

Network Working Group
Internet-Draft
Intended status: Proposed Standard
Updates: 2560 (once approved)
Expires: March 6, 2011

S. Santesson
3xA Security
P. Hallam-Baker
Default Deny Security
September 2, 2010

OCSP Algorithm Agility
draft-ietf-pkix-ocspagility-09

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

Copyright and License Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

The OSCP specification defined in RFC 2560 [RFC2560] requires server responses to be signed but does not specify a mechanism for selecting the signature algorithm to be used. This may lead to avoidable interoperability failures in contexts where multiple signature algorithms are in use. This document specifies rules for server signature algorithm selection and an extension that allows a client to advise a server that specific signature algorithms are supported.

Table of Contents

1	Introduction	3
1.1	Requirements Language	3
2	OCSP Algorithm Agility Requirements	3
3	Updates to Mandatory and Optional Cryptographic Algorithms	4
4	Client Indication of Preferred Signature Algorithms	5
5	Responder Signature Algorithm Selection	6
5.1	Dynamic Response	6
5.2	Static Response	6
6	Acknowledgements	7
7	IANA Considerations	7
8	Security Considerations	7
8.1	Use of insecure algorithms	7
8.2	Man in the Middle Downgrade Attack	8
8.3	Denial of Service Attack	8
9	References	9
9.1	Normative References	9
9.2	Informative References	9
	Appendix A - ASN.1 Modules	10
A.1	ASN.1 Module	10
A.2	1988 ASN.1 Module	11
	Author's Address	12

1 Introduction

1.1 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2 OCSP Algorithm Agility Requirements

OCSP RFC 2560 [RFC2560] defines a protocol for obtaining certificate status information from an online service. An OCSP Responder may or may not be issued an OCSP Responder certificate by the CA that issued the certificate whose status is being queried. An OCSP Responder may provide pre-signed OCSP responses or may sign responses when queried.

RFC 2560 [RFC2560] specifies a means for an OCSP responder to indicate the signature and digest algorithms used in a response but not how those algorithms are specified. The only algorithm requirements established by that protocol specification are that the OCSP client SHALL support the DSA sig-alg-oid specified in section 7.2.2 of [RFC2459] and SHOULD be capable of processing RSA signatures as specified in section 7.2.1 of [RFC2459]. The only requirement placed on responders by RFC 2560 is that they SHALL support the SHA1 hashing algorithm.

Since algorithms other than the mandatory to implement algorithms are Allowed, and since a client currently has no mechanism to indicate it's algorithm preferences, there is always a risk that a server choosing a non-mandatory algorithm, will generate a response that the client may not support.

While an OCSP Responder may apply rules for algorithm selection, e.g., using the signature algorithm employed by the CA for signing CRLs and certificates, such rules may fail in common situations:

- o The algorithm used to sign the CRLs and certificates may not be consistent with key pair being used by the OCSP Responder to sign responses.
- o A request for an unknown certificate provides no basis for a responder to select from among multiple algorithm options.

The last criterion cannot be resolved through the information available from in-band signaling using the RFC 2560 [RFC2560] protocol, without modifying the protocol.

In addition, an OCSP Responder may wish to employ different signature algorithms than the one used by the CA to sign certificates and CRLs for several reasons:

- o The responder may employ an algorithm for certificate status response that is less computationally demanding than for signing the certificate itself.
- o An implementation may wish to guard against the possibility of a compromise resulting from a signature algorithm compromise by employing two separate signature algorithms.

This document describes:

- o A mechanism that allows a client to indicate the set of preferred signature algorithms.
- o Rules for signature algorithm selection that maximizes the probability of successful operation in the case that no supported preferred algorithm(s) are specified.

3 Updates to Mandatory and Optional Cryptographic Algorithms

Section 4.3 "Mandatory and Optional Cryptographic Algorithms" of RFC 2560 [RFC2560] is updated as follows:

OLD: Clients that request OCSP services SHALL be capable of processing responses signed using DSA keys identified by the DSA sig-alg-oid specified in section 7.2.2 of [RFC2459]. Clients SHOULD also be capable of processing RSA signatures as specified in section 7.2.1 of [RFC2459]. OCSP responders SHALL support the SHA1 hashing algorithm.

NEW: Clients that request OCSP services SHALL be capable of processing responses signed using RSA with SHA-1 (identified by sha1WithRSAEncryption OID specified in [RFC3279]) and RSA with SHA-256 (identified by sha256WithRSAEncryption OID specified in [RFC4055]). Clients SHOULD also be capable of processing responses signed using DSA keys (identified by the id-dsa-with-sha1 OID specified in [RFC3279]). Clients MAY support other algorithms.

4 Client Indication of Preferred Signature Algorithms

A client MAY declare a preferred set of algorithms in a request by including a preferred signature algorithms extension in requestExtensions of the OCSPRequest [RFC2560].

```
id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }
```

```
PreferredSignatureAlgorithms ::= SEQUENCE OF  
    PreferredSignatureAlgorithm
```

```
PreferredSignatureAlgorithm ::= SEQUENCE {  
    sigIdentifier    AlgorithmIdentifier,  
    certIdentifier   AlgorithmIdentifier OPTIONAL  
}
```

The syntax of AlgorithmIdentifier is defined in section 4.1.1.2 of RFC 5280 [RFC5280]

sigIdentifier specifies the signature algorithm the client prefers, e.g. algorithm=ecdsa-with-sha256. Parameters are absent for most common signature algorithms.

certIdentifier specifies the subject public key algorithm identifier the client prefers in the server's certificate used to validate the OCSP response. e.g. algorithm=id-ecPublicKey and parameters=secp256r1.

certIdentifier is OPTIONAL and provides means to specify parameters necessary to distinguish among different usages of a particular algorithm, e.g. it may be used by the client to specify what curve it supports for a given elliptic curve algorithm.

The client MUST support each of the specified preferred signature algorithms and the client MUST specify the algorithms in the order of preference.

The server SHOULD use one of the preferred signature algorithms for signing OCSP responses to the requesting client.

5 Responder Signature Algorithm Selection

RFC 2560 [RFC2560] does not specify a mechanism for deciding the signature algorithm to be used in an OCSP response. As previously noted this does not provide a sufficient degree of certainty as to the algorithm selected to facilitate interoperability.

5.1 Dynamic Response

A responder MAY maximize the potential for ensuring interoperability by selecting a supported signature algorithm using the following order of precedence, as long as the selected algorithm meets all security requirements of the OCSP responder, where the first method has the highest precedence:

1. Select an algorithm specified as a preferred signing algorithm in the client request
2. Select the signing algorithm used to sign a CRL issued by the certificate issuer providing status information for the certificate specified by CertID
3. Select the signing algorithm used to sign the OCSPRequest
4. Select a signature algorithm that has been advertised as being the default signature algorithm for the signing service using an out of band mechanism
5. Select a mandatory or recommended signing algorithm specified for the version of the OCSP protocol in use

A responder SHOULD always apply the lowest numbered selection mechanism that is known, supported, and that meets the responder's criteria for cryptographic algorithm strength.

5.2 Static Response

For purposes of efficiency, an OCSP responder is permitted to generate static responses in advance of a request. The case may not permit the responder to make use of the client request data during the response generation, however the responder SHOULD still use the client request data during the selection of the pre-generated response to be returned. Responders MAY use the historical client requests as part of the input to the decisions of what different algorithms should be used to sign the pre-generated responses.

6 Acknowledgements

The authors acknowledges Santosh Chokhani for the helpful comments made on earlier drafts, Sean Turner for proposing the syntax for algorithm identifiers, Jim Schaad for providing and testing the ASN.1 module in Annex A and Stephen Kent for valuable review and input.

7 IANA Considerations

This document requires no actions by IANA.

8 Security Considerations

The mechanism used to choose the response signing algorithm MUST be considered to be sufficiently secure against cryptanalytic attack for the intended application.

In most applications it is sufficient for the signing algorithm to be at least as secure as the signing algorithm used to sign the original certificate whose status is being queried. This criteria may not hold in long term archival applications however in which the status of a certificate is being queried for a date in the distant past, long after the signing algorithm has ceased being considered trustworthy.

8.1 Use of insecure algorithms

It is not always possible for a responder to generate a response that the client is expected to understand and that meets contemporary standards for cryptographic security. In such cases an OCSP responder operator MUST balance the risk of employing a compromised security solution and the cost of mandating an upgrade, including the risk that the alternative chosen by end users will offer even less security or no security.

In archival applications it is quite possible that an OCSP responder might be asked to report the validity of a certificate on a date in the distant past. Such a certificate might employ a signing method that is no longer considered acceptably secure. In such circumstances the responder MUST NOT generate a signature for a signing mechanism that is considered unacceptably insecure.

A client MUST accept any signing algorithm in a response that it specified as a preferred signing algorithm in the request. It follows therefore that a client MUST NOT specify as a preferred signing algorithm any algorithm that is either not supported or not

considered acceptably secure.

8.2 Man in the Middle Downgrade Attack

The mechanism to support client indication of preferred signature algorithms is not protected against a man in the middle downgrade attack. This constraint is not considered to be a significant security concern since the OCSP Responder MUST NOT sign OCSP Responses using weak algorithms even if requested by the client. In addition, the client can reject OCSP responses that do not meet its own criteria for acceptable cryptographic security no matter what mechanism is used to determine the signing algorithm of the response.

8.3. Denial of Service Attack

Algorithm agility mechanisms defined in this document introduces a slightly increased attack surface for Denial of Service attacks where the client request is altered to require algorithms that are not supported by the server, alternatively does not match pre-generated responses.

9 References

9.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.
- [RFC3279] W. Polk, R. Housley, L. Bassham, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.
- [RFC4055] J. Schaad, B. Kaliski, R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.
- [RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [NEWASN] P. Hoffman, J. Schaad, "New ASN.1 Modules for PKIX", draft-ietf-pkix-new-asn1, August 2009.

