

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

P. Hallam-Baker
Comodo Group Inc.
R. Stradling
Comodo CA Ltd.
July 16, 2012

DNS Certification Authority Authorization (CAA) Resource Record
draft-ietf-pkix-caa-11

Abstract

The Certification Authority Authorization (CAA) DNS Resource Record allows a DNS domain name holder to specify one or more Certification Authorities (CAs) authorized to issue certificates for that domain. CAA resource records allow a public Certification Authority to implement additional controls to reduce the risk of unintended certificate mis-issue.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Definitions	3
1.1. Requirements Language	3
1.2. Defined Terms	3
2. Introduction	4
2.1. The CAA RR type	5
3. Certification Authority Processing	7
3.1. Use of DNS Security	8
3.2. Archive	9
4. Mechanism	9
4.1. Syntax	9
4.1.1. Canonical Presentation Format	10
4.2. CAA issue Property	11
4.3. CAA iodef Property	12
5. Security Considerations	12
5.1. Non-Compliance by Certification Authority	13
5.2. Mis-Issue by Authorized Certification Authority	13
5.3. Suppression or spoofing of CAA records	13
5.4. Denial of Service	14
5.5. Abuse of the Critical Flag	14
6. IANA Considerations	14
6.1. Registration of the CAA Resource Record Type	14
6.2. Certification Authority Authorization Properties	15
6.3. Acknowledgements	15
7. References	15
7.1. Normative References	15
7.2. Informative References	16
Authors' Addresses	16

1. Definitions

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 [RFC2119].

1.2. Defined Terms

The following terms are used in this document:

Authorization Entry: An authorization assertion that grants or denies a specific set of permissions to a specific group of entities.

Canonical Domain Name: A Domain Name that is not an alias. See [RFC1035] and future successors for definition of CNAME alias records.

Canonical Domain Name Value: The value of a Canonical Domain Name. The value resulting from applying alias transformations to a Domain Name that is not canonical.

Certificate: An X.509 Certificate, as specified in [RFC5280].

Certificate Evaluator: A party other than a Relying Party that evaluates the trustworthiness of certificates issued by Certification Authorities.

Certification Authority (CA): An Issuer that issues Certificates in accordance with a specified Certificate Policy.

Certificate Policy (CP): Specifies the criteria that a Certification Authority undertakes to meet in its issue of certificates. See [RFC3647].

Certification Practices Statement (CPS): Specifies the means by which the criteria of the Certificate Policy are met. In most cases this will be the document against which the operations of the Certification Authority are audited. See [RFC3647].

Domain: The set of resources associated with a DNS Domain Name.

Domain Name: A DNS Domain name as specified in [RFC1035] and revisions.

Domain Name System (DNS): The Internet naming system specified in [RFC1035] and revisions.

DNS Security (DNSSEC): Extensions to the DNS that provide authentication services as specified in [RFC4033]. and revisions.

Issuer: An entity that issues Certificates. See [RFC5280].

Extended Issuer Authorization Set: The most specific Issuer Authorization Set that is active for a domain. This is either the Issuer Authorization Set for the domain itself, or if that is empty, the Issuer Authorization Set for the corresponding Public Delegation Point.

Issuer Authorization Set: The set of Authorization Entries for a domain name that are flagged for use by Issuers. Analogous to an Access Control List but with no ordering specified.

Property: The tag-value portion of a CAA Resource Record.

Property Tag: The tag portion of a CAA Resource Record.

Property Value: The value portion of a CAA Resource Record.

Public Delegation Point: The Domain Name suffix under which DNS names are delegated by a public DNS registry such as a Top Level Directory.

Public Key Infrastructure X.509 (PKIX): Standards and specifications issued by the IETF that apply the [X.509] certificate standards specified by the ITU to Internet applications as specified in [RFC5280] and related documents.

Resource Record (RR): A set of attributes bound to a Domain Name as defined in [RFC1035].

Relying Party: A party that makes use of an application whose operation depends on use of a Certificate for making a security decision. See [RFC5280].

Relying Application: An application whose operation depends on use of a Certificate for making a security decision.

2. Introduction

The Certification Authority Authorization (CAA) DNS Resource Record allows a DNS domain name holder to specify the Certification

Authorities authorized to issue certificates for that domain. Publication of CAA resource records allow a public Certification Authority (CA) to implement additional controls to reduce the risk of unintended certificate mis-issue.

Like the TLSA record defined in DNS-Based Authentication of Named Entities (DANE) [DANE], CAA records are used as a part of a mechanism for checking PKIX certificate data. The distinction between the two specifications is that CAA records specify a authorization control to be performed by a certificate issuer before issue of a certificate and TLSA records specify a verification control to be performed by a Relying Party after the certificate is issued.

