Lightweight OCSP Profile
for High Volume Environments

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, or will be disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

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 a "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

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

Abstract

This specification defines a profile of the Online Certificate Status Protocol (OCSP) that addresses the scalability issues inherent when using OCSP in large scale (high volume) PKI environments and/or PKI environments that require a lightweight solution to minimize bandwidth and client side processing.

Table of Contents

Introduction......................................................2
1. OCSP Message Profile........................................3
   1.1 OCSP Request Profile..................................3
       1.1.1 OCSPRequest Structure..........................3
       1.1.2 Signed OCSPRequests............................4
   1.2 OCSP Response Profile.................................4
       1.2.1 OCSPResponse Structure........................4
       1.2.2 Signed OCSPResponses...........................5
Introduction

The Online Certificate Status Protocol [OCSP] specifies a mechanism used to determine the status of digital certificates, without requiring CRL’s. Since its definition in 1999, it has been deployed in a variety of environments and has proven to be a useful certificate status checking mechanism.

To date, many OCSP deployments have been used to ensure timely and secure certificate status information for high-value electronic transactions or highly sensitive information, such as in the banking and financial environments. As such, the requirement for an OCSP responder to respond in "real time" (i.e. generating a new OCSP response for each OCSP request) has been important. In addition, these deployments have operated in environments where bandwidth usage is not an issue, and have run on client and server systems where processing power is not constrained.

As the use of PKI continues to grow and move into diverse environments, so does the need for a scalable and cost effective...
certificate status mechanism. Although OCSP as currently defined and deployed meets the need of small to medium sized PKI’s which operate on powerful systems on wired networks, there is a limit as to how these OCSP deployments scale from both a performance and cost perspective. Mobile environments, where network bandwidth is at a premium and client side devices are constrained from a processing point of view, require the careful use of OCSP to minimize bandwidth usage and client side processing complexity.

Similarly, as PKI continues to be deployed into environments where millions if not hundreds of millions of certificates have been issued and an equal number of users (also known as relying parties) have the need to ensure that the certificate they are relying upon have not been revoked, it is important that OCSP is used in such a way that ensures the load on OCSP responders and the network infrastructure required to host those responders is kept to a minimum.

This document addresses the scalability issues inherent when using OCSP in PKI environments described above by defining an message profile and OCSP client and responder behavior that will permit:

1) OCSP response pre-production and distribution
2) Reduced OCSP message size to lower bandwidth usage
3) Response message caching both in the network and on the client

It is intended that the normative requirements defined in this profile apply to OCSP clients and OCSP responders operating in very large scale (high volume) PKI environments or PKI environments that require a lightweight solution to minimize bandwidth and client side processing power (or both), as described above.

1. OCSP Message Profile

This section defines a subset of OCSPRequest and OCSPResponse functionality as defined in [OCSP].

1.1 OCSP Request Profile

1.1.1 OCSPRequest Structure

OCSPRequests conformant to this profile MUST include only one Request in the OCSPRequest.RequestList structure.

Clients MUST use SHA1 as the hashing algorithm for the CertID.issuerNameHash and the CertID.issuerKeyHash values.

Clients MUST NOT include the singleRequestExtensions structure.
Clients SHOULD NOT include the requestExtensions structure. If a requestExtensions structure is included, this profile RECOMMENDS that it contain only the nonce extension (id-pkix-ocsp-nonce). See Section 3 for issues concerning the use of a nonce in high volume OCSP environments.

1.1.2 Signed OCSPRequests

Clients SHOULD NOT create or send signed OCSPRequests. Responders MAY ignore the signature on OCSPRequests.

If the OCSPRequest is signed, the client SHALL specify its name in the OCSPRequest.requestorName field, otherwise clients SHOULD NOT include the requestorName field in the OCSPRequest. OCSP servers MUST be prepared to receive unsigned OCSP requests that contains the requestorName field, but must realize that the provided value is not authenticated.

Note: The suggested use of unsigned requests in this environment does not enable a responder to determine the authenticity of incoming request. Thus, access to the responder is implicitly given to any relying party.