9.2 Informative References

- [RFC2459] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure - Certificate and CRL Profile", January 1999

Appendix A - ASN.1 Modules

A.1 ASN.1 Module

```
OCSP-AGILITY-2009 { iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-ocsp-agility-2009-93(66) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

  EXPORTS ALL;  -- export all items from this module
  IMPORTS

  id-pkix-ocsp
    FROM OCSP-2009 -- From OCSP [RFC2560]
    { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp-02(48) }

  AlgorithmIdentifier{ }, SIGNATURE-ALGORITHM, PUBLIC-KEY
    FROM AlgorithmInformation-2009 -- From [NEWASN]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58) }

  EXTENSION
    FROM PKIX-CommonTypes-2009 -- From [NEWASN]
    { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) } ;

  -- Add re-preferred-signature-algorithms to the set of extensions
  -- for TBSRequest.requestExtensions

  re-preferred-signature-algorithms EXTENSION ::= {
    SYNTAX PreferredSignatureAlgorithms
    IDENTIFIED BY id-pkix-ocsp-pref-sig-algs  }

  id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

  PreferredSignatureAlgorithms ::= SEQUENCE OF
    PreferredSignatureAlgorithm

  PreferredSignatureAlgorithm ::= SEQUENCE {
    sigIdentifier AlgorithmIdentifier{SIGNATURE-ALGORITHM, {...}},
    certIdentifier AlgorithmIdentifier{PUBLIC-KEY, {...}}
    OPTIONAL  }

END
```

A.2 1988 ASN.1 Module

```
OCSP-AGILITY-88 { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-ocsp-agility-2009-88(67) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

    -- EXPORTS ALL;
    IMPORTS

    id-pkix-ocsp
    FROM OCSP {iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}

    AlgorithmIdentifier
    FROM PKIX1Explicit88 { iso(1) identified-organization(3) dod(6)
        internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
        id-pkix1-explicit(18) };

    id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

    PreferredSignatureAlgorithms ::= SEQUENCE OF
        PreferredSignatureAlgorithm

    PreferredSignatureAlgorithm ::= SEQUENCE {
        sigIdentifier AlgorithmIdentifier,
        certIdentifier AlgorithmIdentifier OPTIONAL
    }

END
```

Author's Address

Phillip Hallam-Baker
Default Deny Security

Email: phill@hallambaker.com

Stefan Santesson
3xA Security AB
Sweden

Email: sts@aaa-sec.com

Network Working Group
Internet-Draft
Obsoletes: 2560 (if approved)
Intended Status: Proposed Standard
Expires: April 17, 2011

D. Cooper
NIST
S. Santesson
3xA Security
October 14, 2010

X.509 Internet Public Key Infrastructure
Online Certificate Status Protocol - OCSP
<draft-ietf-pkix-rfc2560bis-02.txt>

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Abstract

This document specifies a protocol useful in determining the current status of a digital certificate without requiring CRLs.

Table of Contents

1.	Introduction	4
1.1.	Terminology	5
1.2.	OCSP Responder Architectures	5
1.2.1.	Integrated OCSP Responders	5
1.2.2.	Designated OCSP Responders	5
1.2.3.	Locally Trusted OCSP Responders	6
1.3.	Dynamic Versus Static Responses	6
2.	Detailed Protocol	6
2.1.	OCSP Requests	7
2.1.1.	Request Extensions	8
2.1.1.1.	Nonce	8
2.1.1.2.	Acceptable Response Types	9
2.1.1.3.	Preferred Signature Algorithms	9
2.1.2.	Service Locator Single Request Extension	10
2.2.	OCSP Responses	11
2.2.1.	Error Responses	11
2.3.	Basic Response Type	12
2.3.1.	Certificate Revocation Status	14
2.3.2.	thisUpdate, nextUpdate, and producedAt	15
2.3.3.	Nonce Response Extension	15
2.3.4.	Single Response Extensions	16
2.3.4.1.	CRL References	16
2.3.4.2.	Archive Cutoff	16
2.3.4.3.	Invalidity Date	17
2.3.5.	Response Signatures	17
2.3.5.1.	Authorized Responders	17
2.3.5.2.	Revocation Information for OCSP Responder's Certificate	18

2.3.5.3.	Responder Signature Algorithm Selection . . .	19
2.3.5.3.1.	Dynamic Response	19
2.3.5.3.2.	Static Response	20
2.3.6.	Response Verification	20
2.3.6.1.	Mandatory and Optional Cryptographic Algorithms	20
2.3.6.2.	Verifying Responder's Authorization	21
3.	OCSP Responder Discovery	21
4.	Security Considerations	22
4.1.	Use of Insecure Algorithms	23
4.2.	Man in the Middle Downgrade Attack	24
4.3.	Denial of Service Attack	24
5.	IANA Considerations	24
6.	Acknowledgments	24
7.	References	24
7.1.	Normative References	24
7.2.	Informative References	25
Appendix A.	Implementation Notes	25
Appendix B.	OCSP over HTTP	26
B.1.	Request	26
B.2.	Response	26
Appendix C.	ASN.1 Modules	26
C.1.	OCSP in ASN.1	27
C.2.	Preferred Signature Algorithms ASN.1	30
C.2.1.	ASN.1 Module	30
C.2.2.	1988 ASN.1 Module	31
Appendix D.	Media Types	32
D.1.	application/ocsp-request	32
D.2.	application/ocsp-response	33
Appendix E.	Example PKIs With OCSP Responders	34
E.1.	Integrated OCSP Responders	34
E.2.	Designated OCSP Responders	36
E.3.	Locally Trusted OCSP Responder	41
Author's Address	42

1. Introduction

In lieu of or as a supplement to checking against a periodic CRL, it may be necessary to obtain timely information regarding the revocation status of a certificate (cf. [RFC5280], Section 3.3). Examples include high-value funds transfer or large stock trades.

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate. OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response.

This protocol specifies the data that needs to be exchanged between an application checking the status of a certificate and the server providing that status.

An overview of the protocol is provided in this section. Details of the protocol are in Section 2. Section 3 specifies how information about the access location of an OCSP responder needs to be distributed. Security issues with the protocol are covered in Section 4. Appendix A includes some implementation notes, Appendix B defines OCSP over HTTP, Appendix C accumulates ASN.1 syntactic elements, Appendix D specifies the media types for the messages, and Appendix E provides some examples of architectures of public key infrastructures (PKI) that include OCSP responders.

This specification obsoletes [RFC2560]. The primary reason for the publication of this document is to address ambiguities that have been found since the publication of RFC 2560. This document differs from RFC 2560 in only a few areas:

- o Section 2.1.1.1 specifies the ASN.1 syntax for the nonce extension, which was missing in RFC 2560.
- o Section 2.1.1.3 specifies a new extension that may be included in a request message to specify signature algorithms the client would prefer the server use to sign the response.
- o Section 2.2.1 extends the use of the "unauthorized" error response, as specified in [RFC5019].
- o Section 2.3 states that a response may include revocation status information for certificates that were not included in the request, as permitted in [RFC5019].

- o Section 2.3.6.1 changes set of cryptographic algorithms that clients must support and the set of cryptographic algorithms that clients should support.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. OCSP Responder Architectures

In order for an OCSP client to accept a response from a responder, the responder MUST be authorized to provide the response. The responder may either be authorized by the certification authority (CA) that issued the certificate for which status information is being requested (the target certificate) or by the OCSP client itself. Authorized OCSP responders fall into one of three categories: integrated, designated, or locally trusted. The category into which an OCSP responder falls is based on the perspective of the OCSP client and depends upon the issuer of the target certificate.

1.2.1. Integrated OCSP Responders

When a CA provides revocation status information for its own certificates, it is an integrated OCSP responder. If a responder is an integrated OCSP responder then the subject distinguished name (DN) in the certificate containing the public key required to verify the signature on the OCSP response will be the same as the issuer DN in the target certificate(s). An integrated OCSP responder may sign certificates and OCSP responses using the same private key or may use different keys to sign certificates and OCSP responses.

1.2.2. Designated OCSP Responders

Most CAs do not act as OCSP responders themselves, but provide OCSP services by designating a separate OCSP responder as being authorized to provide revocation status information for the certificates that they issue. A responder is a designated OCSP responder if subject DN in the certificate containing the public key required to verify the signature on the OCSP response is not the same as the issuer DN in the target certificate(s), but the certificate was issued by the issuer of the target certificate(s) and includes a unique value in the extended key usage extension that indicates that the issuing CA has authorized the OCSP responder to provide revocation status information for the certificates that it has issued. Designated OCSP responders are frequently operated by the same organization as the CA(s) for which they provide revocation status information.

1.2.3. Locally Trusted OCSP Responders

Systems or applications that rely on OCSP responses MAY provide a means of locally configuring one or more OCSP signing authorities, and specifying the set of CAs for which each signing authority is trusted. Just as a relying party may choose which CAs to use as trust anchors for certification path validation, an OCSP client may choose to accept responses from an OCSP responder even if no CA has designated that responder as authorized to sign OCSP responses. Systems and applications may also be designed to only accept OCSP responses from integrated and designated OCSP responders.

Locally trusted OCSP responders are usually set up by an organization (or on behalf of an organization) for use by OCSP clients within that organization. The OCSP responder collects revocation information (e.g., certificate revocation lists (CRL)) from all CAs within a PKI and uses this information to generate OCSP responses for its clients.

1.3. Dynamic Versus Static Responses

OCSP responders may generate fresh OCSP responses for each OCSP request they receive, but they are also permitted to generate static responses in advance of a request. Pre-generating OCSP responses can improve efficiency, which may be necessary in high-volume environments [RFC5019]. Pre-generated responses will generally be the same as freshly generated responses, but can be identified since non-error response messages specify the time at which the response was generated. In addition, an OCSP responder that can only provide pre-generated responses may send an "unauthorized" error response when it does not have a pre-generated response that corresponds to the request.

2. Detailed Protocol

This section presents the OCSP protocol. Section 2.1 presents the format for OCSP requests, Section 2.2 presents the generic syntax for OCSP responses, and Section 2.3 presents the syntax for the basic response type. The ASN.1 syntax in this section imports terms defined in [RFC5280]. The terms imported from RFC 5280 are: AlgorithmIdentifier, Certificate, CertificateSerialNumber, CRLReason, Extensions, GeneralName, id-ad-ocsp, id-kp, Name, and SubjectPublicKeyInfo.

OCSP requests and responses MAY include extensions. Extensions in OCSP requests and responses are based on the extension model employed in X.509 version 3 certificates [RFC5280]. This document specifies some standard extensions that MAY appear in requests and/or responses. Additional extensions MAY be defined in additional RFCs.

Support for any specific extension is OPTIONAL for both clients and responders. The critical flag SHOULD NOT be set for any of the extensions defined in this document. Unrecognized extensions MUST be ignored (unless they have the critical flag set).

For signature calculation, the data to be signed is encoded using the ASN.1 distinguished encoding rules (DER) [X.690]. The actual formatting of the request and response messages could vary depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

ASN.1 EXPLICIT tagging is used as a default unless specified otherwise.

2.1. OCSP Requests

The ASN.1 for an OCSP request message is as follows:

```
OCSPRequest ::= SEQUENCE {
    tbsRequest          TBSRequest,
    optionalSignature [0] EXPLICIT Signature OPTIONAL }

TBSRequest ::= SEQUENCE {
    version              [0] EXPLICIT Version DEFAULT v1,
    requestorName        [1] EXPLICIT GeneralName OPTIONAL,
    requestList          SEQUENCE OF Request,
    requestExtensions [2] EXPLICIT Extensions OPTIONAL }

Signature ::= SEQUENCE {
    signatureAlgorithm    AlgorithmIdentifier,
    signature             BIT STRING,
    certs                 [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }

Version ::= INTEGER { v1(0) }

Request ::= SEQUENCE {
    reqCert              CertID,
    singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL }

CertID ::= SEQUENCE {
    hashAlgorithm          AlgorithmIdentifier,
    issuerNameHash         OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash          OCTET STRING, -- Hash of Issuer's public key
    serialNumber           CertificateSerialNumber }
```

An OCSP request contains the following data:

- o the protocol version, which MUST be v1 (value is 0) for this version of the protocol;

- o a list of the certificates for which status information is being requested; and
- o optional extensions, which MAY be processed by the OCSP responder.

Each certificate for which status information is being requested is identified by:

- o issuerNameHash - the hash of the issuer's distinguished name. The hash MUST be calculated over the DER encoding of the issuer's name field in the certificate being checked.
- o issuerKeyHash - the hash of the issuer's public key. The hash MUST be calculated over the value of the BIT STRING subjectPublicKey (excluding tag, length, and number of unused bits) in the issuer's certificate.
- o serialNumber - the serial number of the certificate for which status is being requested.

The hash algorithm used for both issuerNameHash and issuerKeyHash is identified in hashAlgorithm.

The requestor MAY choose to sign the OCSP request. In that case, the signature SHALL be computed over the DER encoding of tbsRequest. If the request is signed, the requestor MUST specify its name in the requestorName field. Also, for signed requests, the requestor MAY include certificates in the certs field of Signature that help the OCSP responder verify the requestor's signature. If no certificates are included then certs SHOULD be absent.

2.1.1. Request Extensions

This section defines some standard extensions that may appear in the requestExtensions field of an OCSP request message. For each extension, the definition indicates its syntax, processing performed by the OCSP responder, and any extensions that are included in the corresponding response.

2.1.1.1. Nonce

The nonce cryptographically binds a request and a response to prevent replay attacks. The nonce is included as one of the requestExtensions in requests, while in responses it would be included as one of the responseExtensions. In both the request and the response, the nonce will be identified by the object identifier id-pkix-ocsp-nonce, while the extnValue contains an OCTET STRING,

which is the value of the nonce.

```
id-pkix-ocsp          OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-nonce    OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }
```

```
Nonce ::= OCTET STRING
```

2.1.1.2. Acceptable Response Types

An OCSF client MAY wish to specify the kinds of response types it understands. To do so, it SHOULD use an extension with the OID id-pkix-ocsp-response, and the value AcceptableResponses. The OIDs included in AcceptableResponses are the OIDs of the various response types this client can accept (e.g., id-pkix-ocsp-basic).

```
id-pkix-ocsp-response OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }
```

```
AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER
```

As noted in Section 2.3, OCSF clients MUST be capable of receiving and processing responses of the id-pkix-ocsp-basic response type. Thus, when the id-pkix-ocsp-response extension is included in an OCSF request, the id-pkix-ocsp-basic OID MUST be included as one of the AcceptableResponses.

2.1.1.3. Preferred Signature Algorithms

A client MAY declare a preferred set of algorithms in a request by including a preferred signature algorithms extension.

```
id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }
```

```
PreferredSignatureAlgorithms ::= SEQUENCE OF
                                   PreferredSignatureAlgorithm
```

```
PreferredSignatureAlgorithm ::= SEQUENCE {
    sigIdentifier      AlgorithmIdentifier,
    certIdentifier     AlgorithmIdentifier OPTIONAL }
```

The syntax of AlgorithmIdentifier is defined in section 4.1.1.2 of [RFC5280].

sigIdentifier specifies the signature algorithm the client prefers, e.g., algorithm=ecdsa-with-sha256. Parameters are absent for most common signature algorithms. [Editor's note: This should be clarified with respect to RSA (both PKCS #1 v1.5 and PSS) where parameters are required.]

certIdentifier specifies the subject public key algorithm identifier the client prefers in the server's certificate used to validate the OCSF response, e.g., algorithm=id-ecPublicKey and parameters=secp256r1.

certIdentifier is OPTIONAL and provides a means to specify parameters necessary to distinguish among different usages of a particular algorithm, e.g., it may be used by the client to specify what curve it supports for a given elliptic curve algorithm.

The client MUST support each of the specified preferred signature algorithms and the client MUST specify the algorithms in the order of preference.

The server SHOULD use one of the preferred signature algorithms for signing OCSF responses to the requesting client.

2.1.2. Service Locator Single Request Extension

This document specifies one standard extension that may appear in the singleRequestExtensions field of an OCSF request message, the service locator extension.

An OCSF server may be operated in a mode whereby the server receives a request and routes it to the OCSF server that is known to be authoritative for the identified certificate. The serviceLocator request extension is defined for this purpose.

id-pkix-ocsp-service-locator OBJECT IDENTIFIER ::= { id-pkix-ocsp 7 }

ServiceLocator ::= SEQUENCE {
 issuer Name,
 locator AuthorityInfoAccessSyntax OPTIONAL }

Values for these fields are obtained from the corresponding fields in the subject certificate.

[Editor's note: I don't understand how this extension works. Is the client telling the OCSF server where to route the request? Can a single request include multiple certificates, each with a different value for the service locator field? If so, which responder signs the response? Is the response always signed by the server that directly received the request from the client, even though that server is just routing the request to the authoritative server(s)? Can anyone provide text for this section that provides more clarity on the semantics of this extension?]

2.2. OCSP Responses

The ASN.1 for a response message is as follows:

```
OCSPResponse ::= SEQUENCE {
    responseStatus      OCSPResponseStatus,
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED {
    successful          (0), -- Response has valid confirmations
    malformedRequest    (1), -- Illegal confirmation request
    internalError       (2), -- Internal error in issuer
    tryLater            (3), -- Try again later
                        -- (4) is not used
    sigRequired         (5), -- Must sign the request
    unauthorized        (6)  -- Request unauthorized
}

ResponseBytes ::= SEQUENCE {
    responseType        OBJECT IDENTIFIER,
    response             OCTET STRING }
```

Upon receipt of a request, an OCSP responder determines if:

1. the message is well formed;
2. the responder is configured to provide the requested service;
and
3. the request contains the information needed by the responder.

If any one of the prior conditions are not met, the OCSP responder produces an error message consisting of a responseStatus of "malformedRequest", "internalError", "tryLater", "sigRequired", or "unauthorized", and with responseBytes absent. If all of the prior conditions are met, the OCSP responder produces a response with a responseStatus of "successful" and with the responseBytes field present. The value for responseBytes consists of an OBJECT IDENTIFIER and a response syntax identified by that OID, encoded as an OCTET STRING.

2.2.1. Error Responses

An error response message consists of an OCSPResponse in which only the responseStatus field is present. These messages are not signed. For error responses, responseStatus MUST be one of the following:

- o `malformedRequest`: the request received does not conform to the OCSF syntax.
- o `internalError`: the OCSF responder reached an inconsistent internal state. The query should be retried, potentially with another responder.
- o `tryLater`: the OCSF responder is operational, but is temporarily unable to return a status for one or more of the requested certificates.
- o `sigRequired`: the request is unsigned, but the server requires the client to sign the request in order to construct a response.
- o `unauthorized`: the client is not authorized to make this query to this server or the server is not capable of responding authoritatively (cf. [RFC5019], Section 2.2.3).

2.3. Basic Response Type

While OCSF responses may be of various types, this document specifies only one response type, the basic response type, which MUST be supported by all OCSF servers and clients. This section describes the basic response type.

The basic response type is identified by the `id-pkix-ocsp-basic` OID:

`id-pkix-ocsp-basic` OBJECT IDENTIFIER ::= { `id-pkix-ocsp` 1 }

When the `responseType` is `id-pkix-ocsp-basic` the response field SHALL contain the DER encoding of `BasicOCSPResponse`:

```
BasicOCSPResponse ::= SEQUENCE {  
    tbsResponseData      ResponseData,  
    signatureAlgorithm    AlgorithmIdentifier,  
    signature             BIT STRING,  
    certs                 [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

The value for signature SHALL be computed on the DER encoding of `tbsResponseData`. The responder MAY include certificates in the `certs` field of `BasicOCSPResponse` that help the OCSF client verify the responder's signature. If no certificates are included then `certs` SHOULD be absent.

```
ResponseData ::= SEQUENCE {  
    version                [0] EXPLICIT Version DEFAULT v1,  
    responderID             ResponderID,
```



```
    producedAt          GeneralizedTime,
    responses            SEQUENCE OF SingleResponse,
    responseExtensions  [1] EXPLICIT Extensions OPTIONAL }

ResponderID ::= CHOICE {
    byName              [1] Name,
    byKey               [2] KeyHash }

KeyHash ::= OCTET STRING -- SHA-1 hash of responder's public key
                        -- (i.e., the SHA-1 hash of the value of the
                        -- BIT STRING subjectPublicKey [excluding
                        -- the tag, length, and number of unused
                        -- bits] in the responder's certificate)

SingleResponse ::= SEQUENCE {
    certID              CertID,
    certStatus          CertStatus,
    thisUpdate          GeneralizedTime,
    nextUpdate          [0] EXPLICIT GeneralizedTime OPTIONAL,
    singleExtensions    [1] EXPLICIT Extensions OPTIONAL }

CertStatus ::= CHOICE {
    good                [0] IMPLICIT NULL,
    revoked             [1] IMPLICIT RevokedInfo,
    unknown             [2] IMPLICIT UnknownInfo }

RevokedInfo ::= SEQUENCE {
    revocationTime      GeneralizedTime,
    revocationReason    [0] EXPLICIT CRLReason OPTIONAL }

UnknownInfo ::= NULL
```

The basic response type contains:

- o the version of the response syntax, which MUST be v1 (value is 0) for this version of the basic response syntax;
- o either the name of the responder or a hash of the responder's public key;
- o the time at which the response was generated;
- o responses for each of the certificates in a request;
- o optional extensions;
- o a signature computed across a hash of the response; and

- o the signature algorithm OID.

The responder MAY include certificates in the certs field of BasicOCSPResponse that help the OCSP client verify the responder's signature.

The response for each of the certificates in a request consists of:

- o an identifier of the certificate for which revocation status information is being provided (i.e., the target certificate);
- o the revocation status of the certificate (good, revoked, or unknown);
- o the validity interval of the response; and
- o optional extensions.

The response MUST include a SingleResponse for each certificate in the request and SHOULD NOT include any additional SingleResponse elements. OCSP responders that pre-generate status responses MAY return responses that include additional SingleResponse elements if necessary to improve response pre-generation performance or cache efficiency. [Editor's note: From Section 2.2.1 of RFC 5019.]

2.3.1. Certificate Revocation Status

For each of the certificates identified, the response message MUST indicate a status of "good", "revoked", or "unknown".

A status of "good" indicates that the certificate has not been revoked. It does not indicate that the certificate was ever issued or that it is in its validity interval.

A status of "revoked" indicates that the certificate has been revoked (either permanently or temporarily (on hold)). When the "revoked" status is returned the response MUST indicate the time at which revocation occurred and MAY indicate the reason for the revocation.

If an OCSP responder knows that a particular CA's private key has been compromised, it MAY return the "revoked" status for all certificates issued by that CA.

A status of "unknown" indicates that the responder doesn't know about the certificate being requested.

Response extensions may be used to convey additional information on assertions made by the responder regarding the status of the

certificate such as a positive statement about issuance, expiry, etc.

2.3.2. thisUpdate, nextUpdate, and producedAt

Responses can contain three times in them - thisUpdate, nextUpdate, and producedAt. The semantics of these fields are:

- o thisUpdate: The time at which the status being indicated is known to be correct.
- o nextUpdate: The time at or before which newer information will be available about the status of the certificate.
- o producedAt: The time at which the OCSF responder signed this response.

The thisUpdate and nextUpdate fields define a recommended validity interval. This interval corresponds to the {thisUpdate, nextUpdate} interval in CRLs. Responses whose thisUpdate time is later than the local system time SHOULD be considered unreliable. Responses whose nextUpdate value is earlier than the local system time value SHOULD be considered unreliable. If nextUpdate is not set, the responder is indicating that newer revocation information is available all the time.

OCSF responders MAY pre-produce signed responses specifying the status of certificates at a specified time. The time at which the status was known to be correct SHALL be reflected in the thisUpdate field of the response. The time at or before which newer information will be available is reflected in the nextUpdate field, while the time at which the response was produced will appear in the producedAt field of the response.

2.3.3. Nonce Response Extension

This document defines one standard extension that may appear in the responseExtensions field of OCSF response messages. The nonce cryptographically binds a request and a response to prevent replay attacks. The nonce extension in the response is identified using the same object identifier as is used in the request, id-pkix-ocsp-nonce.

If the request included a nonce extension (Section 2.1.1.1) then the response MUST either omit the nonce extension or include a nonce extension that has the same value as the nonce extension in the request.

An OCSF responder MAY include a nonce extension in a response even if no nonce extension was included in the corresponding request.

Clients that do not include a nonce in the request MUST ignore any nonce that may be present in the response.

2.3.4. Single Response Extensions

This section defines some standard extensions that may appear in the singleExtensions field of OCSF response messages. Each extension in this section provides additional information about a single certificate in the response.

2.3.4.1. CRL References

It may be desirable for the OCSF responder to indicate the CRL on which a revoked or on hold certificate is found. This can be useful where OCSF is used between repositories, and also as an auditing mechanism. The CRL may be specified by a URL (the URL at which the CRL is available), a number (CRL number), or a time (the time at which the relevant CRL was created). The identifier for this extension will be id-pkix-ocsp-crl, while the value will be CrlID.

id-pkix-ocsp-crl OBJECT IDENTIFIER ::= { id-pkix-ocsp 3 }

CrlID ::= SEQUENCE {
 crlUrl [0] EXPLICIT IA5String OPTIONAL,
 crlNum [1] EXPLICIT INTEGER OPTIONAL,
 crlTime [2] EXPLICIT GeneralizedTime OPTIONAL }

For crlUrl, the IA5String will specify the URL at which the CRL is available. For crlNum, the INTEGER will specify the value of the CRL number extension of the relevant CRL. For crlTime, the GeneralizedTime will indicate the time at which the relevant CRL was issued.

2.3.4.2. Archive Cutoff

An OCSF responder MAY choose to retain revocation information beyond a certificate's expiration. The date obtained by subtracting this retention interval value from the producedAt time in a response is defined as the certificate's "archive cutoff" date.

OCSF-enabled applications would use an OCSF archive cutoff date to contribute to a proof that a digital signature was (or was not) reliable on the date it was produced even if the certificate needed to validate the signature has long since expired.

OCSF responders that provide support for such historical reference SHOULD include an archive cutoff date extension in responses. The identifier for this extension will be id-pkix-ocsp-archive-cutoff,

while the value will be ArchiveCutoff.

id-pkix-ocsp-archive-cutoff OBJECT IDENTIFIER ::= { id-pkix-ocsp 6 }

ArchiveCutoff ::= GeneralizedTime

To illustrate, if a server is operated with a 7-year retention interval policy and status was produced at time t1 then the value for ArchiveCutoff in the response would be (t1 - 7 years).

Note that when an OCSP response includes status information for more than one certificate, the server may choose to include the archive cutoff extension for only some of the certificates and may specify different ArchiveCutoff values for different certificates.

2.3.4.3. Invalidity Date

For certificates with a revocation status of "revoked", OCSP responders may include the invalidity date CRL entry extension, as described in Section 5.3.2 of [RFC5280], in singleExtensions to provide the date on which it is known or suspected that the private key was compromised or that the the certificate otherwise became invalid.

2.3.5. Response Signatures

All responses of the basic response type MUST be digitally signed. This section addresses the means by which OCSP responders determine which key and signature algorithm to use to ensure that OCSP clients can verify the signatures on OCSP response messages.

2.3.5.1. Authorized Responders

As described in Section 1.2, in order for an OCSP client to accept a response from a responder for which it is not locally configured to accept OCSP responses, the responder MUST be authorized by the issuer of the target certificate(s). This section provides additional information about the requirements for a CA to designate an OCSP responder as authorized to provide revocation status information for the certificates that it issues.

o Integrated OCSP responder

CAs that sign OCSP responses implicitly authorize themselves to provide revocation status information for the certificates that they issue. When a CA signs OCSP responses to provide revocation status information for its own certificates, it may sign the responses using the same key as was used to sign the target

certificate(s) or using a different key. One reason that a CA may sign an OCSF response using a different key would be if the CA had performed a key rollover since it issued the target certificate(s).

Note: Some OCSF clients, when accepting responses from an integrated OCSF responder, will only accept responses that are signed using the same key as the target certificate(s).

o Designated OCSF responder

In order for a CA to designate an OCSF responder other than itself as authorized to provide revocation status information for the certificates that it issues, the CA MUST issue a certificate to the OCSF responder that:

- o contains the public key required to validate the signatures on OCSF responses signed by the responder; and
- o has an extended key usage extension that includes the id-kp-OCSPSigning OID.

id-kp-OCSPSigning OBJECT IDENTIFIER ::= { id-kp 9 }

The CA does not need to sign this certificate using the same key as was used to sign the target certificate(s), but the certificate MUST be issued to the OCSF responder directly by the CA that issued the target certificate(s).

Note: Some OCSF clients, when accepting responses from a designated OCSF responder, will only accept responses if the certificate required to validate the signature on the response was signed using the same key as the target certificate(s).

2.3.5.2. Revocation Information for OCSF Responder's Certificate

Since an authorized OCSF responder provides revocation status information for one or more CAs, OCSF clients need to know how to check that an authorized responder's certificate has not been revoked. CAs may choose to deal with this problem in one of three ways:

- o A CA may specify that an OCSF client can trust a responder for the lifetime of the responder's certificate. The CA does so by including the extension id-pkix-ocsp-nocheck. This SHOULD be a non-critical extension. The value of the extension SHALL be NULL. CAs issuing such a certificate should realize that a compromise of the responder's key is as serious as the compromise of a CA key

used to sign CRLs, at least for the validity period of this certificate. CA's may choose to issue this type of certificate with a very short lifetime and renew it frequently.

[Editor's note: I changed "should be NULL" to "SHALL be NULL".]

id-pkix-ocsp-nocheck OBJECT IDENTIFIER ::= { id-pkix-ocsp 5 }

- o A CA may specify how to check the responder's certificate for revocation. This can be done using CRL Distribution Points if the check should be done using CRLs, or Authority Information Access if the check should be done in some other way. Details for specifying either of these two mechanisms are available in [RFC5280].
- o A CA may choose not to specify any method of revocation checking for the responder's certificate, in which case it would be up to the OCSP client's local security policy to decide whether that certificate should be checked for revocation.

2.3.5.3. Responder Signature Algorithm Selection

This section describes rules for the OCSP responder to use to select the algorithm to use to sign the response.

2.3.5.3.1. Dynamic Response

A responder MAY maximize the potential for ensuring interoperability by selecting a supported signature algorithm using the following order of precedence, as long as the selected algorithm meets all security requirements of the OCSP responder, where the first method has the highest precedence:

1. Select an algorithm specified as a preferred signature algorithm in the client request.
2. Select the signature algorithm used to sign a CRL issued by the certificate issuer providing status information for the certificate specified by CertID.
3. Select the signature algorithm used to sign the OCSP request.
4. Select a signature algorithm that has been advertised as being the default signature algorithm for the signing service using an out-of-band mechanism.
5. Select a mandatory or recommended signing algorithm specified for the version of the OCSP protocol in use (see Section 2.3.6.1).

A responder SHOULD always apply the lowest numbered selection mechanism that is known, supported, and that meets the responder's criteria for cryptographic algorithm strength.

2.3.5.3.2. Static Response

For purposes of efficiency, an OCSP responder is permitted to generate static responses in advance of a request. The case may not permit the responder to make use of the client request data during the response generation. However, the responder SHOULD still use the client request data during the selection of the pre-generated response to be returned. Responders MAY use the historical client requests as part of the input to the decisions of what different algorithms should be used to sign the pre-generated responses.

2.3.6. Response Verification

Prior to accepting a response of the basic response type as valid, OCSP clients MUST confirm that:

1. the certificates identified in the response correspond to the certificates that were identified in the corresponding request;
2. the signature on the response is valid;
3. the signer is currently authorized to sign the response (see Section 2.3.6.2);
4. the time at which the status being indicated is known to be correct (thisUpdate) is sufficiently recent; and
5. when nextUpdate is set in the response, it is greater than the current time.

2.3.6.1. Mandatory and Optional Cryptographic Algorithms

Clients that request OCSP services MUST be capable of processing responses signed using RSA with SHA-1 (identified by the sha1WithRSAEncryption OID specified in [RFC3279]) and RSA with SHA-256 (identified by the sha256WithRSAEncryption OID specified in [RFC4055]). Clients SHOULD also be capable of processing responses signed using DSA keys (identified by the id-dsa-with-sha1 OID specified in [RFC3279]). Clients MAY support other algorithms.

2.3.6.2. Verifying Responder's Authorization

Systems or applications that rely on OCSP responses MUST reject a response if the certificate required to validate the signature on the response fails to meet at least one of the following criteria:

1. Matches a local configuration of OCSP signing authority for the certificate in question (i.e., is signed by a locally trusted OCSP responder).
2. Is the certificate of the CA that issued the certificate in question. (For this criterion to be satisfied, the subject DN in the certificate required to validate the signature on the OCSP response MUST be the same as the issuer DN in the certificate in question.)
3. Has an extended key usage extension that includes the id-kp-OCSPSigning OID and is issued by the CA that issued the certificate in question. (For this criterion to be satisfied, the issuer DN in the certificate required to validate the signature on the OCSP response MUST be the same as the issuer DN in the certificate in question.)

Systems or applications that rely on OCSP responses MUST be capable of detecting and enforcing use of the id-kp-OCSPSigning value as described in Section 2.3.5.1. They MAY provide a means of locally configuring one or more OCSP signing authorities, and specifying the set of CAs for which each signing authority is trusted.

Additional acceptance or rejection criteria may apply to either the response itself or to the certificate used to validate the signature on the response.

3. OCSP Responder Discovery

CAs that operate as integrated OCSP responders or that designate one or more OCSP responders as authorized to provide revocation status information for the certificates that they issue MUST include an authorityInfoAccess extension (defined in [RFC5280], Section 4.2.2.1) in certificates that can be checked using OCSP. The authorityInfoAccess extension MUST include an entry with an accessMethod of id-ad-ocsp and an accessLocation that is a uniformResourceIdentifier (URI). The value of the accessLocation field in the certificate defines the transport (e.g., HTTP) used to access the OCSP responder and may contain other transport dependent information (e.g., a URL).

Information about the access location of OCSP responders that are neither integrated nor designated SHOULD NOT be included in the authorityInfoAccess extension of certificates. Instead, the access location for such OCSP responders SHOULD be provided using out-of-band means and be configured locally at the OCSP client.

[Editor's note: This text was based on my original interpretation of Section 3.1 in RFC 2560. However, emails from 1999 (<http://www.ietf.org/mail-archive/web/pkix/current/msg25976.html>) imply that the text was only intended to impose a requirement on CA products, not to require CAs to include the extension under certain circumstances. Was the second paragraph in Section 3.1 of RFC 2560 only intended to impose a requirement on CA products?]

4. Security Considerations

For this service to be effective, certificate using systems must connect to the certificate status service provider. In the event such a connection cannot be obtained, certificate-using systems could implement CRL processing logic as a fall-back position.

A denial of service vulnerability is evident with respect to a flood of queries. The production of a cryptographic signature significantly affects response generation cycle time, thereby exacerbating the situation. Unsigned error responses open up the protocol to another denial of service attack, where the attacker sends false error responses.

The use of pre-computed responses allows replay attacks in which an old (good) response is replayed prior to its expiration date but after the certificate has been revoked. Deployments of OCSP should carefully evaluate the benefit of pre-computed responses against the probability of a replay attack and the costs associated with its successful execution.

Requests do not contain the responder to which they are directed. This allows an attacker to replay a request to any number of OCSP responders.

While X.509 mandates that names be unambiguous, there is a risk that two unrelated authorities will issue certificates and/or OCSP responses under the same name. As a means of reducing problems and security issues related to issuer name collisions, CA and OCSP responder names SHOULD be formed in a way that reduces the likelihood of name collisions. Implementers should take into account the possible existence of multiple unrelated CAs and OCSP responders with the same name. At a minimum, implementations validating OCSP responses MUST ensure that the certification path of a target

certificate and the certification path of the certificate required to validate the signature on the OCSP response terminate at the same trust anchor, except in the case that the OCSP response was signed by a locally trusted OCSP responder.

The reliance of HTTP caching in some deployment scenarios may result in unexpected results if intermediate servers are incorrectly configured or are known to possess cache management faults. Implementers are advised to take the reliability of HTTP cache mechanisms into account when deploying OCSP over HTTP.

The mechanism used to choose the response signing algorithm **MUST** be considered to be sufficiently secure against cryptanalytic attack for the intended application.

In most applications it is sufficient for the signing algorithm to be at least as secure as the signing algorithm used to sign the original certificate whose status is being queried. However, this criteria may not hold in long-term archival applications in which the status of a certificate is being queried for a date in the distant past, long after the signing algorithm has ceased being considered trustworthy.

4.1. Use of Insecure Algorithms

It is not always possible for a responder to generate a response that the client is expected to understand and that meets contemporary standards for cryptographic security. In such cases an OCSP responder operator **MUST** balance the risk of employing a compromised security solution and the cost of mandating an upgrade, including the risk that the alternative chosen by end users will offer even less security or no security.

In archival applications it is quite possible that an OCSP responder might be asked to report the validity of a certificate on a date in the distant past. Such a certificate might employ a signing method that is no longer considered acceptably secure. In such circumstances the responder **MUST NOT** generate a signature for a signing mechanism that is considered unacceptably insecure.

A client **MUST** accept any signature algorithm in a response that it specified as a preferred signature algorithm in the request. It follows, therefore, that a client **MUST NOT** specify as a preferred signature algorithm any algorithm that is either not supported or not considered acceptably secure.

4.2. Man in the Middle Downgrade Attack

The mechanism to support client indication of preferred signature algorithms is not protected against a man in the middle downgrade attack. This constraint is not considered to be a significant security concern since the OCSP responder MUST NOT sign OCSP responses using weak algorithms even if requested by the client. In addition, the client can reject OCSP responses that do not meet its own criteria for acceptable cryptographic security no matter what mechanism is used to determine the signature algorithm of the response.

4.3. Denial of Service Attack

Algorithm agility mechanisms defined in this document introduce a slightly increased attack surface for denial of service attacks where the client request is altered to require algorithms that are not supported by the server or that do not match pre-generated responses.

5. IANA Considerations

Request types and extensions in OCSP requests and responses are identified using object identifiers. The objects are identified in an arc delegated by IANA to the PKIX Working Group. No further action by IANA is necessary for this document or any anticipated updates.

6. Acknowledgments

TBD

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3279] Polk, W., Housley, R., and L. Bassham, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.
- [RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [X.690] ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

7.2. Informative References

- [RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC5019] Deacon, A. and R. Hurst, "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments", RFC 5019, September 2007.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509", RFC 5912, June 2010.

Appendix A. Implementation Notes

RFC 2560 did not specify a syntax for the nonce extension. As a result, some OCSP clients and responders may not populate extnValue for the nonce extension with a DER encoded OCTET STRING. OCSP responders that support the nonce extension should be prepared to accept such values from OCSP clients and should place the exact same value in the extnValue of the nonce extension in the response as appeared in the request. OCSP clients that do not include a nonce extension in a request should be prepared to accept responses that include nonce extensions in which extnValue does not contain a DER encoded OCTET STRING.

OCSP does not provide a mechanism for an OCSP responder to indicate whether it supports the nonce extension. If an OCSP responder that provides fresh responses to every request that includes a nonce only includes a nonce in responses corresponding to requests that include a nonce then an attacker could obtain a response without a nonce and then later replay it to a client that included a nonce in its

request. The client may be unable to distinguish this replayed response from a response received directly from a responder that does not support the nonce extension. A responder that supports the nonce extension may protect against this by including a randomly generated nonce in any response that corresponds to a request that did not include a nonce extension.

Appendix B. OCSP over HTTP

This appendix describes the formatting that will be done to the request and response to support HTTP [RFC2616].

B.1. Request

HTTP-based OCSP requests can use either the GET or the POST method to submit their requests. To enable HTTP caching, small requests (that after encoding are less than 255 bytes) MAY be submitted using GET. If HTTP caching is not important, or the request is greater than 255 bytes, the request SHOULD be submitted using POST. Where privacy is a requirement, OCSP transactions exchanged using HTTP MAY be protected using either TLS/SSL or some other lower layer protocol.

An OCSP request using the GET method is constructed as follows:

```
GET {url}/{url-encoding of base-64 encoding of the DER encoding of
the OCSPRequest}
```

where {url} may be derived from the value of AuthorityInfoAccess or other local configuration of the OCSP client.

An OCSP request using the POST method is constructed as follows: The Content-Type header has the value "application/ocsp-request" while the body of the message is the binary value of the DER encoding of the OCSPRequest.

B.2. Response

An HTTP-based OCSP response is composed of the appropriate HTTP headers, followed by the binary value of the DER encoding of the OCSPResponse. The Content-Type header has the value "application/ocsp-response". The Content-Length header SHOULD specify the length of the response. Other HTTP headers MAY be present and MAY be ignored if not understood by the requestor.

Appendix C. ASN.1 Modules

This appendix includes the ASN.1 modules for OCSP. Appendix C.1 includes an ASN.1 module that conforms to the 1998 version of ASN.1

for all syntax elements of OCSP other than the preferred signature algorithms extension. An alternative to this module that conforms to the 2002 version of ASN.1 may be found in Section 4 of [RFC5912]. Appendix C.2 includes two ASN.1 modules for the preferred signature algorithms extension, one that conforms to the 1998 version of ASN.1 and one that conforms to the 2002 version of ASN.1.

C.1. OCSP in ASN.1

```
OCSP {iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}
```

```
DEFINITIONS EXPLICIT TAGS ::=
```

```
BEGIN
```

```
IMPORTS
```

```
-- PKIX Certificate Extensions
AuthorityInfoAccessSyntax, CRLReason, GeneralName
FROM PKIX1Implicit88 { iso(1) identified-organization(3)
                      dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                      id-mod(0) id-pkix1-implicit(19) }

Name, CertificateSerialNumber, Extensions,
id-kp, id-ad-ocsp, Certificate, AlgorithmIdentifier
FROM PKIX1Explicit88 { iso(1) identified-organization(3)
                      dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                      id-mod(0) id-pkix1-explicit(18) };
```

```
OCSPRequest ::= SEQUENCE {
    tbsRequest          TBSRequest,
    optionalSignature   [0] EXPLICIT Signature OPTIONAL }
```

```
TBSRequest ::= SEQUENCE {
    version              [0] EXPLICIT Version DEFAULT v1,
    requestorName        [1] EXPLICIT GeneralName OPTIONAL,
    requestList          SEQUENCE OF Request,
    requestExtensions    [2] EXPLICIT Extensions OPTIONAL }
```

```
Signature ::= SEQUENCE {
    signatureAlgorithm    AlgorithmIdentifier,
    signature             BIT STRING,
    certs                 [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
```

```
Version ::= INTEGER { v1(0) }
```

```
Request ::= SEQUENCE {
    reqCert                CertID,
    singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL }

CertID ::= SEQUENCE {
    hashAlgorithm           AlgorithmIdentifier,
    issuerNameHash          OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash           OCTET STRING, -- Hash of Issuers public key
    serialNumber            CertificateSerialNumber }

OCSPResponse ::= SEQUENCE {
    responseStatus          OCSPResponseStatus,
    responseBytes           [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED {
    successful              (0), -- Response has valid confirmations
    malformedRequest        (1), -- Illegal confirmation request
    internalError           (2), -- Internal error in issuer
    tryLater                (3), -- Try again later
                           -- (4) is not used
    sigRequired             (5), -- Must sign the request
    unauthorized            (6)  -- Request unauthorized
}

ResponseBytes ::= SEQUENCE {
    responseType           OBJECT IDENTIFIER,
    response               OCTET STRING }

BasicOCSPResponse ::= SEQUENCE {
    tbsResponseData        ResponseData,
    signatureAlgorithm      AlgorithmIdentifier,
    signature              BIT STRING,
    certs                  [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }

ResponseData ::= SEQUENCE {
    version                 [0] EXPLICIT Version DEFAULT v1,
    responderID            ResponderID,
    producedAt             GeneralizedTime,
    responses              SEQUENCE OF SingleResponse,
    responseExtensions      [1] EXPLICIT Extensions OPTIONAL }

ResponderID ::= CHOICE {
    byName                 [1] Name,
    byKey                   [2] KeyHash }
```



```
KeyHash ::= OCTET STRING --SHA-1 hash of responder's public key
-- (i.e., the SHA-1 hash of the value of the
-- BIT STRING subjectPublicKey [excluding
-- the tag, length, and number of unused
-- bits] in the responder's certificate)
```

```
SingleResponse ::= SEQUENCE {
    certID                CertID,
    certStatus            CertStatus,
    thisUpdate            GeneralizedTime,
    nextUpdate            [0] EXPLICIT GeneralizedTime OPTIONAL,
    singleExtensions      [1] EXPLICIT Extensions OPTIONAL }
```

```
CertStatus ::= CHOICE {
    good                [0] IMPLICIT NULL,
    revoked             [1] IMPLICIT RevokedInfo,
    unknown             [2] IMPLICIT UnknownInfo }
```

```
RevokedInfo ::= SEQUENCE {
    revocationTime      GeneralizedTime,
    revocationReason    [0] EXPLICIT CRLReason OPTIONAL }
```

```
UnknownInfo ::= NULL
```

```
ArchiveCutoff ::= GeneralizedTime
```

```
AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER
```

```
ServiceLocator ::= SEQUENCE {
    issuer              Name,
    locator             AuthorityInfoAccessSyntax }
```

```
-- Object Identifiers
```

```
id-kp-OCSPSigning      OBJECT IDENTIFIER ::= { id-kp 9 }
id-pkix-ocsp            OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-basic      OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }
id-pkix-ocsp-nonce      OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }
id-pkix-ocsp-crl        OBJECT IDENTIFIER ::= { id-pkix-ocsp 3 }
id-pkix-ocsp-response   OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }
id-pkix-ocsp-nocheck    OBJECT IDENTIFIER ::= { id-pkix-ocsp 5 }
id-pkix-ocsp-archive-cutoff OBJECT IDENTIFIER ::= { id-pkix-ocsp 6 }
id-pkix-ocsp-service-locator OBJECT IDENTIFIER ::= { id-pkix-ocsp 7 }
```

```
END
```

C.2. Preferred Signature Algorithms ASN.1

C.2.1. ASN.1 Module

```
OCSP-AGILITY-2009 { iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-ocsp-agility-2009-93(66) }
```

```
DEFINITIONS EXPLICIT TAGS ::=
BEGIN
```

```
EXPORTS ALL;    -- export all items from this module
IMPORTS
```

```
  id-pkix-ocsp
  FROM OCSP-2009 -- From [RFC5912]
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp-02(48) }
```

```
  AlgorithmIdentifier{ }, SIGNATURE-ALGORITHM, PUBLIC-KEY
  FROM AlgorithmInformation-2009 -- From [RFC5912]
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58) }
```

```
  EXTENSION
  FROM PKIX-CommonTypes-2009 -- From [RFC5912]
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) } ;
```

```
-- Add re-preferred-signature-algorithms to the set of extensions
-- for TBSRequest.requestExtensions
```

```
re-preferred-signature-algorithms EXTENSION ::= {
  SYNTAX PreferredSignatureAlgorithms
  IDENTIFIED BY id-pkix-ocsp-pref-sig-algs }
```

```
id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }
```

```
PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm
```

```
PreferredSignatureAlgorithm ::= SEQUENCE {
  sigIdentifier AlgorithmIdentifier{SIGNATURE-ALGORITHM, {...}},
  certIdentifier AlgorithmIdentifier{PUBLIC-KEY, {...}} OPTIONAL
}
```

```
END
```

C.2.2. 1988 ASN.1 Module

```
OCSP-AGILITY-88 { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-ocsp-agility-2009-88(67) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL;
IMPORTS

    id-pkix-ocsp
FROM OCSP {iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}

    AlgorithmIdentifier
FROM PKIX1Explicit88 { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
    id-pkix1-explicit(18) };

id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm

PreferredSignatureAlgorithm ::= SEQUENCE {
    sigIdentifier    AlgorithmIdentifier,
    certIdentifier   AlgorithmIdentifier OPTIONAL }

END
```

Appendix D. Media Types

D.1. application/ocsp-request

Type name: application

Subtype name: ocsp-request

Required parameters: None

Optional parameters: None

Encoding considerations: binary

Security considerations: Carries a request for information. This request may optionally be cryptographically signed.

Interoperability considerations: None

Published specification: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP

Applications that use this media type: OCSP clients

Additional information:

 Magic number(s): None

 File extension(s): .ORQ

 Macintosh File Type Code(s): none

Person & email address to contact for further information:

 Ambarish Malpani <ambarish@yahoo.com>

Intended usage: COMMON

Restrictions on usage: none

Author:

 Ambarish Malpani <ambarish@yahoo.com>

Change controller:

 The IESG <iesg@ietf.org>

D.2. application/ocsp-response

Type name: application

Subtype name: ocsp-response

Required parameters: None

Optional parameters: None

Encoding considerations: binary

Security considerations: Carries a cryptographically signed response.

Interoperability considerations: None

Published specification: X.509 Internet Public Key Infrastructure
Online Certificate Status Protocol - OCSF

Applications which use this media type: OCSF servers

Additional information:

 Magic number(s): None

 File extension(s): .ORS

 Macintosh File Type Code(s): none

Person & email address to contact for further information:

 Ambarish Malpani <ambarish@yahoo.com>

Intended usage: COMMON

Restrictions on usage: none

Author:

 Ambarish Malpani <ambarish@yahoo.com>

Change controller:

 The IESG <iesg@ietf.org>

Appendix E. Example PKIs With OCSP Responders

This appendix provides some examples of valid PKI architectures that include OCSP responders.

E.1. Integrated OCSP Responders

The examples in this appendix involve integrated OCSP responders. None of the certificates depicted in the examples in this appendix include an extended key usage extension that asserts the `id-kp-OCSPSigning` OID.

Figure 1 depicts a scenario in which the OCSP response is signed using the same key as the target certificate.

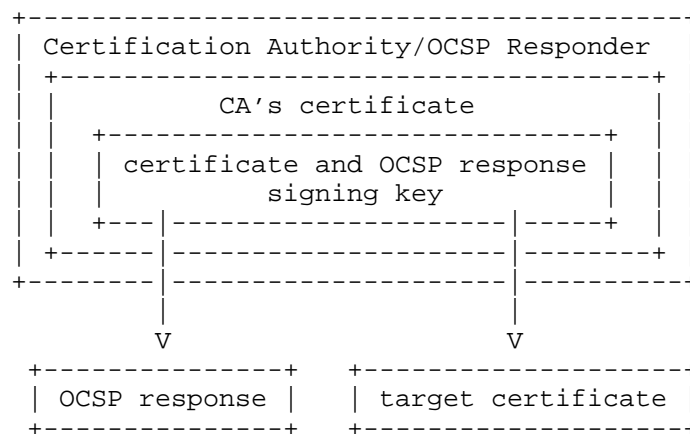


Figure 1. Integrated OCSP Responder With One Certificate and OCSP Response Signing Key

Figures 2 and 3 depict scenarios in which the OCSP response is signed using a different key than the one used to sign the target certificate. In each case, the CA has performed a key rollover since it issued the target certificate, and is using its new key to sign OCSP responses. In Figure 2, the CA has used its new private key to sign a self-issued certificate that contains its old public key. In Figure 3, the CA has been issued a new certificate from the root CA, while the certificate from the root CA certifying the old public key remains valid.

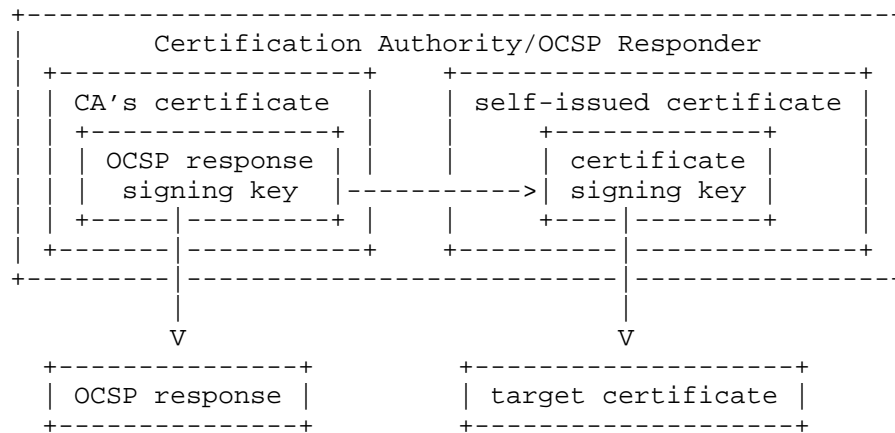


Figure 2. Integrated OCSP Responder With a Self-issued Key Rollover Certificate

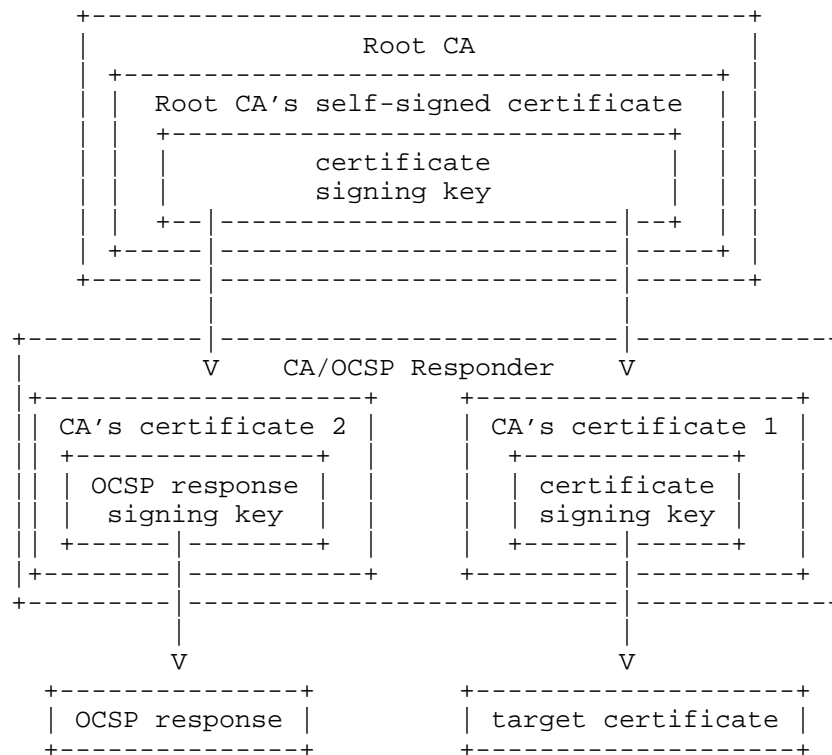


Figure 3. Integrated OCSP Responder With a Separate Key Certified by Root CA

E.2. Designated OCSF Responders

The examples in this appendix involve designated OCSF responders. Certificates that are marked with "***" include an extended key usage extension with the id-kp-OCSPSigning OID.

Figure 4 depicts a scenario in which the OCSF responder's certificate is signed using the same key as the target certificate.

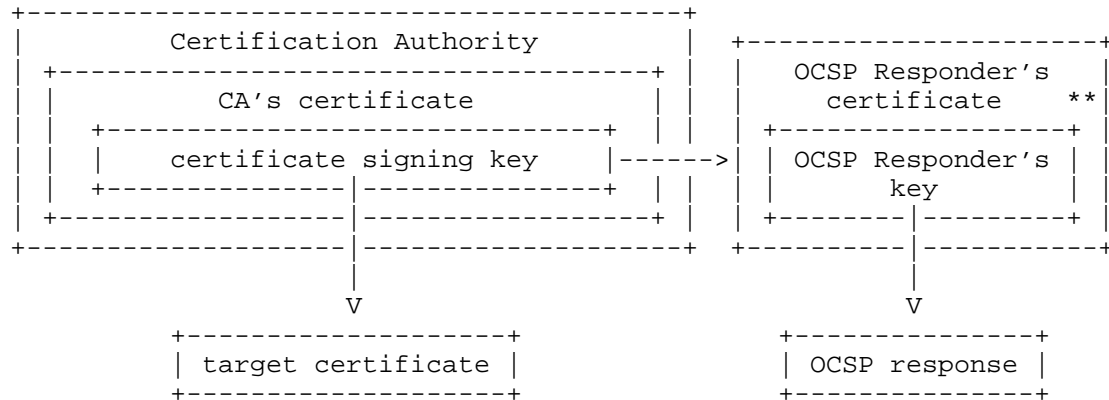


Figure 4. Designated OCSF Responder in Which CA Has One Certificate Signing Key

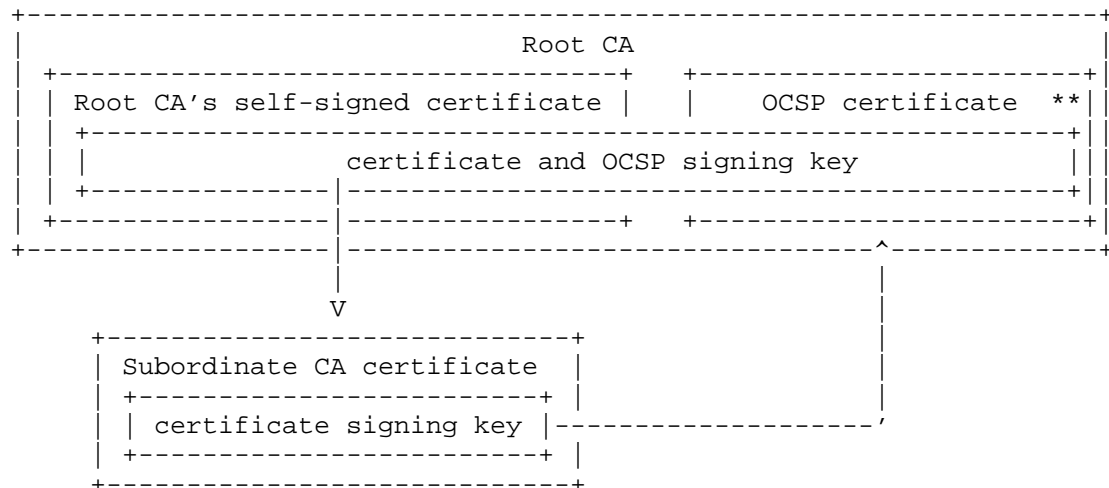


Figure 5. Integrated and Designated OCSF Responder

Figure 5 depicts a hierarchical PKI in which the root CA uses its key to sign OCSF responses for both the certificates that it issues and certificates issued by the subordinate CA. The subordinate CA has designated the root CA as authorized to provide revocation status information for the certificates that it issues by issuing a certificate to the root CA with an extended key usage extension that includes the id-kp-OCSPSigning OID. When an OCSF client is verifying a response from the OCSF responder that provides status information for a target certificate issued by the root CA the responder is an integrated OCSF responder. When an OCSF client is verifying a response from the OCSF responder that provides status information a target certificate issued by the subordinate CA the responder is a designated OCSF responder.

Figure 6 depicts a scenario in which the OCSF responder's certificate is signed using a different key than the one used to sign the target certificate. The CA has performed a key rollover since it issued the target certificate, and the OCSF responder's certificate was signed using the CA's new private key. The CA has used its new private key to sign a self-issued certificate that contains its old public key.

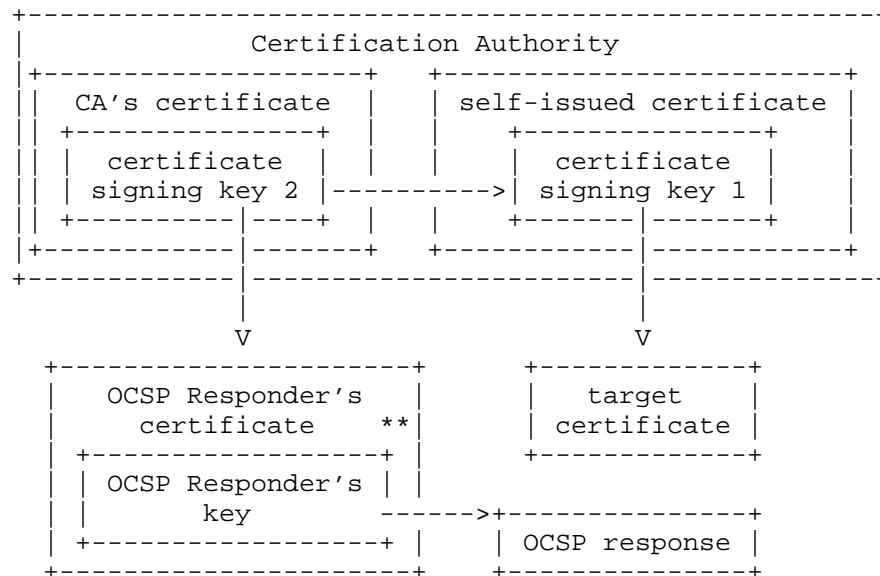


Figure 6. Designated OCSF Responder and CA with a Self-issued Key Rollover Certificate

Figure 7 depicts another scenario in which the OCSF responder's certificate is signed using a different key than the one used to sign

the target certificate. As in Figure 6, the CA has performed a key rollover since it issued the target certificate, and the OCSP responder's certificate was signed using the CA's new private key. In this case, however, the CA has been issued a new certificate from the root CA, while the certificate from the root CA certifying the old public key remains valid.

In the PKI in Figure 7, there is also a designated OCSP responder that provides revocation status information for certificates issued by the root CA. So, the root CA has issued a certificate to this OCSP responder that includes an extended key usage extension with the id-kp-OCSPSigning OID.

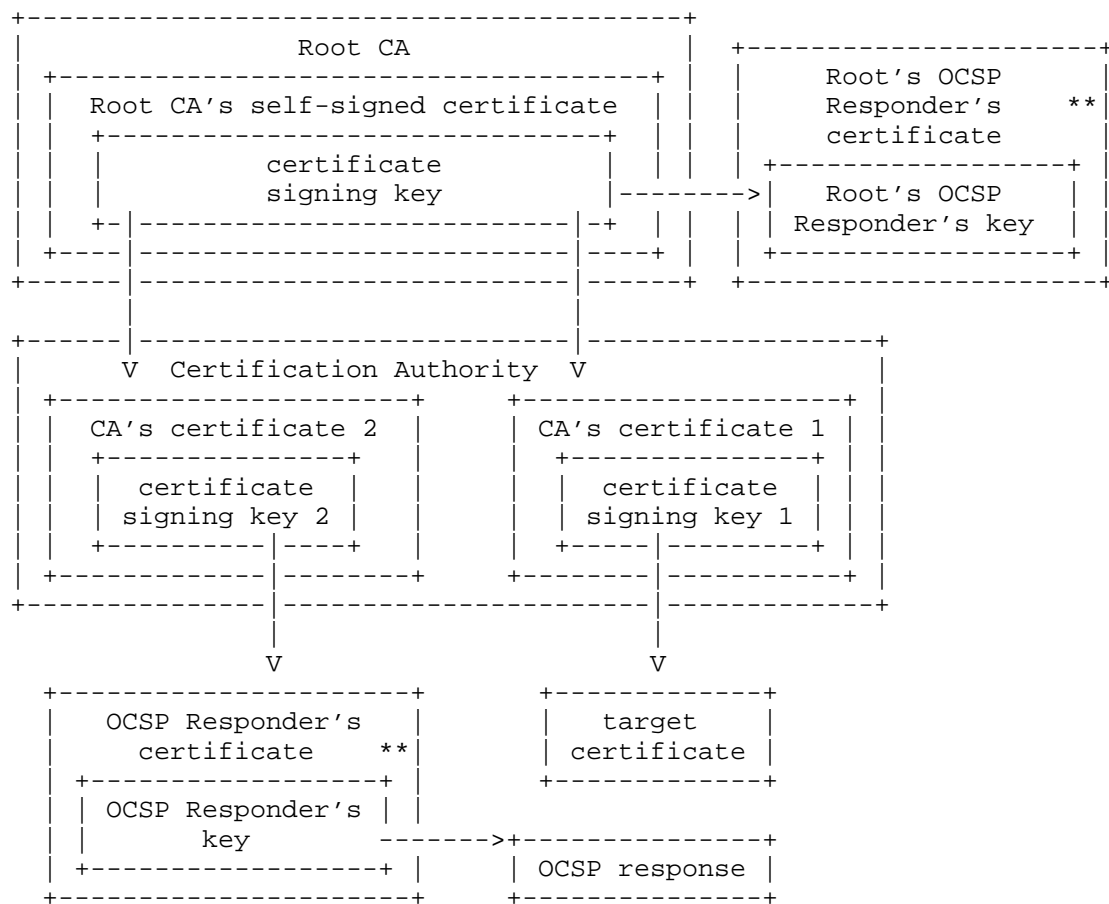


Figure 7. Designated OCSP Responder and CA With Two Keys Certified by Root CA

As noted in Section 2.3.5.1, some OCSF clients cannot validate signatures on OCSF responses created by designated OCSF responders if the OCSF responder's certificate has been signed using a different key than the target certificate. Figure 8 depicts a CA that accommodates this limitation by issuing two certificates to the designated OCSF responder, one signed with its old private key and one signed with its new private key. Both certificates include an extended key usage extension with the id-kp-OCSPSigning OID.

As in Figure 7, there is also a designated OCSF responder that provides revocation status information for certificates issued by the root CA, and so this responder has been issued a certificate by the root CA.

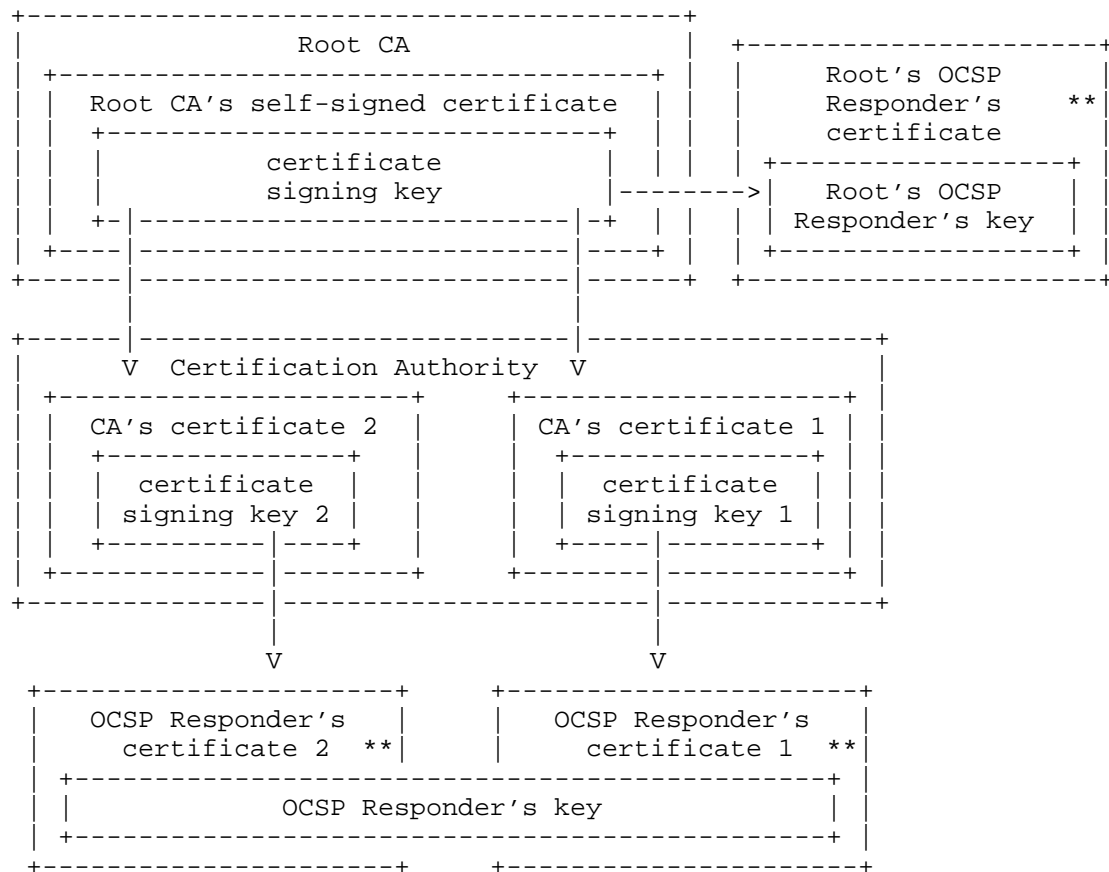


Figure 8. Designated OCSF Responder With Two Certificates

Figure 9 depicts a scenario in which an organization has a hierarchical PKI with three CAs, but has a single designated OCSP responder, which each CA has designated as being authorized to provide revocation status information for the certificates that it has issued. In order to satisfy the requirement that the authorizing CA issue a certificate directly to the designated OCSP responder, each of the CAs has issued a certificate to the OCSP responder that includes the `id-kp-ocspSigning` OID in the extended key usage extension. The subject public key in each of the three certificates issued to the OCSP responder is the same since the OCSP responder uses the same private key to sign all responses.

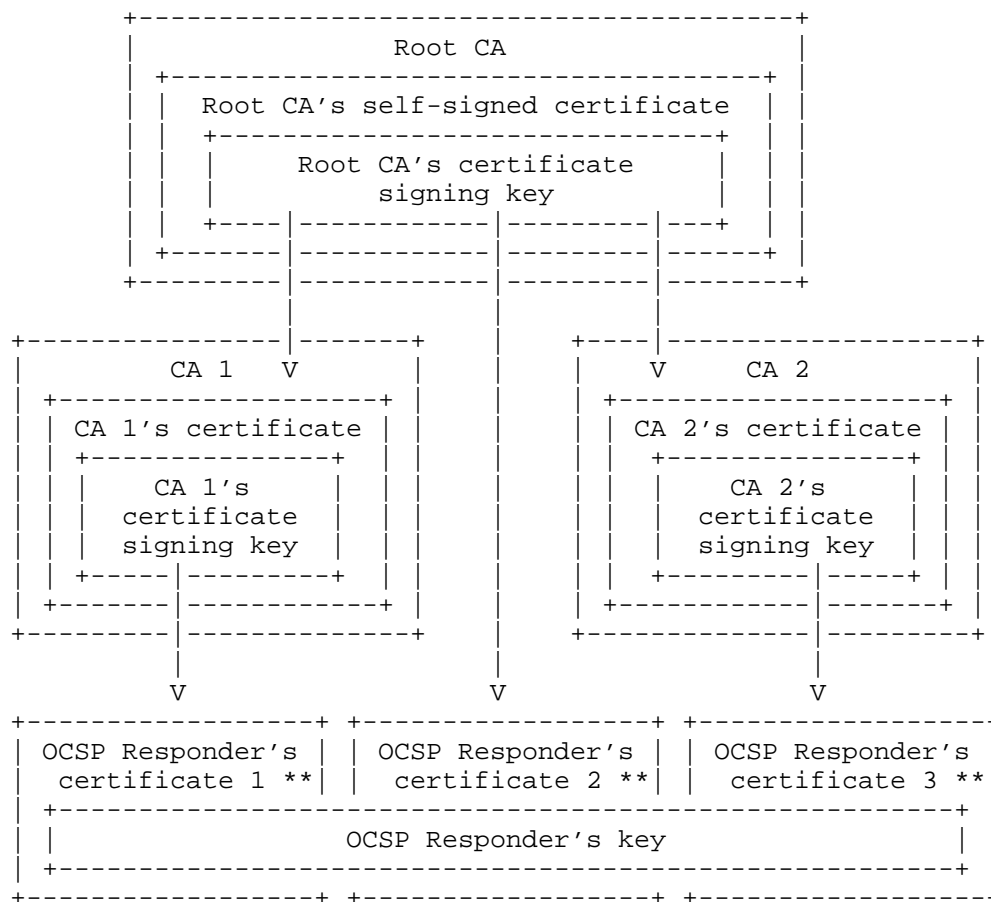


Figure 9. Hierarchical PKI with One Designated OCSP Responder

E.3. Locally Trusted OCSP Responder

Figure 10 depicts a PKI with a locally trusted OCSP responder. The PKI consists of six companies' CAs that are connected through a bridge CA. Company F operates an OCSP responder that it only makes accessible to computers owned by Company F. Company F's computers have been configured to trust the OCSP responder's public key to validate signatures on OCSP responses and have also been configured to use the OCSP responder to obtain revocation status information for all certificates.

Each of the CAs in the PKI issues CRLs, which they make available from publicly accessible directories. The OCSP responder retrieves these CRLs and uses the information in them to generate responses for its clients.

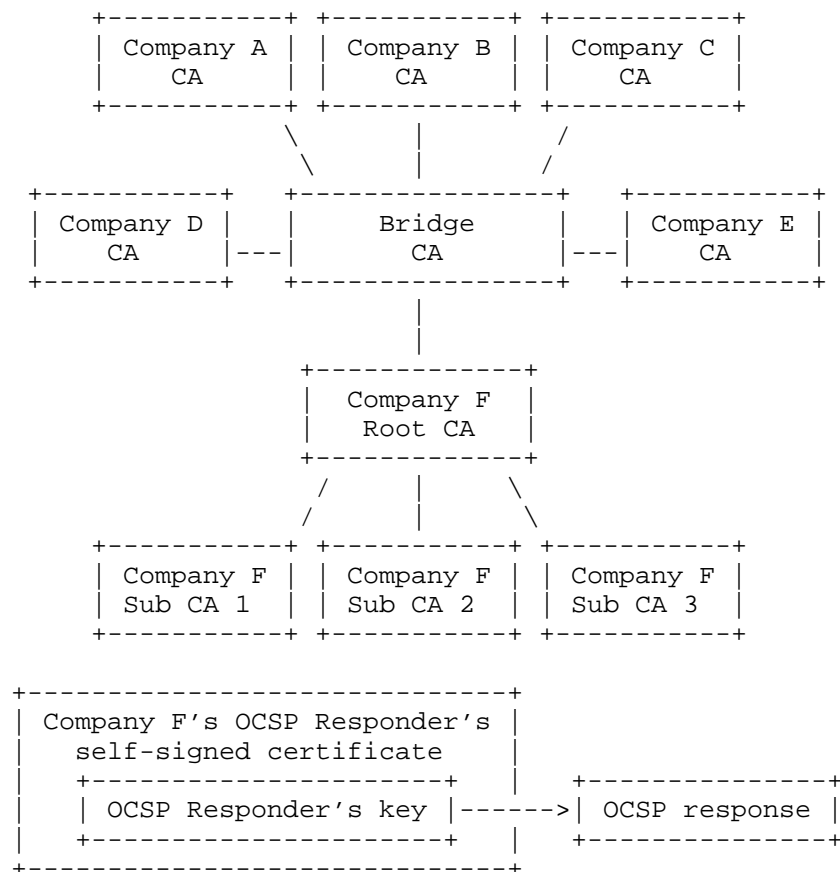


Figure 10. Locally Trusted OCSP Responder

Author's Address

David Cooper
National Institute of Standards and Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
USA
EMail: david.cooper@nist.gov

Stefan Santesson
3xA Security AB
Scheelev. 17
223 70 Lund
Sweden
EMail: sts@aaa-sec.com