Conformance with a published CAA record is a necessary but not sufficient condition for issuance of a certificate. Before issuing a certificate, a PKIX CA is required to validate the request according to the policies set out in its Certificate Policy. In the case of a public CA that validates certificate requests as a third party, the certificate will be typically issued under a public trust anchor certificate embedded in one or more relevant Relying Applications.

Criteria for inclusion of embedded trust anchor certificates in applications are outside the scope of this document. Typically such criteria require the CA to publish a Certificate Practices Statement (CPS) that specifies how the requirements of the Certificate Policy (CP) are achieved. It is also common for a CA to engage an independent third party auditor to prepare an annual audit statement of its performance against its CPS.

A set of CAA records describes only current grants of authority to issue certificates for the corresponding DNS domain. Since a certificate is typically valid for at least a year, it is possible that a certificate that is not conformant with the CAA records currently published was conformant with the CAA records published at the time that the certificate was issued. Relying Applications MUST NOT use CAA records as part of certificate validation.

CAA Records MAY be used by Certificate Evaluators as a possible indicator of a security policy violation. Such use SHOULD take account of the possibility that published CAA records changed between the time a certificate was issued and the time at which the certificate was observed by the Certificate Evaluator.

2.1. The CAA RR type

A CAA RR consists of a flags byte and a tag-value pair referred to as a property. Multiple properties MAY be associated with the same

domain name by publishing multiple CAA RRs at that domain name. The following flag is defined:

Issuer Critical: If set, indicates that the corresponding property entry tag **MUST** be understood if the semantics of the CAA record are to be correctly interpreted by an issuer.

Issuers **MUST NOT** issue certificates for a domain if the Extended Issuer Authorization Set contains unknown property entry tags that have the Critical bit set.

The following property tags are defined:

issue <Issuer Domain Name> [; <tag=value>]* : The issue property entry authorizes the holder of the domain name <Issuer Domain Name> or a party acting under the explicit authority of the holder of that domain name to issue certificates for the domain in which the property is published.

iodef <URL> : Specifies a URL to which an issuer **MAY** report certificate issue requests that are inconsistent with the issuer's Certification Practices or Certificate Policy, or that a certificate evaluator may use to report observation of a possible policy violation. The IODEF format is used [RFC5070].

The following example informs CAs that certificates **MUST NOT** be issued except by the holder of the domain name 'ca.example.net' or an authorized agent thereof. Since the policy is published at the Public Delegation Point, the policy applies to all subordinate domains under example.com.

```
$ORIGIN example.com
.      CAA 0 issue "ca.example.net"
```

If the domain name holder specifies one or more iodef properties, a certificate issuer **MAY** report invalid certificate requests to that address. In the following example the domain name holder specifies that reports **MAY** be made by means of email with the IODEF data as an attachment, a Web service [RFC6546] or both:

```
$ORIGIN example.com
.      CAA 0 issue "ca.example.net"
.      CAA 0 iodef "mailto:security@example.com"
.      CAA 0 iodef "http://iodef.example.com/"
```

A certificate issuer **MAY** specify additional parameters that allow customers to specify additional parameters governing certificate issuance. This might be the Certificate Policy under which the

certificate is to be issued, the authentication process to be used might be specified or an account number specified by the CA to enable these parameters to be retrieved.

For example, the CA 'ca.example.net' has requested its customer 'example.com' to specify the CA's account number '230123' in each of the customer's CAA records.

```
$ORIGIN example.com
.      CAA 0 issue "ca.example.net; account=230123"
```

The syntax and semantics of such parameters is left to site policy and is outside the scope of this document.

Future versions of this specification MAY use the critical flag to introduce new semantics that MUST be understood for correct processing of the record, preventing conforming CAs that do not recognize the record from issuing certificates for the indicated domains.

In the following example, the property 'tbs' is flagged as critical. Neither the example.net CA, nor any other issuer is authorized to issue under either policy unless the processing rules for the 'tbs' property tag are understood.

```
$ORIGIN example.com
.      CAA 0 issue "ca.example.net; policy=ev"
.      CAA 128 tbs "Unknown"
```

Note that the above restrictions only apply to issue of certificates. Since the validity of an end entity certificate is typically a year or more, it is quite possible that the CAA records published at a domain will change between the time a certificate was issued and validation by a relying party.

3. Certification Authority Processing

Before issuing a certificate, a compliant CA MUST check for publication of a relevant CAA Resource Record(s). If such record(s) are published, the requested certificate MUST consistent with them if it is to be issued. If the certificate requested is not consistent with the relevant CAA RRs, the CA MUST NOT issue the certificate.

The Issuer Authorization Set for a domain name consists of the set of all CAA Authorization Entries declared for the canonical form of the specified domain.

The DNS defines the CNAME and DNAME mechanisms for specifying domain name aliases. The canonical name of a DNS name is the name that results from performing all DNS alias operations. An issuer **MUST** perform CNAME and DNAME processing as defined in the DNS specifications [RFC1035] to resolve CAA records.

The Extended Issuer Authorization Set for a domain name is determined as follows:

- o If the Issuer Authorization Set of the domain is not empty, the Extended Issuer Authorization Set is the Issuer Authorization Set of the domain.
- o If the immediately superior node in the DNS hierarchy is a Public Delegation Point, the Extended Issuer Authorization Set is empty.
- o Otherwise the Extended Issuer Authorization Set is that of the immediately superior node in the DNS hierarchy.

For example, if the zone example.com has a CAA record defined for caa.example.com and no other domain in the zone, the Issuer Authorization Set is empty for all domains other than caa.example.com. The Extended Issuer Authorization Set is empty for example.com (because .com is a Public Delegation Point) and for x.example.com. The Extended Issuer set for x.caa.example.com, x.x.caa.example.com, etc. is the Issuer Authorization Set for caa.example.com.

If the Extended Issuer Authorization Set for a domain name is not empty, a Certification Authority **MUST NOT** issue a certificate unless the certificate conforms to at least one authorization entry in the Extended Issuer Authorization Set.

3.1. Use of DNS Security

Use of DNSSEC to authenticate CAA RRs is strongly **RECOMMENDED** but not required. An issuer **MUST NOT** issue certificates if doing so would conflict with the corresponding extended issuer authorization set, irrespective of whether the corresponding DNS records are signed.

Use of DNSSEC allows an issuer to acquire and archive a non-repudiable proof that they were authorized to issue certificates for the domain. Verification of such archives **MAY** be an audit requirement to verify CAA record processing compliance. Publication of such archives **MAY** be a transparency requirement to verify CAA record processing compliance.

3.2. Archive

A compliant issuer SHOULD maintain an archive of the DNS transactions used to verify CAA eligibility.

In particular an issuer SHOULD ensure that where DNSSEC data is available that the corresponding signature and NSEC/NSEC3 records are preserved so as to enable later compliance audits.

4. Mechanism

4.1. Syntax

A CAA RR contains a single property entry consisting of a tag value pair. Each tag represents a property of the CAA record. The value of a CAA property is that specified in the corresponding value field.

A domain name MAY have multiple CAA RRs associated with it and a given property MAY be specified more than once.

The CAA data field contains one property entry. A property entry consists of the following data fields:

0-1-2-3-4-5-6-7-	0-1-2-3-4-5-6-7-	
Flags	Tag Length = n	
+-----+	+-----+	+...+-----+
Tag char 0	Tag Char 1	... Tag Char n-1
+-----+	+-----+	+...+-----+
+-----+	+-----+	+.....+-----+
Value byte 0	Value byte 1 Value byte m-1
+-----+	+-----+	+.....+-----+

Where n is the length specified in the Tag length field and m is the remaining octets in the Value field ($m = d - n - 2$) where d is the length of the RDATA section.

The data fields are defined as follows:

Flags: One octet containing the following fields:

Bit 0: Issuer Critical Flag If the value is set (1), the critical flag is asserted and the property MUST be understood if the CAA record is to be correctly processed by a certificate issuer.

A Certification Authority MUST NOT issue certificates for any Domain that contains a CAA critical property for an unknown or unsupported property tag that for which the issuer critical flag is set.

Note that according to the conventions set out in RFC 1035 [RFC1035] Bit 0 is the Most Significant Bit and Bit 7 is the Least Significant Bit. Thus the Flags value 1 means that bit 7 is set while a value of 128 means that bit 0 is set according to this convention.

All other bit positions are reserved for future use.

To ensure compatibility with future extensions to CAA, DNS records compliant with this version of the CAA specification MUST clear (set to "0") all reserved flags bits. Applications that interpret CAA records MUST ignore the value of all reserved flag bits.

Tag Length: A single octet containing an unsigned integer specifying the tag length in octets. The tag length MUST be at least 1 and SHOULD be no more than 15.

Tag: The property identifier, a sequence of ASCII characters.

Tag values MAY contain ASCII characters 'a' through 'z', 'A' through 'Z' and the numbers 0 through 9. Tag values SHOULD NOT contain any other characters. Matching of tag values is case insensitive.

Tag values submitted for registration by IANA MUST NOT contain any characters other than the (lowercase) ASCII characters 'a' through 'z' and the numbers 0 through 9.

Value: A sequence of octets representing the property value. Property values are encoded as binary values and MAY employ sub-formats.

The length of the value field is specified implicitly as the remaining length of the enclosing Resource Record data field.

4.1.1. Canonical Presentation Format

The canonical presentation format of the CAA record is as follows:

CAA <flags> <tag> <value>

Where:

Flags: Is an unsigned integer between 0 and 255.

Tag: Is a non-zero sequence of ASCII letter and numbers in lower case.

Value: Is the US-ASCII text Encoding of the value field

4.2. CAA issue Property

The issue property tag is used to request that certificate issuers perform CAA issue restriction processing for the domain and to grant authorization to specific certificate issuers.

The CAA issue property value has the following sub-syntax (specified in ABNF as per [RFC5234]).

Property = space [domain] * (space ";" parameter) space

domain = label *("." label)

label = 1* (ALPHA / DIGIT / "-")

space = *(SP / HTAB)

parameter = / space tag "=" value

tag = 1* (ALPHA / DIGIT)

value = *VCHAR | DQUOTE *(%x20-21 / %x23-7E) DQUOTE

A CAA record with an issue parameter tag that does not specify a domain name is a request that certificate issuers perform CAA issue restriction processing for the corresponding domain without granting authorization to any certificate issuer.

This form of issue restriction would be appropriate to specify that no certificates are to be issued for the domain in question.

For example, the following CAA record set requests that no certificates be issued for the domain 'nocerts.example.com' by any certificate issuer.

```
nocerts.example.com      CAA 0 issue ";"
```

A CAA record with an issue parameter tag that specifies a domain name is a request that certificate issuers perform CAA issue restriction processing for the corresponding domain and grants authorization to the certificate issuer specified by the domain name.

For example, the following CAA record set requests that no certificates be issued for the domain 'certs.example.com' by any certificate issuer other than the example.net certificate issuer.

```
certs.example.com      CAA 0 issue "example.net"
```

CAA authorizations are additive. thus the result of specifying both the empty issuer and a specified issuer is the same as specifying just the specified issuer alone.

An issuer MAY choose to specify issuer-parameters that further constrain the issue of certificates by that issuer. For example specifying that certificates are to be subject to specific validation policies, billed to certain accounts or issued under specific trust anchors.

The syntax and semantics of issuer-parameters are determined by the issuer alone.

4.3. CAA iodef Property

The iodef property specifies a means of reporting certificate issue requests or cases of certificate issue for the corresponding domain, that violate the security policy of the issuer or the domain name holder.

The Incident Object Description Exchange Format (IODEF) [RFC5070] is used to present the incident report in machine readable form.

The iodef property takes a URL as its parameter. The URL scheme type determines the method used for reporting:

mailto: The IODEF incident report is reported as a MIME email attachment to an SMTP email that is submitted to the mail address specified. The mail message sent SHOULD contain a brief text message to alert the recipient to the nature of the attachment.

http or https: The IODEF report is submitted as a Web Service request to the HTTP address specified using the protocol specified in [RFC6546].

5. Security Considerations

CAA Records assert a security policy that the holder of a domain name wishes to be observed by certificate issuers. The effectiveness of CAA records as an access control mechanism is thus dependent on observance of CAA constraints by issuers.

The objective of the CAA record properties described in this document is to reduce the risk of certificate mis-issue rather than avoid reliance on a certificate that has been mis-issued. DANE [DANE] describes a mechanism for avoiding reliance on mis-issued certificates.

5.1. Non-Compliance by Certification Authority

CAA records offer CAs a cost-effective means of mitigating the risk of certificate mis-issue: The cost of implementing CAA checks is very small and the potential costs of a mis-issue event include the removal of an embedded trust anchor.

5.2. Mis-Issue by Authorized Certification Authority

Use of CAA records does not prevent mis-issue by an authorized Certification Authority. , i.e., a CA that is authorized to issue certificates for the domain in question by CAA records..

Domain name holders SHOULD verify that the CAs they authorize to issue certificates for their domains employ appropriate controls to ensure that certificates are issued only to authorized parties within their organization.

Such controls are most appropriately determined by the domain name holder and the authorized CA(s) directly and are thus out of scope of this document.

5.3. Suppression or spoofing of CAA records

Suppression of the CAA record or insertion of a bogus CAA record could enable an attacker to obtain a certificate from a CA that was not authorized to issue for that domain name.

A CA MUST mitigate this risk by employing DNSSEC verification whenever possible and rejecting certificate requests in any case where it is not possible to verify the non-existence or contents of a relevant CAA record.

In cases where DNSSEC is not deployed in a corresponding domain, a CA SHOULD attempt to mitigate this risk by employing appropriate DNS security controls. For example all portions of the DNS lookup process SHOULD be performed against the authoritative name server. Data cached by third parties MUST NOT be relied on but MAY be used to support additional anti-spoofing or anti-suppression controls.

5.4. Denial of Service

Introduction of a malformed or malicious CAA RR could in theory enable a Denial of Service attack.