1.2 OCSP Response Profile

1.2.1 OCSPResponse Structure

Responders MUST generate a BasicOCSPResponse as identified by the id-pkix-ocsp-basic OID. Clients MUST be able to parse and accept a BasicOCSPResponse. OCSPResponses conformant to this profile SHOULD include only one SingleResponse in the ResponseData.responses structure, but MAY include additional SingleResponse elements if necessary to improve response pre-generation performance or cache efficiency.

The responder SHOULD NOT include responseExtensions. Clients MUST NOT fail if they encounter non-critical responseExtensions in the response.

In the case a responder does not have the ability to respond to an OCSP request containing a nonce, such as if it only has the ability to use pre-produced responses, it SHOULD return a response that does not include a nonce. Clients SHOULD attempt to accept a response even if the response does not include a nonce. See Section 3 for details on validating responses that do not contain a nonce. See also Section 6 for relevant security considerations. Responders that do not have the ability to respond to OCSP requests that contain a nonce MAY forward the request to an OCSP responder capable of doing so.
The responder MAY include the singleResponse extensions structure.

1.2.2 Signed OCSPResponses

Responders MUST use the sha1WithRSAEncryption signature algorithm when signing the OCSPResponse. Clients MUST validate the signature on the returned OCSPResponse.

<<Editors Note: Need to determine best way to future-proof this draft with the new sha*WithRSAEncryption signature algorithms. Options are to either depend on the next version of OCSP and update when available or simply define that clients and servers MAY use the more modern algs.>>

If the response is signed by a delegate of the issuing CA the responder certificate MUST be referenced in the BasicOCSPResponse.certs structure.

The responder’s certificate SHOULD have a validity window greater than or equal to the validity window of the responses it issues.

It is RECOMMENDED that the OCSP responder’s certificate contain the id-ocsp-nocheck EKU OID to indicate to the client that it need not check its status. In addition, it is RECOMMENDED that neither an OCSP authorityInfoAccess (AIA) extension nor CDP extension be included in the the OCSP responder’s certificate. Accordingly, the responder’s signing certificate SHOULD be relatively short-lived and rolled over regularly.

Clients MUST be able to identify OCSP responder certificates using both the byName and byKey ResponseData.ResponderID choices. Responders MAY use byKey to further reduce the size of the response in scenarios where reducing bandwidth is an issue.

1.2.3 OCSPResponseStatus Values

As long as the responder has records for a particular certificate, an OCSPResponseStatus of "successful" will be returned. In order to ensure the database of revocation information does not grow unbounded over time, the responder MAY remove the status records of expired certificates.

OCSP responders that pre-produce and distribute OCSP responses in advance do not have the ability to properly respond with a signed "successful" yet "unknown" response as it is impossible to pre-produce and sign a response for the set of all possible "unknown" CertID’s in advance. Because of this, the responder will return an OCSPResponseStatus of "unauthorized" when processing
requests for which it is not capable of responding authoritatively. This includes the scenario where a responder has removed the records of expired certificates from its database.

1.2.4 thisUpdate, nextUpdate and producedAt

When pre-producing OCSPResponse messages, the responder MUST set the thisUpdate, nextUpdate and producedAt times as follows:

thisUpdate    The time at which the status being indicated is known to be correct.
nextUpdate    The time at or before which newer information will be available about the status of the certificate. Responders MUST always include this value to aid in response caching. See Section 5 for additional information on caching.
producedAt    The time at which the OCSP response is signed.

Note: In many cases the value of thisUpdate and producedAt will be the same.

For the purposes of this profile, GeneralizedTime values such as thisUpdate, nextUpdate and producedAt MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

2. Client Behavior

2.1 OCSP Responder Discovery

Clients MUST support the authorityInfoAccess extension as defined in [PKIX] and MUST recognize the id-ad-ocsp access method. This enables CAs to inform clients how they can contact the OCSP service.

In the case where a client is checking the status of a certificate that contains both an authorityInformationAccess (AIA) extension pointing to a OCSP responder and a cRLDistributionPoints extension pointing to a CRL, the client MUST contact the OCSP responder first. Clients MAY attempt to retrieve the CRL if and only if no OCSPResponse is received from the responder.

2.2 Sending an OCSP Request

To avoid needless network traffic, applications MUST verify the signature of signed data before asking an OCSP client to check the status of certificates used to verify the data. If the signature is
invalid or the application is not able to verify it, an OCSP check MUST NOT be requested.

Similarly, applications MUST validate the signature on certificates and its chain, before asking an OCSP client to check the status of the certificate. If the certificate signature is invalid or the application is not able to verify it, an OCSP check MUST NOT be requested. Clients SHOULD NOT request the status of expired certificates.

2.3 OCSP response status processing

OCSP response status ‘good’: The client MUST inform the calling application that the certificate has not been revoked. The client SHOULD NOT accept an OCSP response that indicates (in the nextUpdate field) that a newer response is available. See Section 5.1 for details on client caching behavior.

OCSP response status ‘revoked’: The client MUST inform the calling application that the signature is untrusted and abort any further processing of the signed data.

OCSP response status ‘unknown’: The client MUST inform the calling application about the unknown certificate status. This profile RECOMMENDS calling applications to warn the user about the unknown certificate status and give the user the option to continue or abort the processing of the data, with a default option of abort.

3. Ensuring an OCSPResponse is Fresh

In order to ensure a client does not accept an out of date response that indicates a ‘good’ status when in fact there is a more up to date response that specifies the status of ‘revoked’, a client must ensure the responses they receive are fresh.

In general, two mechanisms are available to clients to ensure a response is fresh. The first uses nonces, and the second is based on time. In order for time based mechanisms to work, both clients and responders MUST have access to an accurate source of time.

Because this profile specifies that clients SHOULD NOT include a requestExtensions structure in OCSPRequests (See Section 1.1) clients MUST be able to determine OCSPResponse freshness based on an accurate source of time. Clients that opt to include a nonce in the request MUST NOT reject a corresponding OCSPResponse solely on the basis of the non-existent expected nonce, but MUST fall back to validating the OCSPResponse based on time.
If a client includes a nonce in an OCSPRequest, and receives a nonce in the corresponding OCSPResponse it MUST ensure that the nonce included in the OCSPRequest matches the nonce received in the OCSPRequest. If the nonces do not match the client MUST reject the response as invalid. Clients that do not include a nonce in the request MUST ignore any nonce that may be present in the response.

If there is no nonce in the OCSPResponse, clients MUST check for the existence of the nextUpdate field and MUST ensure the current local time falls between the thisUpdate and nextUpdate times. If the nextUpdate field is absent, and there's no other way for the client to determine the freshness of the response, the client MUST reject the response.

If the nextUpdate field is present the client MUST ensure that it is not earlier than current time. If the current local time on the client is later than the time specified in the nextUpdate field, the client MUST reject the response as stale. Clients MAY allow configuration of a small tolerance period for acceptance of responses after nextUpdate to handle minor clock differences relative to responders and caches. This tolerance period should be no more than a few minutes to avoid introducing increased security risks.

See the security considerations in Section 6 for additional details on replay and man-in-the-middle attacks.

4. Transport Profile

The OCSP responder MUST support requests and responses over HTTP. When sending requests that are less than 255 bytes in total (after encoding) including the method (http://), server name and base64 encoded OCSPRequest structure, clients MUST use the GET method (to enable for OCSP response caching). OCSP requests larger than 255 bytes SHOULD be submitted using the POST method. In all cases, clients MUST follow the descriptions in A.1.1 of [OCSP] when constructing these messages.

When constructing a GET message, OCSP clients MUST base64 encode the OCSPRequest structure and append it to the URI specified in the AIA extension [PKIX]. Clients MUST NOT include CR or LF characters in the base64-encoded string. Clients MUST properly url-encode the base64 encoded OCSPRequest, e.g.

```
http://ocsp.example.com/MEowSDBGMEQwQjAKBggqhkiG9w0CBQQQ7sp6GTKpL2dAdeGaW267owQQInESWQD0mGeBAR8gv%2FBgWQIQLjx%2Fx9x8sYzo180Mbpg%3D%3D
```
In response to properly formatted OCSPRequests that are cachable (i.e. responses that contain a nextUpdate value), the responder will include the binary value of the DER encoding of the OCSPResponse preceded by the following HTTP headers.

```
content-type=application/ocsp-response
content-transfer-encoding=binary
content-length=<OCSP response length>
last-modified: <producedAt HTTP date>
expires: <nextUpdate HTTP date>
cache-control: max-age=<n>, public, no-transform, must-revalidate
date: <current HTTP date>
```

See Section 5.2 for details on the use of these headers.

5. Caching Recommendations

The ability to cache OCSP Responses throughout the network is an important factor in high volume OCSP deployments. This section discusses the recommended caching behavior of OCSP clients and HTTP proxies and the steps that should be taken to minimize the number of times that OCSP clients "hit the wire". In addition the concept of including OCSP responses in protocols exchanges (aka stapling or piggybacking), such as has been defined in TLS, is also discussed.

5.1 Caching at the Client

To minimize bandwidth usage, clients MUST locally cache authoritative OCSP responses. (i.e. those who’s signature has successfully validated and that indicate an OCSPResponseStatus of ‘successful’) Once cached, the client SHOULD NOT send a new OCSP request until the nextUpdate time in the cached response.

Most OCSP clients will send OCSPrequests at or near the nextUpdate time (when the cached response expires). To avoid large spikes in responder load that might occur when many clients refresh cached responses for a popular certificate (e.g. www.bigecommercesite.com), clients MAY use a locally derived value (i.e. 48 hours) to specify a time period before nextUpdate time when a client might consider refreshing its cached response. For example, a client can select a random time between "nextUpdate minus locally configured value" and "nextUpdate" at which to request a refresh of its cached OCSPResponse.

5.2 HTTP Proxies

The responder SHOULD set the HTTP headers of the OCSP response in such a way to allow for the intelligent use of intermediate HTTP proxy servers.
HTTP Header     Description
=============    ====================================================
date            The date and time at which the OCSP server generated
                the HTTP response.
last-modified   This value specifies the date and time at which the
                OCSP responder last modified the response. This
                date and time will be the same as the thisUpdate
                timestamp in the request itself.
expires         Specifies how long the response is considered fresh.
                This date and time will be the same as the
                nextUpdate timestamp in the OCSP response itself.
cache-control   Contains a number of caching directives.

* max-age=<n>- where n is the nextUpdate time
  minus the time the response is
generated (i.e. the Date header)
in seconds.
* public-      makes normally uncachable response
                cachable by both shared and
                nonshared caches.
* no-transform-specified that a proxy cache cannot
                change the type, length , or
                encoding of the object content.
* must-revalidate-prevents caches from
                intentionally returning stale
                responses.

OCSP responders MUST NOT include a Pragma: no-cache or
Cache-Control: no-store header in all OCSP responses.

For example, assume that an OCSP response has the following time
stamp values:

  thisUpdate = May 1, 2003  01:00:00 GMT
  nextUpdate = May 3, 2003 01:00:00 GMT
  producedAt = May 1, 2003 01:00:00 GMT

and that an OCSP client requests the response on May 2, 2003
01:00:00 GMT. In this scenario the HTTP response will look like
this:

  content-type: application/ocsp-response
  content-transfer-encoding: binary
  content-length: <OCSP response length>
  date: Fri, 02 May 2003 01:00:00 GMT
If a client encounters an expired response, it may be a result of an intermediate proxy caching stale data, as such clients SHOULD resend the request specifying that proxies should be bypassed by including an appropriate HTTP header in the request (i.e. Pragma: no-cache or Cache-Control: no-store).

5.3 Caching at Servers

In some scenarios it is advantageous to include OCSP response information within the protocol being utilized between the client and server. Including OCSP responses in this manner has a few attractive effects.

First, it allows for the caching of OCSP responses on the server, thus lowering the number of hits to the OCSP responder.

Second, it simplifies the client side OCSP implementation by enabling a situation where the client need only the ability to parse and recognize OCSP responses.

Third, it reduces the number of round trips the client needs to make in order to validate a certificate.

Fourth, it enables certificate validation in the event the client is not connected to a network and thus eliminates the need for clients to establish a new HTTP session with the responder.

This functionality has been specified as an extension to the TLS [TLS] protocol in Section 3.6 [TLS_EXT], but can be applied to any client-server protocol.

This profile RECOMMENDS that both TLS clients and servers implement the certificate status request extension mechanism for TLS.

6. Security Considerations

The following considerations apply in addition to the security consideration addressed in Section 5 of [OCSP]

6.1 Replay attacks

Because the use of nonce’s in this profile is optional, there is a possibility that an out of date OCSP response could be replayed, thus causing a client to accept good response when in fact there is
a more up to date response that specifies the status of revoked. In order to mitigate this attack, clients MUST have access to an accurate source of time and ensure that the OCSP responses they receive are sufficiently fresh.

Required clock accuracy is relative to the validity duration of the client’s OCSP responses. A client using responses that are good for one hour SHOULD have a clock that is within a few minutes correct time, while a client with 24-hour responses SHOULD be within an hour of correct time.

Clients that do not have an accurate source of date and time are vulnerable to service disruption due to rejection of fresh OCSP responses. If this problem is not repaired, a client with a sufficiently slow clock may also incorrectly accept expired responses for currently revoked certificates.

6.2 Man-in-the-middle attacks

To mitigate risk associated with this class of attack, the client must properly validate the signature on the response.

The use of signed responses in OCSP serves the purpose to authenticate the identity of the OCSP responder that has authority to sign request on the CA’s behalf.

Clients MUST ensure that they are communicating with an authorized responder by the rules described in [OCSP] Section 4.2.2.2.

6.3 Impersonation attacks

The use of signed responses in OCSP serves the purpose to authenticate the identity of OCSP Responder.

Clients must properly validate the signature of the OCSP response and the signature(s) on the OCSP response signer certificate to ensure an authorized responder created it.

6.4 Denial of service attacks

OCSP responders should take measures to prevent or mitigate denial of service attacks. In particular OCSP responders should not perform an unlimited number of resource intensive operations.

In the case where client requests are not signed, as specified by this profile, OCSP responders should take additional steps to detect an attack of this kind.
One such technique could be to attempt to match which response to send based on the hash of the request, this would protect against decode related attacks. However since extensions are supported not all requests for the same certificate will be the same as such it would also be necessary to support a full decode based lookup. As such this technique would only help defend against accidental attacks.

6.5 Modification of HTTP Headers

Values included in HTTP headers as described in Section 4 and 5, are not cryptographically protected, they may be manipulated by an attacker. Clients SHOULD use these values for caching guidance only and should ultimately rely on the values present in the signed OCSPResponse.

7. Acknowledgements

The authors wish to thank Magnus Nyström Of RSA Security, Inc., Jagjeet Sondh of Vodafone Group R&D and David Engberg of CoreStreet, Ltd. for their contributions to this specification.

8. References

8.1 Normative


8.2 Informative


9. Author’s Addresses

Alex Deacon
VeriSign, Inc.
487 E. Middlefield Road Phone: 1-650-426-3478
Mountain View, CA. USA Email: alex@verisign.com

Ryan Hurst
Microsoft
One Microsoft Way Phone: 1-425-707-8979
Redmond, WA. USA Email: rmh@microsoft.com

Appendix A. Useful Response Extensions

Appendix A.1. nextPublish Response Extension

Support for this extension is optional. This extension indicates the time at which the server will be issuing new information about the status of the certificate. If present can be used by the relying party to determine when it is acceptable to begin attempts for new revocation information.

The time specified in the nextPublish extension SHOULD be before the time specified in the nextUpdate field.

This profile RECOMMENDS this value be used by clients to determine when it is possible to check for more up to date information.

\[
\text{id-msft-nextPublish OBJECT IDENTIFIER ::= (1.3.6.1.4.1.311.21.4)}
\]

\[
\text{nextPublish EXTENSION ::= {\text{SYNTAX nextPublishSyntax}}}
\]

\[
\text{IDENTIFIED BY id-msft-nextPublish}
\]
nextPublishSyntax ::= Time

Time ::= CHOICE {
    utcTime    UTCTime,
    generalTime GeneralizedTime
}

Appendix B. Example OCSP Messages

Appendix B.1: OCSP Request

SEQUENCE {
    SEQUENCE {
        SEQUENCE {
            SEQUENCE {
                SEQUENCE {
                    OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
                    NULL
                }
                OCTET STRING
                C0 FE 02 78 FC 99 18 88 91 B3 F2 12 E9 C7 E1 B2
                1A B7 BF C0
                OCTET STRING
                0D FC 1D F0 A9 E0 F0 1C E7 F2 B2 13 17 7E 6F 8D
                15 7C D4 F6
                INTEGER
                6B 26 79 83 A4 9A B7 C2 3D FF 58 E8 81 AA A5 0E
            }
        }
    }
}

Appendix B.2: OCSP Response

SEQUENCE {
    ENUMERATED 0
    [0] {
        SEQUENCE {
            OBJECT IDENTIFIER ocspBasic (1 3 6 1 5 5 7 48 1 1)
            OCTET STRING, encapsulates {
                SEQUENCE {
                    SEQUENCE {
                        [1] {
                            SEQUENCE {
                                SET {
                                    SEQUENCE {
                                        ...
OBJECT IDENTIFIER organizationName (2 5 4 10)
PrintableString 'Example, Inc.'
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    PrintableString Example Trust Network'
  }
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    PrintableString 'Terms of use at https://www.example.com/rpa'
    '(c)02'
  }
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Example Class 3 International Server OCSP'
    'Responder'
  }
}

SEQUENCE {
  SEQUENCE {
    SEQUENCE {
      SEQUENCE {
        OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
        NULL
      }
      OCTET STRING
      C0 FE 02 78 FC 99 18 88 91 B3 F2 12 E9 C7 E1 B2
      1A B7 BF C0
      OCTET STRING
      0D FC 1D F0 A9 E0 F0 1C E7 F2 B2 13 17 7E 6F 8D
      15 7C D4 F6
      INTEGER
      6B 26 79 83 A4 9A B7 C2 3D FF 58 E8 81 AA A5 0E
    }
    [0]
    Error: Object has zero length.
SEQUENCE {
  OBJECT IDENTIFIER
      sha1withRSAEncryption (1 2 840 113549 1 1 5)
}

BIT STRING
  17 C3 A3 0B 87 1A A5 C9 39 D2 1E E4 49 9C 84 48
  DC E7 9A 68 89 77 BE 25 60 97 D9 FB 8C D0 C5 E8
  9B D2 25 F6 52 E9 BA 22 C8 FE C4 B6 B3 9F 1F 71
  58 FC BE 39 DC 9D 4E 85 00 8C F1 A9 92 CD 25 CA
  3C DC B9 61 46 76 87 BD A1 E9 F6 41 E2 B3 D6 7E
  E1 FD A1 5D 2D 08 7C 01 3F 2C 3A 39 60 F1 53 AD
  1E 81 E0 57 55 02 F7 D3 FC 9A F8 CA 09 DA 87 1E
  8A 93 01 58 E0 31 72 A1 4A 05 F7 3E 21 2F D7 93
[0] {
  SEQUENCE {
    SEQUENCE {
      SEQUENCE {
        [0] {
          INTEGER 2
        }
        INTEGER
          24 D4 27 7D 62 AC 2D 92 F8 D3 4E B1 A5 19 84 78
      }
    }
    OBJECT IDENTIFIER
      sha1withRSAEncryption (1 2 840 113549 1 1 5)
      NULL
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER
        organizationName (2 5 4 10)
        PrintableString "Example Trust Network"
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER
        organizationalUnitName (2 5 4 11)
        PrintableString 'Example, Inc.'
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER
      Deacon                   Expires - April 2005                
      [Page 17]
Lightweight OCSP Profile          October 2004

organizationUnitName (2 5 4 11)
  PrintableString
  'Example International Server CA - Class 3'
}
}
SEQUENCE {
  UTCTime 09/07/2002 00:00:00 GMT
  UTCTime 24/10/2011 23:59:59 GMT
}
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER
        organizationName (2 5 4 10)
        PrintableString 'Example, Inc.'
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER
        organizationUnitName (2 5 4 11)
        PrintableString 'Example Trust Network'
    }
  }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString
        'Example OCSP Responder'
    }
  }
}
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER
      rsaEncipherment (1 2 840 113549 1 1 1)
    NULL
  }
  BIT STRING, encapsulates {
    SEQUENCE {
      INTEGER
        00 CF 50 81 96 9A F5 D8 E2 DE 0B CF A3 A6 FB 46
        3E 88 0F 34 DF 5B 28 93 6D 32 EC D1 D0 0B 9B B4
        5C 7E 12 F0 22 79 1E 6E 0D C6 39 7E A8 C5 01 A7
        9F D8 93 D4 48 61 19 28 9A 93 7F ED 2A C4 CA 2C
        E3 47 0C 49 D6 7E D2 FB BC 2C 08 0D 9C FF 05 E6
        B0 EC 4B 93 1C AF 8E A9 F3 00 07 09 CF 9B 60 F6
        ED D1 B9 62 6F F1 A7 D3 61 0A 64 30 93 C9 43 4A
    }
  }
}

Deacon                   Expires - April 2005                [Page 18]
OE 3E A3 3E 47 D7 B2 0D B4 65 53 CC 0A FE CF E5
   [ Another 1 bytes skipped ]
INTEGER 65537
}
}
[3] {
SEQUENCE {
SEQUENCE {
OBJECT IDENTIFIER
   basicConstraints (2 5 29 19)
OCTET STRING, encapsulates {
SEQUENCE {}
}
}
}
SEQUENCE {
OBJECT IDENTIFIER
   certificatePolicies (2 5 29 32)
OCTET STRING, encapsulates {
SEQUENCE {
   SEQUENCE {
      OBJECT IDENTIFIER
         ’2 16 840 1 1 1 7 23 3’
SEQUENCE {
   SEQUENCE {
      OBJECT IDENTIFIER
         cps (1 3 6 1 5 5 7 2 1)
      IA5String
         ’https://www.example.com/rpa’
   }
}
}
}
}
SEQUENCE {
OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
OCTET STRING, encapsulates {
SEQUENCE {
   OBJECT IDENTIFIER
      ocspSigning (1 3 6 1 5 5 7 3 9)
}
}
}
SEQUENCE {
OBJECT IDENTIFIER keyUsage (2 5 29 15)
OCTET STRING, encapsulates {
   BIT STRING 7 unused bits
   ’1’B (bit 0)
SEQUENCE {
  OBJECT IDENTIFIER
    ocspNoCheck (1 3 6 1 5 5 7 48 1 5)
  OCTET STRING, encapsulates {
    NULL
  }
}

SEQUENCE {
  OBJECT IDENTIFIER
    sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
}

BIT STRING
  91 C2 C6 73 75 63 9A 6E A9 A6 F1 4D 99 F6 63 93
  83 78 2A DB DE 56 DE 86 B5 9A B5 E7 27 44 35 28
  2E F3 62 B4 9F 17 9F 2B 21 31 90 00 B0 86 E3 AE
  B6 2C 72 08 9B B8 9D A3 58 61 A8 01 35 8B 3C 6C
  6A D4 FF 01 FA E7 25 0D E8 D4 A5 8D 8E DF 3A 39
  11 DE 8E 7A 41 BC 56 48 98 A5 06 86 64 4E AD 0F
  5B D1 C7 BB 11 57 45 D4 06 F6 FF 3C 7E C5 78 7B
  68 C1 B6 71 9D 45 79 1D F7 03 0E 9E 6A 75 24 51
}

Full Copyright Statement

Copyright (C) The Internet Society 2004. This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.