Internet Engineering Task Force (IETF)
Request for Comments: 6960
Obsoletes: 2560, 6277
Updates: 5912
Category: Standards Track
ISSN: 2070-1721

S. Santesson
3xA Security
M. Myers
TraceRoute Security
R. Ankney

A. Malpani
CA Technologies
S. Galperin
A9
C. Adams
University of Ottawa
June 2013

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

Abstract

This document specifies a protocol useful in determining the current status of a digital certificate without requiring Certificate Revocation Lists (CRLs). Additional mechanisms addressing PKIX operational requirements are specified in separate documents. This document obsoletes RFCs 2560 and 6277. It also updates RFC 5912.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies a protocol useful in determining the current status of a digital certificate without requiring CRLs. Additional mechanisms addressing PKIX operational requirements are specified in separate documents.

This specification obsoletes [RFC2560] and [RFC6277]. 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:

- Section 2.2 extends the use of the "revoked" response to allow this response status for certificates that have never been issued.
- Section 2.3 extends the use of the "unauthorized" error response, as specified in [RFC5019].
- Sections 4.2.1 and 4.2.2.3 state that a response may include revocation status information for certificates that were not included in the request, as permitted in [RFC5019].
- Section 4.2.2.2 clarifies when a responder is considered an Authorized Responder.
- Section 4.2.2.3 clarifies that the ResponderID field corresponds to the OCSP responder signer certificate.
- Section 4.3 changes the set of cryptographic algorithms that clients must support and the set of cryptographic algorithms that clients should support as specified in [RFC6277].
- Section 4.4.1 specifies, for the nonce extension, ASN.1 syntax that was missing in RFC 2560.
- Section 4.4.7 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 as specified in [RFC6277].
- Section 4.4.8 specifies a new extension that indicates that the responder supports the extended use of the "revoked" response for non-issued certificates defined in Section 2.2.
- Appendix B.2 provides an ASN.1 module using the 2008 syntax of ASN.1, which updates [RFC5912].

An overview of the protocol is provided in Section 2. Functional requirements are specified in Section 3. Details of the protocol are discussed in Section 4. We cover security issues with the protocol in Section 5. Appendix A defines OCSP over HTTP, Appendix B provides ASN.1 syntactic elements, and Appendix C specifies the MIME types for the messages.

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. Protocol Overview

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 certificates (cf. [RFC5280], Section 3.3). Examples include high-value funds transfers or large stock trades.

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of identified certificates. 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 certificates 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 one or more certificates and the server providing the corresponding status.

2.1. Request

An OCSP request contains the following data:

- protocol version

- service request

- target certificate identifier

- optional extensions, which MAY be processed by the OCSP responder

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 these conditions is not met, the OCSP responder produces an error message; otherwise, it returns a definitive response.

2.2. Response

OCSP responses can be of various types. An OCSP response consists of a response type and the bytes of the actual response. There is one basic type of OCSP response that MUST be supported by all OCSP servers and clients. The rest of this section pertains only to this basic response type.

All definitive response messages SHALL be digitally signed. The key used to sign the response MUST belong to one of the following:

- the CA who issued the certificate in question

- a Trusted Responder whose public key is trusted by the requestor

- a CA Designated Responder (Authorized Responder, defined in Section 4.2.2.2) who holds a specially marked certificate issued directly by the CA, indicating that the responder may issue OCSP responses for that CA

A definitive response message is composed of:

- version of the response syntax

- identifier of the responder

- time when the response was generated

- responses for each of the certificates in a request

- optional extensions

- signature algorithm OID

- signature computed across a hash of the response

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

- target certificate identifier

- certificate status value

- response validity interval

- optional extensions

This specification defines the following definitive response indicators for use in the certificate status value:

- good

- revoked

- unknown

The "good" state indicates a positive response to the status inquiry. At a minimum, this positive response indicates that no certificate with the requested certificate serial number currently within its validity interval is revoked. This state does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval. 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, validity, etc.

The "revoked" state indicates that the certificate has been revoked, either temporarily (the revocation reason is certificateHold) or permanently. This state MAY also be returned if the associated CA has no record of ever having issued a certificate with the certificate serial number in the request, using any current or previous issuing key (referred to as a "non-issued" certificate in this document).

The "unknown" state indicates that the responder doesn't know about the certificate being requested, usually because the request indicates an unrecognized issuer that is not served by this responder.