This specific threat is not considered to add significantly to the risk of running an insecure DNS service.

An attacker could, in principle, perform a Denial of Service attack against an issuer by requesting a certificate with a maliciously long DNS name. In practice, the DNS protocol imposes a maximum name length and CAA processing does not exacerbate the existing need to mitigate Denial of Service attacks to any meaningful degree.

5.5. Abuse of the Critical Flag

A Certification Authority could make use of the critical flag to trick customers into publishing records which prevent competing Certification Authorities from issuing certificates even though the customer intends to authorize multiple providers.

In practice, such an attack would be of minimal effect since any competent competitor that found itself unable to issue certificates due to lack of support for a property marked critical SHOULD investigate the cause and report the reason to the customer who will thus discover that they had been deceived.

6. IANA Considerations

6.1. Registration of the CAA Resource Record Type

[Note to IANA, the CAA resource record has already been assigned. On issue of this draft as an RFC, the record should be updated to reflect this document as the authoritative specification and this paragraph (but not the following ones deleted)]

IANA has assigned Resource Record Type 257 for the CAA Resource Record Type and added the line depicted below to the registry named Resource Record (RR) TYPEs and QTYPEs as defined in BCP 42 [RFC6195] and located at <http://www.iana.org/assignments/dns-parameters>.

RR Name	Value and meaning	Reference
-----	-----	-----
CAA	257 Certification Authority Restriction	[RFC-THIS]

6.2. Certification Authority Authorization Properties

[Note to IANA, this is a new registry that needs to be created and this paragraph but not the following ones deleted.]

IANA has created the Certification Authority Authorization Properties registry with the following initial values:

Tag	Meaning	Reference
-----	-----	-----
issue	Authorization Entry by Domain	[RFC-THIS]
iodef	Report incident by means of IODEF format report	[RFC-THIS]
auth	Reserved	
path	Reserved	
policy	Reserved	

Addition of tag identifiers requires a public specification and expert review as set out in [RFC6195]

6.3. Acknowledgements

The authors would like to thank the following people who contributed to the design and documentation of this work item: Chris Evans, Stephen Farrell, Jeff Hodges, Paul Hoffman, Stephen Kent, Adam Langley, Ben Laurie, Chris Palmer, Scott Schmit, Sean Turner and Ben Wilson.

7. References

7.1. Normative References

- [DANE] P. Hoffman., J. Schlyter, "draft-ietf-dane-protocol-23: Replace with reference to RFC before issue.", 2012.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC5070] Danyliw, R., Meijer, J., and Y. Demchenko, "The Incident Object Description Exchange Format", RFC 5070, December 2007.

- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [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.
- [RFC6195] Eastlake, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6195, March 2011.
- [RFC6546] Trammell, B., "Transport of Real-time Inter-network Defense (RID) Messages over HTTP/TLS", RFC 6546, April 2012.
- [X.509] International Telecommunication Union, "ITU-T Recommendation X.509 (11/2008): Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, November 2008.

7.2. Informative References

- [RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, November 2003.

Authors' Addresses

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com

Rob Stradling
Comodo CA Ltd.

Email: rob.stradling@comodo.com

PKIX
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2013

M. Pritikin, Ed.
Cisco Systems, Inc.
P. Yee, Ed.
AKAYLA, Inc.
D. Harkins, Ed.
Aruba Networks
July 10, 2012

Enrollment over Secure Transport
draft-ietf-pkix-est-02

Abstract

This document profiles certificate enrollment for clients using Certificate Management over CMS (CMC) messages over a secure transport. This profile, called Enrollment over Secure Transport (EST), describes a simple yet functional certificate management protocol targeting simple Public Key Infrastructure clients that need to acquire client certificate(s) and associated Certification Authority (CA) certificate(s).

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Requirements Language	5
2. Operational Scenario Overviews	5
2.1. Obtaining CA Certificates	6
2.2. Initial Enrollment	6
2.2.1. Previously Installed Signature Certificate	7
2.2.2. Username/Password Distributed Out-of-Band	7
2.2.3. RA Authentication	7
2.3. Re-Enrollment	7
2.3.1. Re-Enrollment of Signature Certificates	7
2.3.2. Re-Enrollment of Key Establishment Certificates	8
2.4. Server Key Generation	8
2.5. Full CMC messages	8
2.6. CSR Attributes Request	8
3. Protocol Design and Layering	8
3.1. Application Layer Design	11
3.2. HTTP Layer Design	12
3.2.1. HTTP headers for control	12
3.2.2. HTTP URIs for control	13
3.2.3. HTTP-Based Client Authentication	14
3.2.4. Message types	15
3.3. TLS Layer Design	16
3.3.1. TLS for transport security	16
3.3.1.1. TLS-Based Server Authentication	16
3.3.1.2. TLS-Based Client Authentication	17
3.4. Proof-of-Possession	17
3.5. Linking Identity and POP information	18
4. Protocol Exchange Details	19
4.1. Server Authorization	19
4.2. Client Authorization	20
4.3. Distribution of CA certificates	20
4.3.1. Distribution of CA certificates response	21
4.4. Simple Enrollment of Clients	22
4.4.1. Simple Re-Enrollment of Clients	23
4.4.2. Simple Enroll and Re-Enroll Response	24
4.5. Full CMC	24
4.5.1. Full CMC Request	25
4.5.2. Full CMC Response	25
4.6. Server-side Key Generation	26

4.6.1.	Server-side Key Generation Request	26
4.6.2.	Server-side Key Generation Response	26
4.7.	CSR Attributes	27
4.7.1.	CSR Attributes Request	27
4.7.2.	CSR Attributes Response	27
5.	IANA Considerations	28
6.	Security Considerations	30
7.	References	32
7.1.	Normative References	32
7.2.	Informative References	33
Appendix A.	Server Discovery	34
Appendix B.	External TLS concentrator	34
Appendix C.	CGI Server implementation	35
Appendix D.	Operational Scenario Example Messages	35
D.1.	Obtaining CA Certificates	35
D.2.	Previously Installed Signature Certificate	36
D.3.	Username/Password Distributed Out-of-Band	38
D.4.	Re-Enrollment	41
D.5.	Server Key Generation	42
Authors' Addresses	46

1. Introduction

This document specifies a protocol for certificate Enrollment over Secure Transport (EST). EST is designed to be easily implemented by clients and servers using common "off the shelf" PKI, HTTP, and TLS components. An EST server providing certificate management functions is operated by (or on behalf of) a CA or RA. The goal is to provide a small set of functions for certificate enrollment that are simpler to implement and use than full CMP or CMC. While less functional than those protocols, EST satisfies basic needs by providing an easily implemented means for both autonomous devices as well as user-operated computers to request certificates.

The TLS [RFC4346] (or later) protocol is used with a limited set of features of the Certificate Management over CMS (CMC) [RFC5272] to provide the security for EST. CMC "simple" messages are used for certificate requests and responses. EST also allows the optional use of "full" CMC messages if needed, but compliant EST client and server implementations need not support full CMC messages. EST adopts the CMP model for CA certificate rollover, but does not incorporate its syntax or protocol. An EST server supports several means of authenticating a certificate requester, leveraging the layering of the protocols that make up EST. EST servers are extensible in that new requests may be defined which provide additional capabilities not specified in the base RFC. One non-CMC-based extension (requesting of CSR attributes) is defined in this document.

EST works by transporting CMC and other messages securely over an HTTPS transport in which HTTP headers and content types are used in conjunction with TLS security. TCP/IP sits under HTTPS; this document does not specify EST over DTLS or UDP. Figure 1 shows how the layers build upon each other.

EST Layering:

Protocols:

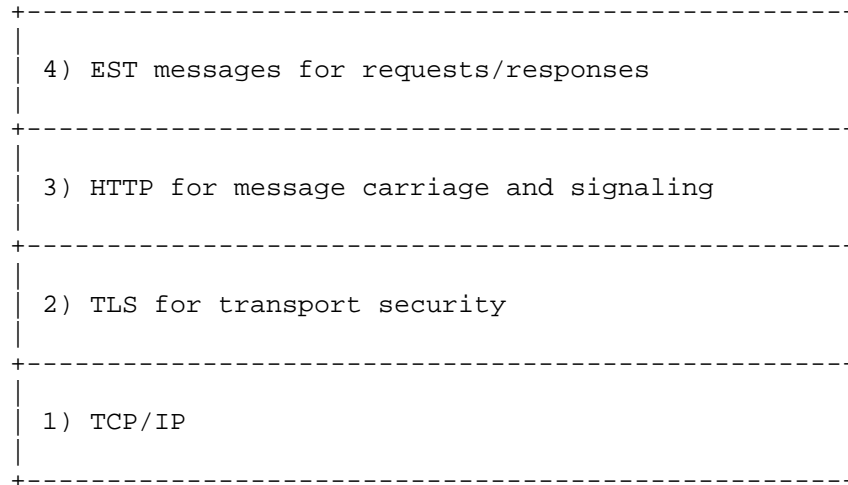


Figure 1

[[EDNOTE: Comments such as this one, included within double brackets and initiated with an 'EDNOTE', are for editorial use and shall be removed as the document is polished.]]

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 [RFC2119].

2. Operational Scenario Overviews

This EST specification provides a profile of CMC using round-trip communication between the EST client and the EST server in which CMC "simple" messages are transmitted. The basic framework can be extended with additional capabilities that leverage the transport and security features supplied by EST.

The EST server is assumed to be configured with an identity certificate and appropriate policy regarding authenticated clients. An EST server likely communicates with a CA for signing but for simplicity we indicate that a 'certificate is signed' as if by the EST server. The EST client is initially configured with only the HTTPS URI of the EST server.

This section illustrates several potential certificate enrollment and rekey scenarios supported by this profile. For clarity the EST client is assumed to perform "Obtaining CA Certificates" before performing other operations.

This section does not intend to place any limits or restrictions on the use of full CMC. Sections 2.1-2.3 very closely mirror the exact text of the Scenarios Appendix of [RFC6403] with such modifications as are appropriate for this profile. (Our thanks are extended to the authors of that document).

2.1. Obtaining CA Certificates

The EST client can request a copy of the current CA certificates.

Following the logic laid out in Section 3.3.1.1 the EST client authenticates and authorizes the EST server. Available options include verifying the EST server URI against the EST server certificate (similar to a common HTTPS exchange), or using a "pinned" copy of the CA certificate. As a fallback the EST client can accept manual authentication performed by the end user (in which case the certificates received are be "pinned" for authenticating future communications with the EST server).

Client authentication is not required for this exchange so it is trivially serviced by the EST server.

2.2. Initial Enrollment

The EST client can enroll with the CA server by submitting an enrollment request to the EST server. Following the logic laid out in Section 3.3.1.1 the EST client authenticates and authorizes the EST server.

Three scenarios for the EST server to authenticate the enrollment requests are:

1. Previously installed signature certificate (e.g., Manufacturer Installed Certificate or 3rd party issued certificate);
2. Username/password distributed out-of-band
3. RA authentication

2.2.1. Previously Installed Signature Certificate

If the EST client has a previously installed signature certificate issued by a trust anchor listed by the EST server during the TLS handshake it can be used to authenticate the request for a new certificate. The EST client responds to the TLS certificate request with the existing certificate as defined for TLS. The EST server will recognize the authorization of the previously installed certificate and issue an appropriate certificate to the EST client.

2.2.2. Username/Password Distributed Out-of-Band

If the EST client did not have a previously installed signature certificate, or if the EST server wishes additional authentication information, the EST server requests the EST client submit a username/password using the HTTP authentication methods.

2.2.3. RA Authentication

In this scenario the EST client submits the certification request using either the /simpleEnroll or /fullCMC method. The EST server forwards the received request using either CMC or other methods out-of-scope of this document.

2.3. Re-Enrollment

The EST client can renew/rekey an existing client certificate by submitting a re-enrollment request to the EST server. As for initial enrollment the EST server authenticates the client using any combination of the existing client certificate and an HTTP username/password. Because the client specifically requests renew/rekey the EST server can adjust its policy accordingly.

There are two scenarios to support the renew/rekey of clients that are already enrolled. One addresses the renew/rekey of signature certificates and the other addresses the renew/rekey of key establishment certificates. Typically, organizational policy will require certificates to be currently valid to be renewed/rekeyed, and it may require initial enrollment to be repeated when renew/rekey is not possible.

2.3.1. Re-Enrollment of Signature Certificates

When a signature certificate is re-enrolled the existing certificate is used by the EST client for authentication. The EST server uses this information along with any supplemental HTTP authentication information and the certification request itself to determine the parameters of the certificate to issue in response. If there is no

current signature certificate available the EST server can fallback on the HTTP authentication method. The certification request message will include the same Subject/SubjectAltName as the current signature certificate.

2.3.2. Re-Enrollment of Key Establishment Certificates

When a key establishment certificate is re-enrolled an existing signature certificate is used by the EST client for authentication. The EST server uses this information along with any supplemental HTTP authentication information and the certification request itself to determine the parameters of the certificate to issue in response. If there is no current signature certificate available the EST server can fallback on the HTTP authentication method. The certification request message will include the same Subject/SubjectAltName as the current key establishment certificate.

2.4. Server Key Generation

The EST client can request a server generated certificate and keypair. The EST server authenticates the client using any existing client signature certificate and/or HTTP username/password.

2.5. Full CMC messages

Full CMC messages can be transported thus allowing access to functionality not provided by the simple CMC message. "Full" CMC messages are as defined in Sections 3.2 and 4.2 of [RFC5272]. Support for full CMC message transport is optional for EST clients and servers.

2.6. CSR Attributes Request

Prior to sending an enrollment request to an EST server, an EST client may request that the EST server send it a (set of) additional attribute(s) that the client is requested to supply in the subsequent enrollment (certificate signing) request.

3. Protocol Design and Layering

The following provides an expansion of Figure 1 describing how the layers are used. Each aspect is described in more detail in the sections below.

EST Layering:

Protocols and uses:

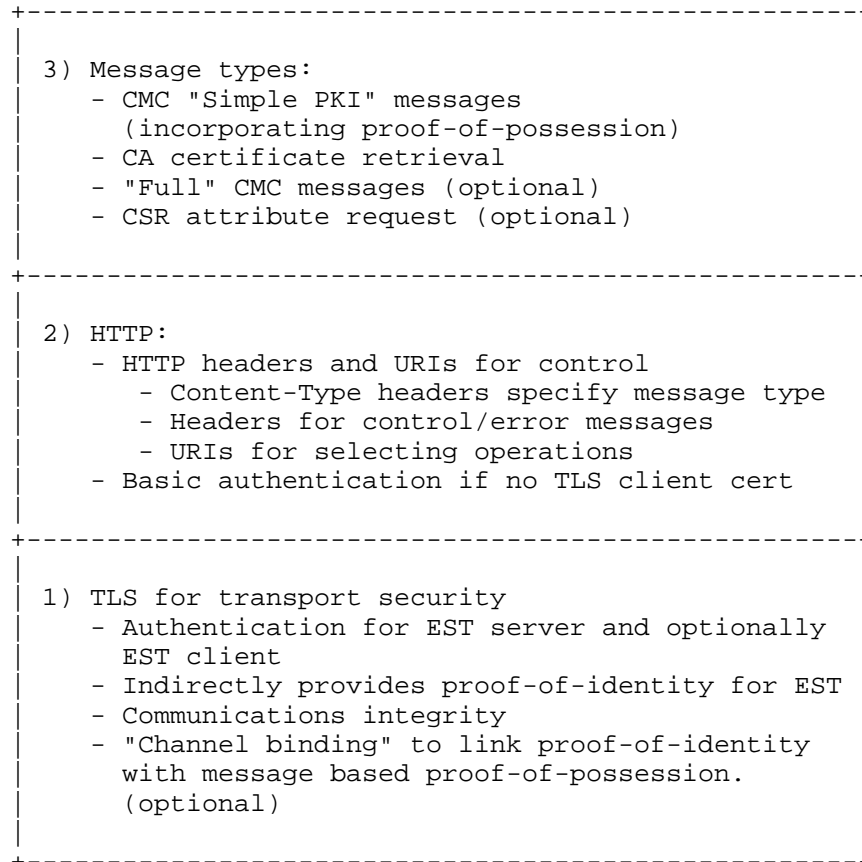


Figure 2

Specifying HTTPS as the secure transport for PKI enrollment messages introduces two 'layers' for communication of authentication and control messages during the protocol exchange: TLS and HTTP.

The TLS layer provides message authentication and integrity during transport. The proof-of-identity is supplied by either the certificate exchange during the TLS handshake or within the HTTP layer headers. The message type along with control/error messages are included in the HTTP headers.

The TLS and HTTP layer provided proof-of-identity means the CMC [RFC5272] Section 3.1 note that "the Simple PKI Request MUST NOT be

used if a proof-of-identity needs to be included" is not applicable and thus the "Simple PKI" message types are used.

The TLS layer certificate exchange provides a method for authorizing client enrollment requests using existing certificates. Such existing certificates may have been issued by the Certification Authority (CA) (from which the client is requesting a certificate) or they may have been issued under a distinct PKI (e.g., an IEEE 802.1AR IDevID [IDevID] credential).

Proof-of-possession is a distinct issue from proof-of-identity and is included in the "Simple PKI" message type as described in Section 3.4. A method of linking proof-of-identity and proof-of-possession is described in Section 3.5.

This document also defines transport for the full CMC [RFC5272] specification compliant with CMC Transport Protocols [RFC5273].

During the protocol operations various different certificates can be used. The following table provides an informative overview. End entities MAY have one or more certificates of each type as is appropriate:

Certificates/Trust-anchors and their corresponding uses:

End Entity	Issuer	Use
EST server	The CA served by the EST server	To authenticate servers that have certs issued by the CA Section: 3.3.1.1.
EST server	An unrelated CA e.g., a Web site CA	To authenticate servers that have certs issued by Web site CAs Section: 3.3.1.1.
EST client Trust Anchor Database	Trust anchors for third party CAs e.g., a list of Web site CA root certs	EST clients can leverage a trust anchor database to authenticate EST servers using a configured URI Section: 3.3.1.1
EST client	An unrelated CA e.g., a device manufacturer	To authenticate clients that have not yet enrolled Section: 3.3.1.2
EST client	The CA served by the EST server	To authenticate clients that have already enrolled (for re-enroll or obtaining additional certs) Section: 3.3.1.2
EST client	The CA served by the EST server	Clients can obtain certs that can not be used for EST authentication (e.g., