Int. J. Communications, Network and System Sciences, 2009, 5, 378-384
doi:10.4236/ijcns.2009.25042 Published Online August 2009 (http://www.SciRP.org/journal/ijcns/).

Notification Services for the Server-Based Certificate Validation Protocol

Johannes BUCHMANN, Vangelis KARATSIOLIS
Technische Universität Darmstadt, Department of Computer Science, Cryptography and Computeralgebra, Darmstadt, Germany
Email: {*buchmann,karatsio*}*@cdc.informatik.tu-darmstadt.de*
Received May 15, 2009; *revised June* 17, 2009; *accepted July* 22, 2009

ABSTRACT

The Server-Based Certificate Validation Protocol allows PKI clients to delegate to a server the construction or validation of certification paths. The protocol's specification focuses on the communication between the server and the client and its security. It does not discuss how the servers can efficiently locate the necessary PKI resources like certificate or certificate revocation lists. In this paper we concentrate on this topic. We present a simple and effective method to facilitate locating and using various PKI resources by the servers, without modifying the protocol. We use the extension mechanism of the protocol for notifying the servers about PKI repositories, certificates, and revocations. We specify the tasks of the servers and certificate issuers and define the messages that are exchanged between them. A proof of concept is given by implementing an SCVP server, a client, and the proposed method in Java.

Keywords: SCVP, Certification Path, Certification Path Construction, Certification Path Validation, X.509 Certificate

1. Introduction

A Public Key Infrastructure (PKI) has lots of protocols and processes that support important functions of the infrastructure. The building of a certification path and its validation are two of them. The PKI clients need to perform these operations before they can securely use an X.509 certificate. But there are clients that are not able or they simply do not want to perform certification path construction or validation themselves.

For these clients a protocol has been specified by the IETF. This is the server-based certificate validation protocol (SCVP) [1]. This protocol allows clients to delegate the building or validation of a certification path to a server.

Once a request reaches the server, the server tries to build the certification path. For performing this task it needs to contact various repositories and download certificates and CRLs. It is not always possible to construct such a path, if the repositories are not reachable by the server. Further, it is not always possible to locate the correct certificates or CRLs. Moreover, certification authorities that operate an SCVP server need to configure this server in such a way that it is able to efficiently locate their resources.

In this paper we concentrate on such implementation issues. Especially we see how to use the extension mechanisms of the protocol to provide the SCVP server with important resources for its functioning. These are for example the trust anchors of a PKI, the revocation lists, or the location of repositories. We show how to create appropriate messages that are sent by a purpose-specific client charged with this task by the CA. We also present the prototype implementation of an SCVP server in Java. This SCVP server is notified about the PKI resources by a notification client using our proposed method.

1.1. Notation

For the rest of the paper we denote by C^A_B the certificate issued to entity B by entity A. By CP: [C^A_C and C^C_E] we

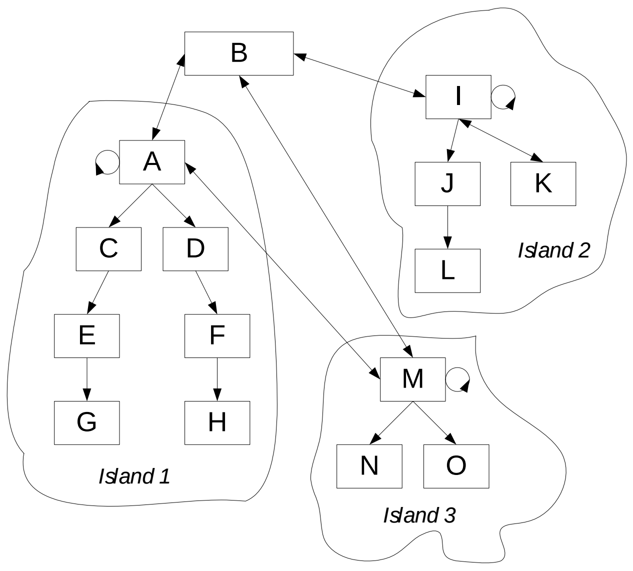

Figure 1. An example of a PKI topology with three independent infrastructures (islands) and a CA acting as Bridge (entity B).

denote the certification path which consists of the certificates C^A_C and C^C_E. We will use the PKI topology depicted in Figure 1 in our examples. The boxes represent entities and the arrows represent certificates issued by one entity to another (in the arrow direction).

