indicates that RootPA has two child PAs, both PAs are a LEAF in the PA tree; one is "PA1.bb.co.jp" and another is "PA2.ff.co.jp". Note that an union of these two child PA realms makes RootPA realm.

Similarly, PA1 manages the database in the following form:

```
Me:LEAF:PA1.bb.co.jp:aaa.bb.co.jp, ccc.bb.co.jp
Parent:TOP:RootPA.lll.or.jp:aaa.bb.co.jp, ccc.bb.co.jp, foo.ff.co.jp,
    bar.ff.co.jp
```

In this example, the first line indicates that PA1 is a LEAF in the PA tree, the hostname is "PA1.bb.co.jp", and the realm includes the two e-mail domains; "aaa.bb.co.jp" and "cc.bb.co.jp". The second line indicates that PA1 has a parent PA, named with "RootPA.lll.or.jp", and the parent PA realm includes the four e-mail domains; "aaa.bb.co.jp", "ccc.bb.co.jp", "foo.ff.co.jp" and "bar.ff.co.jp".

If the realm of RootPA is changed, the all subordinate PAs, PA1 and PA2 must update the realm field of "Parent" line in the database. To solve this problem, special server such as SecureDNS is required to manage the correspondences between a realm of a PA and an access point to the PA server, such as URL. Thus, each PA can examine the access point to the target PA by asking to the special server instead of RootPA. Even if the topology of the PA tree is changed, only the special server's database is to be updated.

All the PAs must manage these databases consistently to work this protocol correctly. If a database is incorrect, a forwarding loop happens. To detect such a loop, we propose a solution to use a "query_path" variable in a "lookup" request.

name	value
query_path	request forwarder's list

All the PAs forwarding a "lookupreq" request must check the "query_path" and add the own DN to the variable in the request before

forwarding. The value is added in reverse-order of the query. For example, a "lookupreq" request is forwarded from the PA1 to the Root PA, the "query_path" in the request becomes "PA1". Then the request is forwarded from the Root PA to the PA2, the "query_path" value becomes "Root PA--PA1" In case of the chaining model, it is in-loop if a PA finds the top of the "query_path" value is same with the next PA to be forwarded. In case of the referral model, it is in-loop if a PA finds the top of the "query_path" value is own DN and the next PA to be forwarded is also the own PA.

3.6.4.1 Referral Model

To redirect a request to another PA server, the root PA responds to the requester with the following format.

statusCode	statusMessage	INFORMATION
202	"ask other CA"	URLs

The example of URL is "http://xxx.yy:zz/cgi-bin/lookupreq", which provides information where the query should be sent next. A root PA may include multiple URLs as INFORMATION. In this case, a URL must be one-line represented.

The root PA must respond with either correct PA location or error message to mean that there is no certificate.

Each round trip time is short, but the neighbor PA for a requestor has to send the same query to several servers.

3.6.4.2 Chaining Model

To implement chaining model, the CGI script of the root PA produces an extra CGI message before it responds to the request originator.

The request originator, PA1 or PKI application, does not have to send request multiple times, but have to wait longer time than that of referral model. According to [Mine], the estimated total round trip time is longer than that of the referral model. Multiple requests in a short time makes load of the root PA server heavy. But, there are two advantages in the chaining model under some practical situations. One is that if the network speed between the PA1 and the PA2 is slower than that between the PA2 and the root PA, chaining model is useful. Another is that if the organization which the PA1 belongs to has a policy that the PA1 should limit accesses from external point, this chaining model enables the PA1 communicate only with the root PA and the access controll can be easy. The chaining model is a firewall-friendly model.

In our experiment, ten requests a minute makes no significant difference in round trip time between these 2 models.

3.7 CA certificate retrieval type "calookupreq"

The "calookupreq" type is used for retrieving and searching the CA certificate from PA.

3.7.1 request

The calookupreq query is sent with the following pairs of name and value.

name	value
cert	(optional) certificate encoded in Base64 format
TimePeriod	(optional) months and years string conforming to the following syntax YYYYMM[HH|HHMM|HHMMSS]
KeyUsage	(optional) one of the following strings "keyCertSign", "cRLSign", etc.

All the pairs are optional in "calookupreq". If the "cert" is not specified, it is assumed that the PA is required to response the certificate of own CA. "TimePeriod" may be used when the end entity wants some expired or revoked CA certificates. If "TimePeriod" is not specified, expired or revoked certificates are not searched. "KeyUsage" may be used if the end entity wants to get some CA certificate for a specific purpose.

3.7.2 response

The response consists of statusCode, statusMessage and INFORMATION. In "calookupreq" type, statusCode definition is as follows:

statusCode	meaning
200	successfully processed
303	request is rejected
304	service is not available

If the status code is "200", the CA certificate encoded in Base64 format is appeared as INFORMATION in response format. Otherwise, INFORMATION field is empty.

statusMessage is defined as follows:

statusCode	statusMessage
200	"accept your request"
303	"not certificate format"
303	"unknown CA"
303	"the URL not found"
303	"AuthorityInfoAccess not included"
304	"timeout"
304	"service not available"

3.7.3 PA-PA protocol in "calookupreq"

A PA server may forward a request to another PA server when it does not have sufficient information to response to the request. If a PA which does not support PA-PA protocol should response the statusCode "303" with the statusMessage "unknown CA".

Suppose that there are PA1 corresponding to CA1, and PA2 corresponding to CA2, and PA1 has a request for CA2 certificate from PA2. PA1 gets the URL to PA2 from the AuthorityInfoAccess field in the certificate which PA1 received from the requester. Then PA1 forwards the request to PA2, receiving the response from PA2, and returns the response to the requester.

3.8 CRL retrieval type "crlreq"

The "crlreq" type is used for retrieving a CRL from PA.

3.8.1 request

The crlreq query is sent with the following pairs of name and value.

name	value
cert	(optional) certificate encoded in Base64 format

reason	(optional) specifying CRL reason code one of the following strings "keyCompromise","cACompromise", "affiliationChanged","superseded", "cessationOfOperation", "certificateHold", etc.

All the pairs are optional in crlreq. If the "cert" is not specified, it is assumed that the PA is required to response the CRL of own CA. The "reason" may be used when the end entity wants some reason-specific CRL.

3.8.2 response

The response consists of statusCode, statusMessage and INFORMATION. In "crlreq" type, statusCode definition is as follows:

statusCode	meaning
200	successfully processed
201	successfully processed, but CRL doesn't exist
202	successfully processed, but multiple CRLs exist
303	request is rejected
304	service is not available

If the status code is "200", INFORMATION in response format is the CRL encoded in Base64 format. If the status code is "202", INFORMATION in response format is the reason list strings which shows the CRLreasoncode of each CRL, so that the end entity could specify the "reason" in the next request. Otherwise, INFORMATION in response format is empty.

The reason list is defined as comma separated strings. Each string is the same with "reason" value in the request. The order is not considered.

["keyCompromise"][,]["caCompromise"][,]["affiliationChanged"][,]
["superseded"][,]["cessationOfOperation"][,]["certificateHold"]

The statusMessage is defined as follows:

statusCode	statusMessage
200	"accept your request"
201	"not issued"

```
202          "specify CRL reason"

303          "not certificate format"

303          "unknown CA"

303          "CRLDistributionPoints not included"

303          "the URL not found"

304          "timeout"

304          "service not available"
```

3.8.3 PA-PA protocol in "crlreq"

A PA server may forward a request to other PA server when it does not have sufficient information to response to the request. If a PA which does not support PA-PA protocol should response the statusCode "303" with the statusMessage "unknown CA".

Suppose that there are PA1 corresponding to CA1, and PA2 corresponding to CA2, and PA1 has a request for CA2 CRL from PA2. PA1 gets the URL to PA2 from the cRLDistributionPoints field in the certificate which PA1 received from the requester. Then PA1 forwards the request to PA2, receiving a response from PA2, and returns the response to the requester.

3.9 Certificate Verification type "verifyreq"

The "verifyreq" type is used for validation check of certificate. This document does not support path validation.

3.9.1 request

The verifyreq request shall be sent with the following pairs of name and value.

name	value
cert	certificate encoded in Base64 format
resptype	response type string, "1" or "2" "1" requires the response not to be signed, "2" requires the response to be signed.

All these pairs are mandatory in verifyreq. The "resptype" is used

for specifying whether the response is required to be signed ("2") or not ("1"). The end entity should decide the response type depending on the transport or network environment.

The response is defined separately according to the "resptype".

3.9.2 response

3.9.2.1 the response in resptype "1" (not to be signed)

The response consists of statusCode, statusMessage and INFORMATION. The statusCode definition is as follows:

statusCode	meaning
200	successfully processed and valid
201	successfully processed and invalid
202	successfully processed and revoked
203	successfully processed and expired
204	successfully processed and hold
303	request is rejected
304	service is not available

If the statusCode is "202", the INFORMATION in the response format is the UTCtime string of revoked time. Otherwise, the INFORMATION in the response format is empty.

The statusMessage is defined as follows:

statusCode	statusMessage
200	"valid"
201	"invalid"
202	"revoked"
203	"expired"
204	"hold"
303	"not certificate format"
303	"the URL not found"
303	"unknown response type"

```
          303          "unknown CA"

          303          "AuthorityInfoAccess not included"

          303          "signature verification failed"

          304          "timeout"

          304          "service not available"
```

3.9.2.2 the response in resptype "2" (to be signed)

The response consists of statusCode, statusMessage and INFORMATION. The statusCode definition is as follows:

statusCode	meaning
200	successfully processed
303	request is rejected
304	service is not available

If the statusCode is "200", the INFORMATION in the response format is the digitally signed verification result defined originally in ASN.1, encoded in Base64 format. Otherwise, the INFORMATION in the response format is empty.

The signed verification result format is defined as follows:

```
 Reply ::= SEQUENCE { SIGNED SET {
     version [0] INTEGER default 0,
             SEQUENCE {
     issuer  [1] Name,
     serialNumber [2] Integer } OPTIONAL,
     verificationResult [3] VerificationResult,
     validationTime  [4] GeneralizedTime
     revokedReason   [5] RevokedReason OPTIONAL,
     revokedOrHoldTime [6] GeneralizedTime OPTIONAL,
     holdExpirationTime [7] GeneralizedTime OPTIONAL,
     holdInstructionCode [8] OBJECT IDENTIFIER OPTIONAL}
     CertificationPath OPTIONAL }

     VerificationResult ::= ENUMERATE {
      notRevoked(0), revoked(1), expired(2), hold(3) }

     RevokedReason ::= ENUMERATE {
       unspecified (0),
       keyCompromise (1),
```

```
          caCompromise (2),
          affiliationChanged (3),
          superseded (4),
          cessationOfOperation (5) }

       holdInstructionCode ::= OBJECT IDENTIFIER

          holdInstruction    OBJECT IDENTIFIER ::=
                      { iso(1) member-body(2) us(840) x9-57(10040) 2 }

          id-holdinstruction-none   OBJECT IDENTIFIER ::=
                      {holdInstruction 1}
          id-holdinstruction-callissuer
                                    OBJECT IDENTIFIER ::=
                      {holdInstruction 2}
          id-holdinstruction-reject OBJECT IDENTIFIER ::=
                      {holdInstruction 3}
```

The statusMessage is defined as follows:

statusCode	statusMessage
200	"accept your request"
303	"not certificate format"
303	"the URL not found"
303	"unknown response type"
303	"unknown CA"
303	"AuthorityInfoAccess not included"
304	"timeout"
304	"service not available"

3.9.3 VA-VA protocol in "verifyreq"

A VA server may forward a request to other VA server when it does not have sufficient information to response to the request. If a VA which does not support VA-VA protocol should response the statusCode "303" with the statusMessage "unknown CA".

Suppose that there are VA1 corresponding to CA1, and VA2

corresponding to CA2, and VA1 has a request for CA2 CRL from VA2. VA1 gets the URL to VA2 from the authorityInfoAccess field in the certificate which VA1 received
from the requester. Then VA1 forwards the request to VA2, receiving a response from VA2, and returns the response to the requester.

3.10 certificate update type "updatereq"

The "updatereq" is a request to update a certificate.

[TBD]

3.11 certificate revocation type "revokereq"

The "revokereq" is a request to revoke a certificate. To prevent malicious PKI user from revoking other's certificate, this request should be sent with a proof of possession of the secret key. The simplest way is to use conventional application that supports digital signature.

[TBD]

3.12 Correspondence to preceding PKI draft

This document corresponds to PKI management protocol defined in [PKIX-CMP],[PKIX-OCSP],[PKIX-OPP],[PKIX-LDAP]. Table 1 shows the correspondence.

Table 1. Correspondence of methods

ICAP method	PKI method
certreq	CMP(PKCS #10 Cert. Req), WebCAP(MKCERT)
lookupreq	OPP(FTP and HTTP),OPP(LDAP), WebCAP(GETCERT)
calookupreq	OPP(LDAP)
crlreq	OPP(LDAP), WebCAP(GETCERT)
verifyreq	OCSP, WebCAP(VRFYCERT)
revokereq	CMP(Revocation Request), WebCAP(RMCERT)
updatereq	

4. Security Considerations

4.1 Confidentiality of transaction

To prevent message from being eavesdropped, secure communication channel such as SSL(TLS) shall be used. Especially, initial

registration process is critical to eavesdropping. Since user authentication is checked with account and password, a client software is not required to have its own certificate. Under this assumption, PKI message protection proposed in [PKIX-CMP] need not here.

4.2 Non-Repudiation

The verifyreq supports the time to be checked and digitally signed response. This can avoid a message sender from denying the message. To enable this service, any PA must manage all certificates which it has already been issued, including revoked certificates.

4.3 Privacy

In the lookupreq, the support of substring matching facility may distribute private information to outsiders, and thereby may be used for sending an advertisement via email.

5. Examples

5.1 certreq

```
%telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
POST /cgi-bin/certreq HTTP/1.0
Content-length: 1137

PKCS10=MIIDPzCCAwYCAQAwggEUMQswCQYDVQQGEwJKUDERMA8GA1UECBMIWW9rb2hh
bWExEzARBgNVBAcTClRvdHN1a2Eta3UxFDASBgNVBAkTC1RvdHN1a2EtY2hvMS8wLQY
DVQQKEyZIaXRhY2hpIFNvZnR3YXJlIEVuZ2luZWVyaW5nIENvLiwgTHRkLjEbMBkGA1
UECxMSRGF0YSBDb21tdW5pY2F0aW9uMQ0wCwYDVQQMEwRISVJBMQwwCgYDVQQREwMyN
DQxFTATBgNVBBQTDDAxMjAtNDY4ODEyMTEYMBYGA1UEAxMPSGl0b3NoaSBLdW1hZ2Fp
MSswKQYJKoZIhvcNAQkBExxoaXRvc2hpQG9yaS5oaXRhY2hpLXNrLmNvLmpwMIHXMIG
pBgoqgwiGjScHAQUBMIGaAgEBMCMGCiqDCIaNJwcBBgECFQD//////////mvyqjt91
qfEHYKrzAsBBQAAAAAAAAAAAAAAAAAAAAAAAAAAQUAAAAAAAAAAAAAAAAAAAAAAAAA
MAEKAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAABACFQD/////
//////mvyqjt91qfEHYKrwIBAAMpAPI7g4TgW3RXWs2beo0BFCGHgNaQp//etDYwV9H
F3DlMy5u/ZP3CrTmgggENMBQGCSqGSIb3DQEJAjEHFgVTTUlNRTAUBgkqhkiG9w0BCQ
cxBxMFU01JTUUwFAYJqqqqqqqqqqqqMQcTBVBFUE9QMIHIBgm7u7u7u7u7u7sxgbowg
bcwgbQGBv///////wEBAASBpjCBozCBgjAvBgNVBAoxKBMmSGl0YWNoaSBTb2Z0d2Fy
ZSBFbmdpbmVlcmluZyBDby4sIEx0ZC4wEwYDVQQHMQwTClRvdHN1a2Eta3UwDQYDVQQ
MMQYTBEhJUkEwKwYJKoZIhvcNAQkBMR4WHGhpdG9zaGlAb3JpLmhpdGFjaGktc2suY2
8uanAXDTAwMDAwMDAwMDAwMFoXDTAwMDAwMDAwMDAwMFowCQYF/////wQFAAMokGUe2
```

```
   gyH92D5W7J/ms8119ibcQlXXU54Rtne9GEh15462oYwqnAraw%3d%3d

   HTTP/1.0 200 Document follows
   Date: Thu, 25 Sep 1997 10:50:46 GMT
   Server: NCSA/1.5.1
   Content-type: application/x-pkixicap

   certreq
   200 accept your request
    MIIENzCCA9SgAwIBAgIBJjAOBgoqgwiGjScHAQIHBQAwTTELMAkGA1UEBhMCSlAx
    DDAKBgNVBAoTA05FQzEwMC4GA1UECxMnTmV0d29ya2luZyBTeXN0ZW1zIExhYnMg
    RXhwZXJpbWVudGFsIENBMB4XDTk3MDkyNTEwNTA0NFoXDTk3MTIyNDEwNTA0NFow
    gf0xCzAJBgNVBAYTAkpQMREwDwYDVQQIEwhZb2tvaGFtYTETMBEGA1UEBxMKVG90
    c3VrYS1rdTEUMBIGA1UECRMLVG90c3VrYS1jaG8xDDAKBgNVBBETAzI0NDEvMC0G
    A1UEChMmSGl0YWNoaSBTb2Z0d2FyZSBFbmdpbmVlcmluZyBDby4sIEx0ZC4xGzAZ
    BgNVBAsTEkRhdGEgQ29tbXVuaWNhdGlvbjEYMBYGA1UEAxMPSGl0b3NoaSBLdW1h
    Z2FpMQ0wCwYDVQQMEwRISVJBMSswKQYJKoZIhvcNAQkBFhxoaXRvc2hpQG9yaS5o
    aXRhY2hpLXNrLmNvLmpwMIHXMIGpBgoqgwiGjScHAQUBMIGaAgEBMCMGCiqDCIaN
    JwcBBgECFQD//////////mvyqjt91qfEHYKrzAsBBQAAAAAAAAAAAAAAAAAAAA
    AAAAAAQUAAAAAAAAAAAAAAAAAAAAAAAAAMAEKAAAAAAAAAAAAAAAAAAAAAAAAAE
    AAAAAAAAAAAAAAAAAAAAAAAAABACFQD//////////mvyqjt91qfEHYKrwIBAAMp
    API7g4TgW3RXWs2beo0BFCGHgNaQp//etDYwV9HF3DlMy5u/ZP3CrTmjggFvMIIB
    azAPBgNVHRMBAQAEBTADAQEAMGcGBisGAQUDAgEBAARaMFigK4YpaHR0cDovL3d3
    dy5teWNhLmNvLmpwL2NnaS1iaW4vY2Fsb29rdXByZXGhKYYnaHR0cDovL3d3dy5t
    eWNhLmNvLmpwL2NnaS1iaW4vdmVyaWZ5cmVxMIG0Bgb///////8BAQAEgaYwgaMw
    gYIwLwYDVQQKMSgTJkhpdGFjaGkgU29mdHdhcmUgRW5naW5lZXJpbmcgQ28uLCBM
    dGQuMBMGA1UEBzEMEwpUb3RzdWthLWt1MA0GA1UEDDEGEwRISVJBMCsGCSqGSIb3
    DQEJATEeFhxoaXRvc2hpQG9yaS5oaXRhY2hpLXNrLmNvLmpwFw05NzA5MjUxMDUw
    NDRaFw05NzEwMjUxMDUwNDRaMDgGA1UdHwEBAAQuMCwwKqAooCaGJGh0dHA6Ly93
    d3cubXljYS5jby5qcC9jZ2ktYmluL2NybHJlcTAOBgoqgwiGjScHAQIHBQADTQAp
    EQJUCpr0S23/YhvkPKTVblhX1YQ60/Dy+5xiB9zxvdG6RsCZ9Zd58Eh8HKUSbpnm
    AQFx37tMe5jXXT0VyU4HyIdTh17L2u27uZtp

   Connection closed by foreign host.
```

5.2 lookupreq

5.2.1 lookupreq with e-mail address

5.2.1.1 in case of multiple hits

```
   Note that the result line is folded in this example.

   % telnet cahost1 80
   Trying 123.16.5.41 ...
   Connected to cahost1.
   Escape character is '^]'.
   POST /cgi-bin/lookupreq HTTP/1.0
```

```
   Content-length: 32

   EmailAddress=alpha@abc.nec.co.jp
   HTTP/1.1 200 OK
   Date: Sat, 25 Oct 1997 09:30:01 GMT
   Server: Apache/1.2.1
   Connection: close
   Content-Type: application/x-pkixicap

   lookupreq
   201 found several certs
   5
   ME8wSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
   1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgED:CN=ALPHA Tom,
   EM=alpha@abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div.,
   O=NEC Corporation, C=JP
   ME8wSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
   1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgEE:CN=ALPHA Tom, EM=alpha
   @abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div., O=NEC
   Corporation, C=JP
   MFEwSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
   1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgMAgAA=:CN=ALPHA Tom, EM=
   alpha@abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div.,
   O=NEC Corporation, C=JP
   MFIwSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
   1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgQAgAAA:CN=ALPHA Tom, EM=
   alpha@abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div.,
   O=NEC Corporation, C=JP
   MFMwSjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA
   1UECxMYTmV0bGFiIEV4cGVyaW1lbnRhbCAxMDI0AgUAgAAAAA==:CN=ALPHA Tom,
   EM=alpha@abc.nec.co.jp, rpEM=alpha@abc.nec.co.jp, ou=Internet Div.
   , O=NEC Corporation, C=JP

   Connection closed by foreign host.
```

5.2.1.2 in case of using Latest option

```
   % telnet cahost1 80
   Trying 123.16.5.41 ...
   Connected to cahost1.
   Escape character is '^]'.
   POST /cgi-bin/lookupreq HTTP/1.0
   Content-length: 41

   EmailAddress=alpha@abc.nec.co.jp&Latest=1
   HTTP/1.1 200 OK
   Date: Sat, 25 Oct 1997 09:34:17 GMT
   Server: Apache/1.2.1
```

```
Connection: close
Content-Type: application/x-pkixicap

lookupreq
200 accept your request
 MIIDmTCCA1qgAwIBAgIFAIAAAAAwDgYKKoMIgZxfCwEEAQUAMEoxCzAJBgNVBAYT
 AkpQMRgwFgYDVQQKEw9ORUMgQ29ycG9yYXRpb24xITAfBgNVBAsTGE5ldGxhYiBF
 eHBlcmltZW50YWwgMTAyNDAeFw05NzEwMjQxMDM0NDdaFw05ODAxMjIxMDM0NDda
 MIGsMQswCQYDVQQGEwJKUDEYMBYGA1UEChMPTkVDIENvcnBvcmF0aW9uMSAwHgYD
 VQQLExdOZXR3b3JraW5nIFN5c3RlbXMgTGFiczEWMBQGA1UECxMNSW50ZXJuZXQg
 RGl2LjESMBAGA1UEAxMJQUxQSEEgVG9tMREwDwYDVQQMEwhFbmdpbmVlcjEiMCAG
 CSqGSIb3DQEJARYTYWxwaGFAYWJjLm5lYy5jby5qcDCB1zCBqQYKKoMIho0nBwEF
 ATCBmgIBATAjBgoqgwiGjScHAQYBAhUA//////////5r8qo7fdanxB2Cq8wLAQU
 AAAAAAAAAAAAAAAAAAAAAAAAAAEFAAAAAAAAAAAAAAAAAAAAAAAAADABCgAAAAA
 AAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAQAhUA//////////5
 r8qo7fdanxB2Cq8CAQADKQAqhr1NSXL41WmWPilsNHPU7QqkTXou5PE9BggsfFxy
 h4JOt5TS9uU3o4IBRTCCAUEwDwYDVR0TAQEABAUwAwEBADBsBgsqgwiBnF8LAwEd
 AgEBAARaMFigK4YpaHR0cDovL3d3dy5teWNhLmNvLmpwL2NnaS1iaW4vY2Fsb29r
 dXByZXGhKYYnaHR0cDovL3d3dy5teWNhLmNvLmpwL2NnaS1iaW4vdmVyaWZ5cmVx
 MIGFBgsqgwiBnF8LAwEdAQEBAARzMHEwUTAYBgNVBAoxERMPTkVDIENvcnBvcmF0
 aW9uMBEGA1UEDDEKEwhFbmdpbmVlcjAiBgkqhkiG9w0BCQExFRYTYWxwaGFAYWJj
 Lm5lYy5jby5qcBcNOTcxMDI0MTAzNDQ3WhcNOTcxMjIzMTAzNDQ3WjA4BgNVHR8B
 AQAELjAsMCqgKKAmhiRodHRwOi8vd3d3Lm15Y2EuY28uanAvY2dpLWJpbi9jcmxy
 ZXEwDgYKKoMIgZxfCwEEAQUAAykAVKRtTcur9tuvYOQxG2RJtt4ONcCEV/yoPqQo
 jXiDaD6Qg0BkVyELYg==

Connection closed by foreign host.
```

5.2.2 lookupreq with Distinguished Name

```
% telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
POST /cgi-bin/lookupreq HTTP/1.0
Content-length: 18

c=JP&o=NEC&cn=koji
HTTP/1.1 200 OK
Date: Sat, 04 Oct 1997 08:05:32 GMT
Server: Apache/1.2.1
Connection: close
Content-Type: text/plain

lookupreq
200 accept your request
 MIIBlTCCAT8CAQQwDQYJKoZIhvcNAQECBQAwZjELMAkGA1UEBhMCSlAxGDAWBgNV
 BAoTD05FQyBDb3Jwb3JhdGlvbjElMCMGA1UECxMcTmV0d29ya2luZyBTeXN0ZW1z
```

```
 IExhYnMuIENBNDEWMBQGA1UECxMNaW50ZXJuZXQgZGl2LjAeFw05NzEwMDIxMjIz
 MDZaFw05ODAzMzExMjIzMDZaMHAxCzAJBgNVBAYTAkpQMRgwFgYDVQQKEw9ORUMg
 Q29ycG9yYXRpb24xITAfBgNVBAsTGE5ldHdvcmtpbmcgU3lzdGVtcyBMYWJzLjEO
 MAwGA1UECxMFU291bXUxFDASBgNVBAMTC0tvamkgVGFraWRhMDEwDQYJKoZIhvcN
 AQEBBQADIAAwHQIWABIwmAmAmAm5gICYCYyYCY2YCKCYMAIDAQABMA0GCSqGSIb3
 DQEBAgUAA0EAHuIJIRU70hz/QzEFKOty5a7d7gb6nQ2iyxNUA/ykAyZJPcFPOuCT
 IeaoEKGu7oijoDWCMCyPQied5bi2fEK2UK==

Connection closed by foreign host.
```

5.2.3 lookupreq with Issuer and Serial Number

```
% telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
POST /cgi-bin/lookupreq HTTP/1.0
Content-length: 158

IASN=MGswZjELMAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjElMCMG
 A1UECxMcTmV0d29ya2luZyBTeXN0ZW1zIExhYnMuIENBNDEWMBQGA1UECxMNaW50
 ZXJuZXQgZGl2LgIBBA%3d%3d
HTTP/1.1 200 OK
Date: Sat, 04 Oct 1997 08:34:37 GMT
Server: Apache/1.2.1
Connection: close
Content-Type: application/x-pkixicap

lookupreq
200 accept your request
 MIIBlTCCAT8CAQQwDQYJKoZIhvcNAQECBQAwZjELMAkGA1UEBhMCSlAxGDAWBgNV
 BAoTD05FQyBDb3Jwb3JhdGlvbjElMCMGA1UECxMcTmV0d29ya2luZyBTeXN0ZW1z
 IExhYnMuIENBNDEWMBQGA1UECxMNaW50ZXJuZXQgZGl2LjAeFw05NzEwMDIxMjIz
 MDZaFw05ODAzMzExMjIzMDZaMHAxCzAJBgNVBAYTAkpQMRgwFgYDVQQKEw9ORUMg
 Q29ycG9yYXRpb24xITAfBgNVBAsTGE5ldHdvcmtpbmcgU3lzdGVtcyBMYWJzLjEO
 MAwGA1UECxMFU291bXUxFDASBgNVBAMTC0tvamkgVGFraWRhMDEwDQYJKoZIhvcN
 AQEBBQADIAAwHQIWABIwmAmAmAm5gICYCYyYCY2YCKCYMAIDAQABMA0GCSqGSIb3
 DQEBAgUAA0EAHuIJIRU70hz/QzEFKOty5a7d7gb6nQ2iyxNUA/ykAyZJPcFPOuCT
 IeaoEKGu7oijoDWCMCyPQied5bi2fEK2UK==

Connection closed by foreign host.
```

5.3 calookupreq

```
% telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
```

```
POST /cgi-bin/calookupreq HTTP/1.0
Content-length: 601

cert=MIIBqTCCAVMCAQIwDQYJKoZIhvcNAQECBQAwZjELMAkGA1UEBhMCSlAxGDAWBgNV
BAoTD05FQyBDb3Jwb3JhdGlvbjElMCMGA1UECxMcTmV0d29ya2luZyBTeXN0ZW1z
IExhYnMuIENBNDEWMBQGA1UECxMNaW50ZXJuZXQgZGl2LjAeFw05NzEwMDIxMDQy
NDJaFw05ODAzMzExMDQyNDJaMGIxCzAJBgNVBAYTAkpQMRgwFgYDVQQKEw9ORUMg
Q29ycG9yYXRpb24xITAfBgNVBAsTGE5ldHdvcmtpbmcgU3lzdGVtcyBMYWJzLjEW
MBQGA1UEAxMNVG9yYSBMdXRyYSgyKTBTMAQGAAUAA0sAMEgCQQCVs6HJAXV0qtMV
wP89UeMbmHNaVPBi5ceQDJMPxux3JvPxDwQ9bNVo5ZTFp7rvRBQP/KKxpWAPgh0V
%2blh6IwvLAgMBAAEwDQYJKoZIhvcNAQECBQADQQBxImito7%2b4omMh1TPbhyqZ/ghm
NUC/GBeZaFN29Wm8rmcgH7RjgrcD9iht3FFTW5Lq8Zw5%2bpEh6bKrFiYbKQsY
HTTP/1.1 200 OK
Date: Sat, 04 Oct 1997 08:54:56 GMT
Server: Apache/1.2.1
Connection: close
Content-Type: application/x-pkixicap

calookupreq
200 accept your request
 MIIBtjCCAWACAQAwDQYJKoZIhvcNAQECBQAwZjELMAkGA1UEBhMCSlAxGDAWBgNV
 BAoTD05FQyBDb3Jwb3JhdGlvbjElMCMGA1UECxMcTmV0d29ya2luZyBTeXN0ZW1z
 IExhYnMuIENBNDEWMBQGA1UECxMNaW50ZXJuZXQgZGl2LjAeFw05NzEwMDIxMTIw
 NDJaFw05ODEwMDIxMTIwNDJaMGYxCzAJBgNVBAYTAkpQMRgwFgYDVQQKEw9ORUMg
 Q29ycG9yYXRpb24xJTAjBgNVBAsTHE5ldHdvcmtpbmcgU3lzdGVtcyBMYWJzLiBD
 QTQxFjAUBgNVBAsTDWludGVybmV0IGRpdi4wXDANBgkqhkiG9w0BAQEFAANLADBI
 AkEAgDwky4mQrRadX7WT5AtpV9CFpFk0QhwL43mOwhSnykl5wDksC33KMThg+sBC
 nC3HNl7fb1iB6nYzsJSUirK3+wIDAQABMA0GCSqGSIb3DQEBAgUAA0EAXXGVLbQw
 lJXTLO6iNUzDXpMhN3aUsYc3dHfS0hWXE0s7yKIMxZrqg8Z6WDI4gVAnstLq6YPo
 mgewuYcONPWq6p==

Connection closed by foreign host.
```

5.4 crlreq

```
% telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
POST /cgi-bin/crlreq HTTP/1.0
Content-length: 495

cert=MIIBYzCCARYCAR4wBAYABQAwTTELMAkGA1UEBhMCSlAxDDAKBgNVBAoTA05FQzEw
MC4GA1UECxMnTmV0d29ya2luZyBTeXN0ZW1zIExhYnMgRXhwZXJpbWVudGFsIENB
MB4XDTk3MDcyMjA5NTIxMloXDTk3MTAzMTIzNTk1OVowPjELMAkGA1UEBhMCSlAx
FTATBgNVBAoTDFdJREUgUHJvamVjdDEYMBYGA1UEAxMPbWluZSBzYWt1cmFpKDMp
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAJpfMGvJfeRBelpIfdRDj4a9PkGlFMny
OX78pm8kKkD3pCNdHcMXac%2bAIp8CApA186O40O9Q9QHjftNI5scCs8sCAwEAATAE
```

```
BgAFAANBAGoH44rvFVZf6HdrbCu5est417ISth5ZfL5zZlRZlhxL3GlcEFBuqpI
xp7QbKOfstV4ppcVIKX48IJcehn4RK9%3d
HTTP/1.0 200 Document follows
Date: Sun, 14 Sep 1997 10:13:43 GMT
Server: NCSA/1.5.1
Content-type: application/x-pkixicap

crlreq
200 accept your request
 MIHnMIGSMA0GCSqGSIb3DQEBAgUAME0xCzAJBgNVBAYTAkpQMQwwCgYDVQQKEwNO
 RUMxMDAuBgNVBAsTJ05ldHdvcmtpbmcgU3lzdGVtcyBMYWJzIEV4cGVyaW1lbnRh
 bCBDQRcNOTcwNDIyMDc1MzQ5WhcNOTcwNTIyMDc1MzQ5WjAUMBICAQEXDTk3MDQy
 MjA3NTMwMlowDQYJKoZIhvcNAQECBQADQQBKUqMwSBwssoKOACJAN9jY8QvZhKCq
 JFoyIfFyaKgoIOHzCw5LY/MlIPVuSZ4a/mZAP4k89BbLxID26ZpBe8+/

Connection closed by foreign host.
```

5.5 verifyreq

```
% telnet cahost1 80
Trying 123.16.5.41 ...
Connected to cahost1.
Escape character is '^]'.
POST /cgi-bin/verifyreq HTTP/1.0
Content-length: 1200

resptype=1&cert=MIIDVDCCAxWgAwIBAgIBAjAOBgoqgwiBnF8LAQQBBQAwSjEL
MAkGA1UEBhMCSlAxGDAWBgNVBAoTD05FQyBDb3Jwb3JhdGlvbjEhMB8GA1UECxMY
TkVDIEV4cGVyaW1lbnRhbCBDQSAxMjAyMB4XDTk3MTIwMjA4Mjc1OVoXDTk4MDMw
MjA4Mjc1OVowga8xCzAJBgNVBAYTAkpQMQ4wDAYDVQQIEwVUb2t5bzEMMAoGA1UE
ERMDMTA4MRgwFgYDVQQKEw9ORUMgQ29ycG9yYXRpb24xGDAWBgNVBAsTD0NvbXB1
dGVyIENlbnRlcjEVMBMGA1UEAxMMV2Fyd2ljayBNb29uMREwDwYDVQQMEwhFbmdp
bmVlcjEkMCIGCSqGSIb3DQEJARYVd2Fyd2lja0BhYmMubmVjLmNvLmpwMIHWMIGo
BgoqgwiGjScHAQUBMIGZAgEBMCMGCiqDCIaNJwcBBgECFQD/////////////////
///////4IzAsBBQAAAAAAAAAAAAAAAAAAAAAAAAQUAAAAAAAAAAAAAAAAAAAAA
AAAAAAQEKHREkIEzBgj2pQP3jxfqq/HGpF5IjXaPj5/323zM8GqV5jc7d9QTWlIC
FAaQaQaQaQaQaQaJGF5nDdBMGhjrAgEAAykADlDr4ahA3gN89hnQHQQUGShmZJxe
FCXBf5qqQhtFszd9scSZyvjrN6OCAQIwgf8wDwYDVR0TAQEABAUwAwEBADAOBgNV
HQ8BAQAEBAMCAaAwPQYDVR0gAQEABDMwMTAvBgkqgwiBnF8LBAEwIjAgBgYrBgEF
AwQWFmh0dHA6Ly93d3cuaWNhdC5vci5qcC8wZgYLKoMIgZxfCwMBHQIBAQAEVDBS
oCiGJmh0dHA6Ly9zcGxzOTU6ODAwMC9jZ2ktYmluL2NhbG9va3VwcmVxoSaGJGh0
dHA6Ly9zcGxzOTU6ODAwMC9jZ2ktYmluL3ZlcmlmeXJlcTA1BgNVHR8BAQAEKzAp
MCegJaAjhiFodHRwOi8vc3Bsczk1OjgwMDAvY2dpLWJpbi9jcmxyZXEwDgYKKoMI
gZxfCwEEAQUAAykATQ4X5uxUoVw0rvgX1G1mygXkNYuZGQM3g18eboLgvFrQo5xA
AoCDaA%3d%3d
HTTP/1.1 200 OK
Date: Wed, 03 Dec 1997 02:39:00 GMT
Server: Apache/1.2.1
```

```
   Connection: close
   Content-Type: application/x-pkixicap

   verifyreq
   200 valid
   Connection closed by foreign host.
```

Acknowledgement

The authors thank Mr. Ohbayashi, Mr. Kitano, Mr. Kobayashi, Mr. Kuroda, Mr.Fujimoto, and Mr. Wada for their comments to this proposal. The authors also thank the other researchers joining the Initiatives for Computer Authentication Technology (ICAT), and the members of WIDE project.

References

[X.509] ITU-T Recommendation X.509 (1197 E):
Information Technology - Open Systems Interconnection -
The Directory: Authentication Framework, June 1997

[PKIX-PROF] R. Housley, et. al.,
"Internet Public Key Infrastructure
X.509 Certificate and CRL Profile,"
<draft-ietf-pkix-ipki-part1-08.txt>, June 1998

[PKIX-CMP] C. Adams and S. Farrell,
Internet Public Key Infrastructure
Certificate Management Protocols,"
<draft-ietf-pkix-ipki3cmp-07.txt>, February 1998

[PKIX-OCSP] C. Adams, et. al.,
"X.509 Internet Public Key Infrastructure
Online Certificate Status Protocol - OCSP",
<draft-ietf-pkix-ocsp-05.txt>, July 1998

[PKIX-OPP] R. Housley,
"Internet X.509 Public Key Infrastructure
Operational Protocols: FTP and HTTP",
<draft-ietf-pkix-opp-ftp-http-03.txt>, April 1998

[PKIX-LDAP] S. Boeyen, et. al.,
"Internet X.509 Public Key Infrastructure
LDAPv2 Schema",
<draft-ietf-pkix-ldapv2-schema-00.txt>, March 1998

[HTTP] T. Berners-Lee, R. Fielding, H. Nielsen, "Hypertext Transfer
Protocol -- HTTP/1.0", RFC 1945, May 1996

[SSL] http://home.netscape.com/eng/ssl3/draft302.txt, 1997

[ASN.1] B. Kaliski, "A Layman's Guide to a Subset of ASN.1, BER, and DER", ftp://ftp.rsa.com (layman.ps), June 1991

[Mine] M. Sakurai, et.al., "A Design of Certificate or CRL Distribution Architecture between Certification Authorities", The 1997 Symp. on Cryptography and Information Security (SCIS'97), 8D, January 1997

[RFC1779] S. Kille, "A String Representation of Distinguished Names", RFC 1779, March 1995

[PKCS-7] B. Kaliski, "PKCS 7: Cryptographic Message Syntax Version 1-5", RFC 2315, March 1998

[PKCS-9] RSA Laboratories, "PKCS #9: Selected Attribute Types", An RSA Laboatories Technical Note, November 1993

[PKCS-10] B. Kaliski, "PKCS 10: Certification Request Syntax Version 1-5", RFC 2314, March 1998

[RFC2045] N. Freed & N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC2045, November 1996

Security Considerations

This entire memo is about security mechanisms.

Author Addresses:

Mine Sakurai
NEC Corporation
Igarashi bldg., 2-11-5 Shibaura, Minato-ku, Tokyo 108-8557, Japan
m-sakura@ccs.mt.nec.co.jp

Hiroaki Kikuchi
Dept. of Electrical Engineering
Tokai University
1117 Kitakaname, Hiratsuka, Kanagawa 259-1292, Japan
kikn@ep.u-tokai.ac.jp

Hiroyuki Hattori
Meiji University
1-1, Kandasurugadai, Chiyoda-ku, Tokyo 101-8301, Japan
hhat@isc.meiji.ac.jp

Yoshiki Sameshima
Initiatives for Computer Authentication Technology
Information Security Office,
Japan Information Processing Development Center
3-5-8 Shiba-Kouen, Minato Ku, 105, Tokyo, Japan

Hitoshi Kumagai
Initiatives for Computer Authentication Technology
Information Security Office,
Japan Information Processing Development Center
3-5-8 Shiba-Kouen, Minato Ku, 105, Tokyo, Japan

Appendix: ICAT-local OIDs

In our sample implementation, two OIDs are originally defined and used. One is "App", and another is "V3Extn".

```
ICAT OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) jisc(392) JIPDEC(20063) 11}

App OBJECT IDENTIFIER ::=
{ICAT PKI(2) PKCS10(1) 1}

V3Extn OBJECT IDENTIFIER ::=
{ICAT PKI(2) PKCS10(1) 2}
```

The example of extended PKCS#10 message, encoded in BER, is as follows:

```
30 (739)
    30 (676)
        02 (1) 00
        30 (181)
            31 (11)
                30 (9)
                    06 (3) 550406
                    13 (2) [JP]
            31 (14)
                30 (12)
                    06 (3) 550408
                    13 (5) [Tokyo]
            31 (18)
                30 (16)
                    06 (3) 550407
                    13 (9) [Minato-ku]
            31 (24)
                30 (22)
                    06 (3) 55040a

                    13 (15) [NEC Corporation]
            31 (24)
                30 (22)
                    06 (3) 55040b
                    13 (15) [Computer Center]
            31 (20)
                30 (18)
                    06 (3) 550403
                    13 (11) [Taro Tanaka]
            31 (17)
                30 (15)
                    06 (3) 55040c
                    13 (8) [Engineer]
            31 (37)
                30 (35)
                    06 (9) 2a864886f70d010901
                    16 (22) [taro@aaa.bbb.nec.co.jp]
        30 (214)
            30 (168)
                06 (10) 2a8308868d2707010501
                30 (153)
                    02 (1) 01
                    30 (35)
                        06 (10) 2a8308868d2707010601
                        02 (21) 00ffffffffffffffffffffffffffffffff
fff823
                    30 (44)
```

```
                        04 (20) 0000000000000000000000000000000000000000
   0000
                        04 (20) 0000000000000000000000000000000000000000
   0004
                    04 (40) 74449081330608f6a503f78f17eaabf1c6a45e48
   8d768f8f9ff7db7cccf06a95e6373b77d4135a52
                    02 (20) 0690690690690690690689185e670dd04c1a18eb
                    02 (1) 00
               03 (41) 006d4cf2c4c6ba1c75f4dd6b330dea86f82934b99704937d
   7b85f4a4a6010b539df905329342f10a17
           A0 (268)
               30 (23)
                   06 (9) 2a864886f70d010902
                   31 (10)
                       16 (8) [TESTUSER]
               30 (23)
                   06 (9) 2a864886f70d010907
                   31 (10)
                       13 (8) [TESTPASS]
               30 (21)
                   06 (10) 2a8308819c5f0b020101
                   31 (7)
                       13 (5) [PEPOP]
               30 (192)
                   06 (10) 2a8308819c5f0b020102
                   31 (177)
                       30 (174)
                           30 (15)
                               06 (3) 551d13
                               01 (1) 00
                               04 (5) 3003010100
                           30 (154)
                               06 (11) 2a8308819c5f0b03011d01
                               01 (1) 00
                               04 (135) 308184306430180603550 40a3111130
   f4e454320436f72706f726174696f6e300e06035504083107130 5546f6b796f30110
   60355040c310a1308456e67696e656572302506092a864886f70d010901311816167
   461726f406161612e6262622e6e65632e636f2e6a70170d393730383235303030303
   0305a170d393731313235303030303030305a
       30 (14)
           06 (10) 2a8308819c5f0b010401
           05 (0)
       03 (41) 00fa54b614e767e8a8ccd6c595dae74fce02a25d8faf78bd080c70ca
   85b00b5c0354966f9f02ad2c9a
```

Note1: Strings in () are represented in decimal.

Note2: In this example, we use Matsushita's Elliptic Curve Cryptosystems, My-Ellty, as public-key algorithm.