NOTE: The "revoked" status indicates that a certificate with the requested serial number should be rejected, while the "unknown" status indicates that the status could not be determined by this responder, thereby allowing the client to decide whether it wants to try another source of status information (such as a

CRL). This makes the "revoked" response suitable for non-issued certificates (as defined above) where the intention of the responder is to cause the client to reject the certificate rather than trying another source of status information. The "revoked" status is still optional for non-issued certificates in order to maintain backwards compatibility with deployments of RFC 2560. For example, the responder may not have any knowledge about whether a requested serial number has been assigned to any issued certificate, or the responder may provide pre-produced responses in accordance with RFC 5019 and, for that reason, is not capable of providing a signed response for all non-issued certificate serial numbers.

When a responder sends a "revoked" response to a status request for a non-issued certificate, the responder MUST include the extended revoked definition response extension (Section 4.4.8) in the response, indicating that the OCSP responder supports the extended definition of the "revoked" state to also cover non-issued certificates. In addition, the SingleResponse related to this non-issued certificate:

- MUST specify the revocation reason certificateHold (6),

- MUST specify the revocationTime January 1, 1970, and

- MUST NOT include a CRL references extension (Section 4.4.2) or any CRL entry extensions (Section 4.4.5).

2.3. Exception Cases

In case of errors, the OCSP responder may return an error message. These messages are not signed. Errors can be of the following types:

- malformedRequest

- internalError

- tryLater

- sigRequired

- unauthorized

A server produces the "malformedRequest" response if the request received does not conform to the OCSP syntax.

The response "internalError" indicates that the OCSP responder reached an inconsistent internal state. The query should be retried, potentially with another responder.

In the event that the OCSP responder is operational but unable to return a status for the requested certificate, the "tryLater" response can be used to indicate that the service exists but is temporarily unable to respond.

The response "sigRequired" is returned in cases where the server requires that the client sign the request in order to construct a response.

The response "unauthorized" is returned in cases where 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).

2.4. Semantics of thisUpdate, nextUpdate, and producedAt

Responses defined in this document can contain four times -- thisUpdate, nextUpdate, producedAt, and revocationTime. The semantics of these fields are:

thisUpdate — The most recent time at which the status being indicated is known by the responder to have been correct.

nextUpdate — The time at or before which newer information will be available about the status of the certificate.

producedAt — The time at which the OCSP responder signed this response.

revocationTime — The time at which the certificate was revoked or placed on hold.

2.5. Response Pre-Production

OCSP 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.

2.6. OCSP Signature Authority Delegation

The key that signs a certificate's status information need not be the same key that signed the certificate. A certificate's issuer explicitly delegates OCSP signing authority by issuing a certificate containing a unique value for the extended key usage extension (defined in [RFC5280], Section 4.2.1.12) in the OCSP signer's certificate. This certificate MUST be issued directly to the responder by the cognizant CA. See Section 4.2.2.2 for details.

2.7. CA Key Compromise

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

3. Functional Requirements

3.1. Certificate Content

In order to convey to OCSP clients a well-known point of information access, CAs SHALL provide the capability to include the authority information access extension (defined in [RFC5280], Section 4.2.2.1) in certificates that can be checked using OCSP. Alternatively, the accessLocation for the OCSP provider may be configured locally at the OCSP client.

CAs that support an OCSP service, either hosted locally or provided by an Authorized Responder, MUST provide for the inclusion of a value for a Uniform Resource Identifier (URI) [RFC3986] accessLocation and the OID value id-ad-ocsp for the accessMethod in the AccessDescription SEQUENCE.

The value of the accessLocation field in the subject certificate defines the transport (e.g., HTTP) used to access the OCSP responder and may contain other transport-dependent information (e.g., a URL).

3.2. Signed Response Acceptance Requirements

Prior to accepting a signed response for a particular certificate as valid, OCSP clients SHALL confirm that:

1. The certificate identified in a received response corresponds to the certificate that was identified in the corresponding request;

2. The signature on the response is valid;

3. The identity of the signer matches the intended recipient of the request;

4. The signer is currently authorized to provide a response for the certificate in question;

5. The time at which the status being indicated is known to be correct (thisUpdate) is sufficiently recent;

6. When available, the time at or before which newer information will be available about the status of the certificate (nextUpdate) is greater than the current time.

4. Details of the Protocol

The ASN.1 syntax imports terms defined in [RFC5280]. For signature calculation, the data to be signed is encoded using the ASN.1 distinguished encoding rules (DER) [X.690].

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

The terms imported from elsewhere are Extensions, CertificateSerialNumber, SubjectPublicKeyInfo, Name, AlgorithmIdentifier, and CRLReason.

4.1. Request Syntax

This section specifies the ASN.1 specification for a confirmation request. The actual formatting of the message could vary, depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

4.1.1. ASN.1 Specification of the OCSP Request

The ASN.1 structure corresponding to the OCSPRequest is:

```
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 }
```

The fields in OCSPRequest have the following meanings:

o tbsRequest is the optionally signed OCSP request.

o optionalSignature contains the algorithm identifier and any associated algorithm parameters in signatureAlgorithm; the signature value in signature; and, optionally, certificates the server needs to verify the signed response (normally up to but not including the client's root certificate).

The contents of TBSRequest include the following fields:

o version indicates the version of the protocol, which for this document is v1(0).

o requestorName is OPTIONAL and indicates the name of the OCSP requestor.

o requestList contains one or more single certificate status requests.

o requestExtensions is OPTIONAL and includes extensions applicable to the requests found in reqCert. See Section 4.4.

The contents of Request include the following fields:

o reqCert contains the identifier of a target certificate.

o singleRequestExtensions is OPTIONAL and includes extensions applicable to this single certificate status request. See Section 4.4.

The contents of CertID include the following fields:

o hashAlgorithm is the hash algorithm used to generate the issuerNameHash and issuerKeyHash values.

o issuerNameHash is the hash of the issuer's distinguished name (DN). The hash shall be calculated over the DER encoding of the issuer's name field in the certificate being checked.

o issuerKeyHash is the hash of the issuer's public key. The hash shall be calculated over the value (excluding tag and length) of the subject public key field in the issuer's certificate.

o serialNumber is the serial number of the certificate for which status is being requested.

4.1.2. Notes on OCSP Requests

The primary reason to use the hash of the CA's public key in addition to the hash of the CA's name to identify the issuer is that it is possible that two CAs may choose to use the same Name (uniqueness in the Name is a recommendation that cannot be enforced). Two CAs will never, however, have the same public key unless the CAs either explicitly decided to share their private key or the key of one of the CAs was compromised.

Support for any specific extension is OPTIONAL. The critical flag SHOULD NOT be set for any of them. Section 4.4 suggests several useful extensions. Additional extensions MAY be defined in additional RFCs. Unrecognized extensions MUST be ignored (unless they have the critical flag set and are not understood).

The requestor MAY choose to sign the OCSP request. In that case, the signature is computed over the tbsRequest structure. If the request is signed, the requestor SHALL specify its name in the requestorName field. Also, for signed requests, the requestor MAY include certificates that help the OCSP responder verify the requestor's signature in the certs field of Signature.

4.2. Response Syntax

This section specifies the ASN.1 specification for a confirmation response. The actual formatting of the message could vary, depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

4.2.1. ASN.1 Specification of the OCSP Response

An OCSP response at a minimum consists of a responseStatus field indicating the processing status of the prior request. If the value of responseStatus is one of the error conditions, the responseBytes field is not set.

```
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
}
```

The value for responseBytes consists of an OBJECT IDENTIFIER and a response syntax identified by that OID encoded as an OCTET STRING.

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

For a basic OCSP responder, responseType will be id-pkix-ocsp-basic.

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

OCSP responders SHALL be capable of producing responses of the id-pkix-ocsp-basic response type. Correspondingly, OCSP clients SHALL be capable of receiving and processing responses of the id-pkix-ocsp-basic response type.

The value for response SHALL be 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 hash of the DER encoding of ResponseData. The responder MAY include certificates in the certs field of BasicOCSPResponse that help the OCSP 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
(excluding the tag and length fields)

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
```

4.2.2. Notes on OCSP Responses

4.2.2.1. Time

Responses can contain four times -- thisUpdate, nextUpdate, producedAt, and revocationTime. The semantics of these fields are defined in Section 2.4. The format for GeneralizedTime is as specified in Section 4.1.2.5.2 of [RFC5280].

The thisUpdate and nextUpdate fields define a recommended validity interval. This interval corresponds to the {thisUpdate, nextUpdate} interval in CRLs. Responses whose nextUpdate value is earlier than the local system time value SHOULD be considered unreliable. Responses whose thisUpdate time is later than the local system time SHOULD be considered unreliable.

If nextUpdate is not set, the responder is indicating that newer revocation information is available all the time.

4.2.2.2. Authorized Responders

The key that signs a certificate's status information need not be the same key that signed the certificate. It is necessary, however, to ensure that the entity signing this information is authorized to do so. Therefore, a certificate's issuer MUST do one of the following:

- sign the OCSP responses itself, or

- explicitly designate this authority to another entity

OCSP signing delegation SHALL be designated by the inclusion of id-kp-OCSPSigning in an extended key usage certificate extension included in the OCSP response signer's certificate. This certificate MUST be issued directly by the CA that is identified in the request.

The CA SHOULD use the same issuing key to issue a delegation certificate as that used to sign the certificate being checked for revocation. Systems relying on OCSP responses MUST recognize a delegation certificate as being issued by the CA that issued the certificate in question only if the delegation certificate and the certificate being checked for revocation were signed by the same key.

Note: For backwards compatibility with RFC 2560 [RFC2560], it is not prohibited to issue a certificate for an Authorized Responder using a different issuing key than the key used to issue the certificate being checked for revocation. However, such a practice is strongly discouraged, since clients are not required to recognize a responder with such a certificate as an Authorized Responder.

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

Systems or applications that rely on OCSP responses MUST be capable of detecting and enforcing the use of the id-kp-OCSPSigning value as described above. 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. They MUST reject the response if the certificate required to validate the signature on the response does not meet at least one of the following criteria:

1. Matches a local configuration of OCSP signing authority for the certificate in question, or

2. Is the certificate of the CA that issued the certificate in question, or

3. Includes a value of id-kp-OCSPSigning in an extended key usage extension and is issued by the CA that issued the certificate in question as stated above.

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

4.2.2.2.1. Revocation Checking of an Authorized Responder

Since an authorized OCSP responder provides status information for one or more CAs, OCSP 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:

- A CA may specify that an OCSP 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. CAs may choose to issue this type of certificate with a very short lifetime and renew it frequently.

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

- A CA may specify how the responder's certificate is to be checked for revocation. This can be done by using CRL Distribution Points if the check should be done using CRLs, or by using 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].

- 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 or not.

4.2.2.3. Basic Response

The basic response type contains:

- the version of the response syntax, which MUST be v1 (value is 0) for this version of the basic response syntax;

- either the name of the responder or a hash of the responder's public key as the ResponderID;

- the time at which the response was generated;

- responses for each of the certificates in a request;

- optional extensions;

- a signature computed across a hash of the response; and

- the signature algorithm OID.

The purpose of the ResponderID information is to allow clients to find the certificate used to sign a signed OCSP response. Therefore, the information MUST correspond to the certificate that was used to sign the response.

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); if revoked, it indicates the time at which the certificate was revoked and, optionally, the reason why it was revoked;

- o the validity interval of the response; and

- o optional extensions.

The response MUST include a SingleResponse for each certificate in the request. The response SHOULD NOT include any additional SingleResponse elements, but, for example, OCSP responders that pre-generate status responses might include additional SingleResponse elements if necessary to improve response pre-generation performance or cache efficiency (according to [RFC5019], Section 2.2.1).

4.3. Mandatory and Optional Cryptographic Algorithms

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

4.4. Extensions

This section defines some standard extensions, based on the extension model employed in X.509 version 3 certificates (see [RFC5280]). Support for all extensions is optional for both clients and responders. For each extension, the definition indicates its syntax, processing performed by the OCSP responder, and any extensions that are included in the corresponding response.

4.4.1. 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 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
```

4.4.2. CRL References

It may be desirable for the OCSP responder to indicate the CRL on which a revoked or onHold certificate is found. This can be useful where OCSP 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). These extensions will be specified as singleExtensions. 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 the choice 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.

4.4.3. Acceptable Response Types

An OCSP 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. This extension is included as one of the requestExtensions in requests. 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 4.2.1, OCSP responders SHALL be capable of responding with responses of the id-pkix-ocsp-basic response type. Correspondingly, OCSP clients SHALL be capable of receiving and processing responses of the id-pkix-ocsp-basic response type.