This paper is organised as follows: In Section 2 we briefly discuss certification path construction and validation. In Section 3 we describe the SCVP. In Section 4 we present the extension of the protocol for sending notifications to the server. In Section 5 we give the prototype implementation of an SCVP server which uses our proposed method. We conclude our work in Section 6.

2. Certification Path Building and Validation

One client receives a certificate. The client wants to verify whether the binding of the public key and the certificate's subject (found in the subject distinguished name and/or the subject alternative name) is valid [2]. For verifying that, the client needs, among others, all certificates in the certification chain between one of its trust anchors and the certificate in question. Suppose for example that the client wants to verify the certificate C^E_G and it possesses one trust anchor. The trust anchor is entity *A*. In this case the certification path is *CP: [*C^A_C, C^C_E, *and* C^E_G*]*. The certificate of the trust anchor C^A_A is not part of the certification path. Building this certification path can be easy. But more complicated paths are necessary to be constructed. For example, if the same certificate C^E_G needs to be verified but the trust anchor is entity *K*, then the construction of this certification path is more complex. Guidelines for building certification paths are given in [3].

If a client does not want to perform certification path construction on its own then it can delegate this to the SCVP server. The server will then try to construct the path (following the guidelines from [3]).

The validation of the certification path is the next step in the verification. The most commonly used algorithm for this purpose is described in Section 6 of RFC 3280 [2]. This algorithm takes as input the certification path, the current time of the validation, the set of allowed policies, some other policy related parameters, and information about the trust anchor. The last is the name of the trust anchor, the algorithm of its public key (with optional parameters), and the public key itself. This information is trusted. The algorithm outputs the result of the validation and, in case of successful validation, the public key that has been validated (with parameters and algorithm) and policy related information. An SCVP server must implement this algorithm (see [1]).

Certification path validation assumes that a certification path already exists. Therefore validation of a certification path implies that a certification path building process has been already conducted.

For attribute certificates [4] these processes are similar. The default validation algorithm is described in [4].

3. The Server-Based Certificate Validation Protocol (SCVP)

SCVP [1] is a protocol specified by the IETF. The goal of the protocol is to allow clients that cannot perform certification path building or certification path validation to delegate this task to a server. A reason for doing this is that the clients cannot locate the resources themselves or they do not support the necessary protocols (for example OCSP [5]). The process of delegating the certification path building is also known as DPD (delegated path discovery) and this of delegating the validation as DPV (delegated path validation). They are defined along with their requirements in [6].

For delegating the above tasks, the client sends a CVRequest [1] (see Figure 2) to the server. This request can be signed or a MAC value can be calculated over the request and be sent with it. In these two cases the request is encapsulated in a CMS [7] message.

The query (of the type *Query*) contains the certificates for which the clients request the certification path to be built or validated. The specification of a query can be seen in Figure 3. It is possible to define whether the certification path should be built, validated, or validated with revocation checking. This is specified in the *checks*. The protocol also allows the client to specify the type of

```
CVRequest ::= SEQUENCE {
cvRequest Version     INTEGER DEFAULT 1,
query                 Query,
requestorRef          [0] GeneralNames OPTIONAL,
requestNonce          [1] OCTET STRING OPTIONAL,
requestorName         [2] GeneralName OPTIONAL,
responderName         [3] GeneralName OPTIONAL,
requestExtensions     [4] Extensions OPTIONAL,
signatureAlg          [5] AlgorithmIdentifier OPTIONAL,
hashAlg               [6] OBJECT IDENTIFIER
                      OPTIONAL,
requestorText         [7] UTF8String (SIZE (1..256))
                      OPTIONAL}
```

Figure 2. CVRequest.

```
Query ::= SEQUENCE {
queriedCerts          CertReferences,
checks                CertChecks,
wantBack              [1] WantBack OPTIONAL,
validationPolicy      ValidationPolicy,
responseFlags         ResponseFlags OPTIONAL,
serverContextInfo     [2] OCTET STRING OPTIONAL,
validationTime        [3] GeneralizedTime OP TIONAL,
intermediateCerts     [4] CertBundle OPTIONAL,
revInfos              [5] RevocationInfos OPTIONAL,
producedAt            [6] GeneralizedTime OPTIONAL,
queryExtensions       [7] Extensions OPTIONAL }
```

Figure 3. Query.

```
ValidationPolicy ::= SEQUENCE {
validationPolRef        ValidationPolRef,
validationAlg           [0] ValidationAlg OPTIONAL,
userPolicySet           [1] SEQUENCE SIZE (1..MAX) OF
                        OBJECT IDENTIFIER OPTIONAL,
inhibitPolicyMapping    [2] BOOLEAN OPTIONAL,
requireExplicitPolicy   [3] BOOLEAN OPTIONAL,
inhibitAnyPolicy        [4] BOOLEAN OPTIONAL,
trustAnchors            [5] TrustAnchors OPTIONAL,
keyUsages               [6] SEQUENCE OF KeyUsage OP-
                        TIONAL,
extendedKeyUsages       [7] SEQUENCE OF KeyPurposeId
                        OPTIONAL,
specifiedKeyUsages      [8] SEQUENCE OF KeyPurposeId
                        OPTIONAL }
```

Figure 4. Validation policy.

objects that must be returned by the server. This is covered by the *wantBack* element.

For defining the policies that the server should use for validating a certificate, the *validationPolicy* element is used (see Figure 4).

```
notification OBJECT IDENTIFIER ::=
      {1.3.6.1.4.1.8301.3.8.1.1}

Notification ::= SEQUENCE OF EXTENSIONS
```

Figure 5. Notification request.

To facilitate the building and validation of certification paths by the server, we extend the CVRequest by providing notifications about PKI resources within the request. Many aspects of the protocol are reused in order not to affect it significantly.

4. The Notification Messages

The notification message is a standard CVRequest. To distinguish it as a notification message it contains an extension (as this is defined in [2]) called *Notification*. The specification of the extension and its object identifier (OID) is found in Figure 5.

The Notification is a sequence of already existing extensions that are used in the X.509 based PKI. These extensions can hold all necessary information that is required for notifying the server for new resources. This is specified like that in order to minimise the effort of PKI practitioners to implement the proposed notification method. In addition by being a sequence of extensions it is possible to notify the server about various resources within one notification request. The Notification extension is non-critical.

There are six types of notification. These notify the server about: a) trust anchors, b) other certificates (for example of CRL signers), c) CRLs and delta-CRLs, d) repositories for revocation purposes, e) repositories for certificates, and f) cross certificates.

4.1. Notification about Trust Anchors

This type of notification notifies the server about the trust anchors of a PKI. Trust anchors are all entities that are allowed to issue certificates.[1] However, in practice only entities that possess a self-signed certificate are considered trust anchors. Therefore we propose to include only such certificates in this notification.

This Notification is an empty sequence. In the *trustAnchors* element of the ValidationPolicy the trust anchors of the PKI are sent. In our example, for the first PKI island (Island 1) entities *A*, *C*, *E*, *D*, and *F* are certification authorities. In this case the trustAnchors element may consist of five certificates. These are C^{A}_{A}, C^{A}_{C}, C^{C}_{E}, C^{A}_{D}, and C^{D}_{F}. We propose to include only C^{A}_{A}. The other four certificates can be included in a notification about

[1]Such entities possess a certificate which contains the basic constraints extension and has the value true for the *cA* Boolean flag (see [9]).

cross certificates (see Subsection 4.6).

Trusting C^A_A is a very critical operation. If this certificate is not a legitimate one, then the SCVP server may return wrong results. For this reason some PKIs may introduce an out-of-band mechanism that provides the SCVP server with information about which self-signed certificates are trusted or not. One technical realisation of this concept is to have a configuration file, signed by an administrator, which contains the fingerprints of known valid self-signed certificates. The SCVP server compares the fingerprint of the self-signed certificate provided in the request with those in the file. If a match is found then it accepts the certificate, otherwise it discards it. All other (non self-signed) certificates are verified before they are considered trust anchors.

4.2. Notification about Other Certificates

There are certificates that are required during a validation but do not belong to certification authorities. These are the certificates of CRL signers (entities that issue indirect CRLs), of OCSP signers, and of SCVP servers. These certificates are used for verifying signatures on revocation lists, on OCSP responses, and on SCVP responses respectively.

These certificates are sent within the *intermediateCerts* element of the Query. The Notification element is an empty sequence.

4.3. Notification about Revocation Lists

This type of notification is used for sending the CRLs or delta-CRLs to the SCVP server. To send the CRLs to the server the *revInfos* element of the Query of the type RevocationInfos (see Figure 6) is used. From this element only the *crl* and the *delta-crl* fields are used.

The Notification is an empty sequence. These notifications can also be used in a "push-mode". In this mode the CRLs are sent to the server as soon as they are issued. Such a mechanism is useful in certain environments. In this case the SCVP server has always fresh revocation information.

```
RevocationInfos ::= SEQUENCE SIZE
    (1..MAX) OF RevocationInfo

RevocationInfo ::= CHOICE {
    crl          [0] CertificateList,
    delta-crl    [1] CertificateList,
    ocsp         [2] OCSPResponse,
    other        [3] OtherRevInfo }
```

Figure 6. Revocation Infos.

Table 1. Elements of general names.

Type of resource	Element of GeneralName
LDAP	directoryName
X.500	directoryName
Web or FTP	uniformResourceIdentifier
HTTP, WebDAV	uniformResourceIdentifier
DNS	dNSName

4.4. Notification about Revocation Repositories

It may not be possible or desired that the CA or a CRL issuer sends every CRL to the SCVP server. In addition the location of an OCSP server may be unknown to it. In these cases the SCVP server can be notified about the location where these resources can be found. This is very helpful if the certificates issued by the CA do not contain the CRLDistributionPoint (for CRLs), FreshestCRL (for delta-CRLs), or Authority Information Access (for OCSP) extensions. But even if these values are present, once they are set in a certificate they cannot be changed. This is a problem if the resources have been moved or do not exist at all[1] and they cannot be accessed anymore.

The revocation resources can be located in diverse repositories. Examples of typical repositories that are used in a PKI are X.500 directories [8], LDAP directories [9], DNS servers [10], WebDAV [11], Web or FTP servers [12], or HTTP stores according to [13] specified additionally in [14] as an RFC. To notify about the location of a CRL the CRLDistributionPoint [2] extension is added to the sequence of extensions of the Notification. For the location of delta-CRLs the FreshestCRL [2] extension is added. In these extensions the GeneralNames [2] element is used for specifying the different locations. In Table 1 the elements of GeneralNames that are used for describing the resources are given.

For notifying about the location of OCSP servers, the Authority Information Access [2] extension is added to the list of extensions. It is possible to notify the SCVP server for more than one repository within one notification request.

4.5. Notification about Certificate Repositories

The CA may not wish to send any certificates to the SCVP server but just notify it about the repositories in which these are located.

In this case it sends a Notification request which contains the Subject Information Access extension (see [2]). This extension contains the *caRepository* access method. This specifies the location of the repository used by a CA. The value of the location is specified as GeneralName.

[1]Typical example is that of a CA stopping operation.

The same principles as in the case of the revocation resources apply here. It is possible to send a notification about more than one repository location by defining more AccessDescription elements inside the extension.

4.6. Notification about Cross Certificates

Cross certifications may occur any time and the number of cross certificates of a CA can be large. For notifying the servers about such certificates the SCVP notifier sends a Notification request with the Authority Information Access extension present. This extension contains the *caIssuers* access method (see [2]) which points to the location where cross certificates are stored. An example value for this location is: *ldap://host:389/CN=CA, C=DE, DC=Org, DC=COM/ cross CertificatePair; binary? sub? object Class=pkiCA.* This address is found in an LDAP directory and follows the LDAP URL format. This method allows the CA to notify the SCVP server only once, stating where past and possibly future cross certificates can be located.[1] A condition for this notification to function properly is that the CA publishes all cross certificates (issuedTo and issuedBy) in the directory.

An alternative to this approach is to send a Notification as an empty sequence with the cross certificates stored in the *intermediateCerts* element of the Query. These can be distinguished from the certificates discussed in Section 4.2, because the basic constraints extension identifies them as CA certificates. For example, entity *A* sends the certificates C^A_M, C^M_A, C^A_B, *and* C^B_A. Certificates C^A_C, C^C_E, C^A_D, *and* C^D_F are also sent within this type of notification.

4.7. Summary of the Messages

A summary of the types of notification that can be sent to the SCVP server is given in Table 2. The type of resource for which the notification is performed is given in the first column. The Notification Extension column describes the contents of the Notification extension and the Influenced Element column the element of the CVRequest that is used in the request.

4.8. The Notification Client

The notification client is the entity inside a PKI which is responsible for notifying SCVP servers about the resources of the PKI. One choice for being a notification client is the online part of the CA. Another choice is the components that administrate the certificates and revocations and are usually employed for updating an OCSP or an LDAP server.

The CA issues a special certificate to the notification client. A notification client certificate is an X.509 certificate that has the extended key usage (see [9]) extension set. The value of this extension contains only one KeyPurposeId which is identified by the OID "1.3.6.1.4.1.8301.3.8.1.2". The extension is marked critical.

The notification client always sends signed requests to the SCVP server. The client can also check whether the SCVP server has accepted the information that was sent to it and has been successfully updated. This is very useful when the notification messages are used by a CA to update the backend of its own SCVP server. In this case it includes certain certificates within the notification, namely in the *queriedCerts* element of the Query. Typical choices for certificates to include in the query for testing whether the server can build and verify a certification path are valid certificates issued recently by the CA. Another choice is certificates that have been revoked. By asking for a validation of the latest revoked certificate the client can test whether the SCVP server has received the freshest CRL. To properly evaluate the result of the verification the client chooses proper values for the *checks* (see [1]) element of the Query.

Table 2. Overview of notifications.

Type of resource	Notification Extension	Influenced Element
Trust Anchors	empty	trustAnchors
Other Certificates	empty	intermediateCerts
CRLs	empty	crl
Delta-CRLs	empty	delta-crl
CRL Repository	CRLDistributionPoints	none
Delta-CRL Repository	FreshestCRL	none
OCSP server	Authority Information Access	none
Cross Certifications	Authority Information Access	intermediateCerts (opt.)
General Repository	Subject Information Access	none

4.9. Implementation Guidelines for the Server

We present a small catalogue of implementation guide lines for SCVP servers that need to be taken into account

[1]This is a supported by the value of the URL. The URL of this example allows a search of unlimited depth beneath this CA entry for all cross certificates in the directory. Other URLs may not be able to support this type of "dynamic" notification.

for a proper and secure use of the notification requests.

• The notification client must present a valid certificate that has a critical extended key usage extension which contains only the OID "1.3.6.1.4.1.8301.3.8.1.2".

• All notification requests (Notification extension present) that contain a valid signature are accepted. The server may ignore notification requests in some cases, for example when it is overloaded. Notification requests that are neither signed nor have a valid signature are rejected.

• Self-signed certificates provided in a notification request, may be considered trustworthy only if there is an out-of-band mechanism that ensures that these are indeed trusted. This depends on the PKI. All other certificates are verified.

• Information retrieved by the server or provided by the notification client should be stored in a local repository. A database or an LDAP directory can be used for this purpose. When a CVRequest reaches the server, this should try first to access its backend and if no information is found or is not recent enough, then it should try to contact external resources.

• Optionally the SCVP server can forward a notification request to other SCVP servers.

5. Design and Implementation

We have designed and implemented a prototype SCVP server and client as well as the proposed notification request. Some features like attribute certificates and delta-crls are not supported in the current implementation.

The SCVP server is implemented as a Java servlet. The servlet container is Apache Tomcat 6.0. The backend of the server is the file system. That is all certificates and revocation lists that are used by the server to answer requests are stored on the hard disc. The server signs its responses with keys stored either in software in the PKCS#12 format or in hardware by using a smart card. The connection to the smart card is realised with the classes contained in *javax.smartcardio* package which are available since Java 6.0. The communication with the card reader and the smart card is done over PC/SC.

The SCVP server operates for the first PKI of our example (Island 1 in Figure 1). The server starts operation without having any certificate or CRL stored in its backend at all. The backend is updated exclusively by notification requests. The test scenario is to notify the server about the trust anchor of the PKI and the other intermediate certificates. Afterwards the client sends a regular CVRequest about a certificate which has not been revoked. The server should be able to build and return a valid certification path. Then, this certificate is revoked, the CA issues a CRL, and the notification client informs the server about the new CRL by sending a notification request as described in our method. The expected answer is that the certificate is revoked. For concentrating only on the performance of the communication all certificates, CRLs, and requests are pre-produced and are just sent to the server.

For testing the implementation and the efficiency of the notification method the client sends 1000 requests to the server. Half of them regard not revoked certificates while the other half revoked ones. When the server signs its requests using keys stored in software it takes approximately 57 ms to answer a request. That is to accept it, verify and process it, create and sign the response and finally send this back to the client. When keys (1024 bits RSA) stored in a smartcard are used the required time is about 480 ms. The server runs on an Intel Core Duo with 1.6 GHz.

The server starts operating without any certificates or CRLs stored in its backend. However, by using the notification method described in the paper it is possible to update its backend and enable it to produce useful and reasonable answers. In addition, when certificates are revoked the server is immediately notified about it and responds taking these revocations into account. Moreover, this server can be used by any PKI that wishes to add SCVP services without modifying the current PKI. Old certificates and CRLs need to be sent once in the beginning and newly produced ones need to be sent to the server as a notification request. Only a notification client should be implemented and customised according to the requirements of the PKI.

6. Conclusions

In this paper we presented a simple method for notifying the SCVP servers about PKI resources. We showed the necessary steps that a CA and the SCVP server perform and the messages exchanged between them. This method can be used to notify general purpose SCVP servers as well as the own SCVP server of a CA. This method is very useful when an SCVP server may not be able to locate the resources of a PKI. For example the certificates are stored in a database in which the server does not have access. This is common when PKIs of different organisations are involved. Moreover, this method can be used for forwarding notification requests to other SCVP servers. We also provided a prototype implementation of an SCVP server and client as well as an implementation of the proposed method in Java. It was shown that the method is effective for notifying an SCVP server about certificates and revocation lists.

7. References

[1] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk, "Server-based certificate validation protocol (SCVP)," IETF Request for Comments, Vol. 5055, December 2007.

[2] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," IETF Request for Comments, Vol. 3280, April 2002.

[3] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, and R. Nicholas, "Internet X.509 public key infrastructure: Certification path building," IETF Request for Comments, Vol. 4158, September 2005.

[4] S. Farrell and R. Housley, "An internet attribute certificate profile for authorization," IETF Request for Comments, Vol. 3281, April 2002.

[5] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol–OCSP," IETF Request for Comments, Vol. 2560, June 1999.

[6] D. Pinkas and R. Housley, "Delegated path validation and delegated path discovery protocol requirements," IETF Request for Comments, Vol. 3379, September 2002.

[7] R. Housley, "Cryptographic message syntax (CMS)," IETF Request for Comments, Vol. 3852, July 2004.

[8] "Recommendation X.500 ITU-T information technology – open systems interconnection – the directory: Overview of concepts, models and services," August 2005.

[9] J. Sermersheim, "Lightweight directory access protocol (LDAP): The protocol," IETF Request for Comments, Vol. 4511, June 2006.

[10] S. Josefsson, "Storing certificates in the domain name system (DNS)," IETF Request for Comments, Vol. 4398, March 2006.

[11] D. W. Chadwick and S. Anthony, "Using WebDAV for improved certificate revocation and publication," In Proceedings of Public Key Infrastructure: 4th European PKI Workshop: Theory and Practice, EuroPKI, Lecture Notes in Computer Science, Vol. 4582, pp. 265–279, June 2007.

[12] R. Housley and P. Hoffman, "Internet X.509 public key infrastructure operational protocols: FTP and HTTP," IETF Request for Comments, Vol. 2585, May 1999.

[13] P. Gutmann and A. Reliable, "Scalable general-purpose certificate store," In Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00*)*, pp. 278–287, December 2000.

[14] P. Gutmann, "Internet X.509 public key infrastructure operational protocols: Certificate store access via HTTP," IETF Request for Comments, Vol. 4387, February 2006.