4.4.4. Archive Cutoff

An OCSP 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.

OCSP-enabled applications would use an OCSP 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.

OCSP servers that provide support for such a historical reference SHOULD include an archive cutoff date extension in responses. If included, this value SHALL be provided as an OCSP singleExtensions extension identified by id-pkix-ocsp-archive-cutoff and of syntax GeneralizedTime.

```
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).

4.4.5. CRL Entry Extensions

All the extensions specified as CRL entry extensions -- in Section 5.3 of [RFC5280] -- are also supported as singleExtensions.

4.4.6. Service Locator

An OCSP server may be operated in a mode whereby the server receives a request and routes it to the OCSP server that is known to be authoritative for the identified certificate. The serviceLocator request extension is defined for this purpose. This extension is included as one of the singleRequestExtensions in requests.

```
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.

4.4.7. Preferred Signature Algorithms

Since algorithms other than the mandatory-to-implement algorithms are allowed, and since a client currently has no mechanism to indicate its 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:

- The algorithm used to sign the CRLs and certificates may not be consistent with the key pair being used by the OCSP responder to sign responses.

- 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 two 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 section describes:

- o An extension that allows a client to indicate the set of preferred signature algorithms.

- o Rules for signature algorithm selection that maximize the probability of successful operation in the case that no supported preferred algorithm(s) are specified.

4.4.7.1. Extension Syntax

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

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

PreferredSignatureAlgorithms ::= SEQUENCE OF
                                 PreferredSignatureAlgorithm

PreferredSignatureAlgorithm ::= SEQUENCE {
   sigIdentifier        AlgorithmIdentifier,
   pubKeyAlgIdentifier  SMIMECapability OPTIONAL
   }
```

The syntax of AlgorithmIdentifier is defined in Section 4.1.1.2 of RFC 5280 [RFC5280]. The syntax of SMIMECapability is defined in RFC 5751 [RFC5751].

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

pubKeyAlgIdentifier 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.

pubKeyAlgIdentifier 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, from the most preferred to the least preferred.

Section 4.4.7.2 of this document describes how a server selects an algorithm for signing OCSP responses to the requesting client.

4.4.7.2. Responder Signature Algorithm Selection

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

4.4.7.2.1. 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 selection mechanism 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 certificate revocation list (CRL) issued by the certificate issuer providing status information for the certificate specified by CertID.

3. Select the signature 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 signature algorithm specified for the version of OCSP in use.

A responder SHOULD always apply the lowest-numbered selection mechanism that results in the selection of a known and supported algorithm that meets the responder's criteria for cryptographic algorithm strength.

4.4.7.2.2. 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.

4.4.8. Extended Revoked Definition

This extension indicates that the responder supports the extended definition of the "revoked" status to also include non-issued certificates according to Section 2.2. One of its main purposes is to allow audits to determine the responder's type of operation. Clients do not have to parse this extension in order to determine the status of certificates in responses.

This extension MUST be included in the OCSP response when that response contains a "revoked" status for a non-issued certificate. This extension MAY be present in other responses to signal that the responder implements the extended revoked definition. When included, this extension MUST be placed in responseExtensions, and it MUST NOT appear in singleExtensions.

This extension is identified by the object identifier id-pkix-ocsp-extended-revoke.

```
id-pkix-ocsp-extended-revoke OBJECT IDENTIFIER ::= {id-pkix-ocsp 9}
```

The value of the extension SHALL be NULL. This extension MUST NOT be marked critical.

5. 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 vulnerability to denial of service 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 precomputed 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 precomputed responses against the probability of a replay attack and the costs associated with its successful execution.

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

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. Implementors are advised to take the reliability of HTTP cache mechanisms into account when deploying OCSP over HTTP.

Responding with a "revoked" state to a certificate that has never been issued may enable someone to obtain a revocation response for a certificate that is not yet issued, but soon will be issued, if the certificate serial number of the certificate that will be issued can be predicted or guessed by the requestor. Such a prediction is easy for a CA that issues certificates using sequential certificate serial number assignment. This risk is handled in the specification by requiring compliant implementations to use the certificateHold reason code, which avoids permanently revoking the serial number. For CAs that support "revoked" responses to status requests for non-issued certificates, one way to completely avoid this issue is to assign random certificate serial number values with high entropy.

5.1. Preferred Signature Algorithms

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 criterion 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.

5.1.1. 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 using a signing mechanism that is not considered acceptably secure.

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.

5.1.2. 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.

5.1.3. 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. Denial-of-service considerations as discussed in RFC 4732 [RFC4732] are relevant for this document.

6. IANA Considerations

This document includes media type registrations (in Appendix C) for ocsp-request and ocsp-response that were registered when RFC 2560 was published. Because this document obsoletes RFC 2560, IANA has updated the references in the "Application Media Types" registry for ocsp-request and ocsp-response to point to this document.

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.

[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.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[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.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", RFC 6277, June 2011.

[X.690] ITU-T Recommendation X.690 (2008) | ISO/IEC 8825-1:2008, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", November 2008.

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.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.

[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 (PKIX)", RFC 5912, June 2010.

8. Acknowledgements

Development of this document has been made possible thanks to extensive inputs from members of the PKIX working group.

Jim Schaad provided valuable support by compiling and checking the ASN.1 modules of this specification.

Appendix A. OCSP over HTTP

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

A.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 if 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 Transport Layer Security/Secure Socket Layer (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 the authority information access extension in the certificate being checked for revocation, 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.

A.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 B. ASN.1 Modules

This appendix includes the ASN.1 modules for OCSP. Appendix B.1 includes an ASN.1 module that conforms to the 1998 version of ASN.1 for all syntax elements of OCSP, including the preferred signature algorithms extension that was defined in [RFC6277]. This module replaces the modules in Appendix B of [RFC2560] and Appendix A.2 of [RFC6277]. Appendix B.2 includes an ASN.1 module, corresponding to the module present in B.1, that conforms to the 2008 version of

ASN.1. This module replaces the modules in Section 12 of [RFC5912] and Appendix A.1 of [RFC6277]. Although a 2008 ASN.1 module is provided, the module in Appendix B.1 remains the normative module as per the policy of the PKIX working group.

B.1. OCSP in ASN.1 - 1998 Syntax

```
OCSP-2013-88
      {iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-ocsp-2013-88(81)}

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 issuer's 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 }

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

PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm

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

```
-- 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 }
id-pkix-ocsp-pref-sig-algs   OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }
id-pkix-ocsp-extended-revoke OBJECT IDENTIFIER ::= { id-pkix-ocsp 9 }

END
```

B.2. OCSP in ASN.1 - 2008 Syntax

```
OCSP-2013-08
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp-2013-08(82)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

Extensions{}, EXTENSION, ATTRIBUTE
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)}

AlgorithmIdentifier{}, DIGEST-ALGORITHM, 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)}

AuthorityInfoAccessSyntax, GeneralName, CrlEntryExtensions
FROM PKIX1Implicit-2009 -- From [RFC5912]
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}

Name, CertificateSerialNumber, id-kp, id-ad-ocsp, Certificate
FROM PKIX1Explicit-2009 -- From [RFC5912]
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}
```

```
sa-dsaWithSHA1, sa-rsaWithMD2, sa-rsaWithMD5, sa-rsaWithSHA1
FROM PKIXAlgs-2009 -- From [RFC5912]
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-pkix1-algorithms2008-02(56)};

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 {{re-ocsp-nonce |
                     re-ocsp-response, ...,
                     re-ocsp-preferred-signature-algorithms}} OPTIONAL }

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

Version  ::=  INTEGER  {  v1(0) }

Request ::=     SEQUENCE {
    reqCert                    CertID,
    singleRequestExtensions    [0] EXPLICIT Extensions
                                       { {re-ocsp-service-locator,
                                              ...}} OPTIONAL }

CertID ::= SEQUENCE {
    hashAlgorithm            AlgorithmIdentifier
                                 {DIGEST-ALGORITHM, {...}},
    issuerNameHash     OCTET STRING, -- Hash of issuer's DN
    issuerKeyHash      OCTET STRING, -- Hash of issuer's 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
}

RESPONSE ::= TYPE-IDENTIFIER

ResponseSet RESPONSE ::= {basicResponse, ...}

ResponseBytes ::=       SEQUENCE {
    responseType        RESPONSE.
                            &id ({ResponseSet}),
    response            OCTET STRING (CONTAINING RESPONSE.
                            &Type({ResponseSet}{@responseType}))}

basicResponse RESPONSE ::=
    { BasicOCSPResponse IDENTIFIED BY id-pkix-ocsp-basic }

BasicOCSPResponse       ::= SEQUENCE {
   tbsResponseData      ResponseData,
   signatureAlgorithm   AlgorithmIdentifier{SIGNATURE-ALGORITHM,
                            {sa-dsaWithSHA1 | sa-rsaWithSHA1 |
                                 sa-rsaWithMD5 | sa-rsaWithMD2, ...}},
   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
                               {{re-ocsp-nonce, ...,
                                 re-ocsp-extended-revoke}} OPTIONAL }

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

KeyHash ::= OCTET STRING -- SHA-1 hash of responder's public key
                         -- (excluding the tag and length fields)
```

```
SingleResponse ::= SEQUENCE {
   certID                       CertID,
   certStatus                   CertStatus,
   thisUpdate                   GeneralizedTime,
   nextUpdate           [0]     EXPLICIT GeneralizedTime OPTIONAL,
   singleExtensions     [1]     EXPLICIT Extensions{{re-ocsp-crl |
                                             re-ocsp-archive-cutoff |
                                             CrlEntryExtensions, ...}
                                             } 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 RESPONSE.&id({ResponseSet})

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

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

PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm

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

-- Certificate Extensions

ext-ocsp-nocheck EXTENSION ::= { SYNTAX NULL IDENTIFIED
                                 BY id-pkix-ocsp-nocheck }
```

```
-- Request Extensions

re-ocsp-nonce EXTENSION ::= { SYNTAX OCTET STRING IDENTIFIED
                              BY id-pkix-ocsp-nonce }

re-ocsp-response EXTENSION ::= { SYNTAX AcceptableResponses IDENTIFIED
                                 BY id-pkix-ocsp-response }

re-ocsp-service-locator EXTENSION ::= { SYNTAX ServiceLocator
                                        IDENTIFIED BY
                                        id-pkix-ocsp-service-locator }

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

-- Response Extensions

re-ocsp-crl EXTENSION ::= { SYNTAX CrlID IDENTIFIED BY
                                id-pkix-ocsp-crl }

re-ocsp-archive-cutoff EXTENSION ::= { SYNTAX ArchiveCutoff
                                       IDENTIFIED BY
                                       id-pkix-ocsp-archive-cutoff }

re-ocsp-extended-revoke EXTENSION ::= { SYNTAX NULL IDENTIFIED BY
                                        id-pkix-ocsp-extended-revoke }

-- 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 }
id-pkix-ocsp-pref-sig-algs   OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }
id-pkix-ocsp-extended-revoke OBJECT IDENTIFIER ::= { id-pkix-ocsp 9 }

END
```

Appendix C. MIME Registrations

C.1. application/ocsp-request

To: ietf-types@iana.org
Subject: Registration of MIME media type application/ocsp-request

MIME media type name: application

MIME 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: IETF PKIX Working Group document on the Online Certificate Status Protocol - OCSP

Applications which 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:
Stefan Santesson <sts@aaa-sec.com>

Intended usage: COMMON

Author/Change controller: IETF

C.2. application/ocsp-response

To: ietf-types@iana.org
Subject: Registration of MIME media type application/ocsp-response

MIME media type name: application

MIME 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: IETF PKIX Working Group document on the
Online Certificate Status Protocol - OCSP

Applications which use this media type: OCSP 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:
Stefan Santesson <sts@aaa-sec.com>

Intended usage: COMMON

Author/Change controller: IETF

Authors' Addresses

Stefan Santesson
3xA Security AB
Scheelev. 17
223 70 Lund
Sweden

EMail: sts@aaa-sec.com

Michael Myers
TraceRoute Security

EMail: mmyers@fastq.com

Rich Ankney

Ambarish Malpani
CA Technologies
455 West Maude Ave. Suite 210
Sunnyvale, CA 94085
United States

EMail: ambarish@gmail.com

Slava Galperin
A9.com Inc.
130 Lytton Ave. Suite 300
Palo Alto, CA 94301
United States

EMail: slava.galperin@gmail.com

Carlisle Adams
University of Ottawa
800 King Edward Avenue
Ottawa ON K1N 6N5
Canada

EMail: cadams@eecs.uottawa.ca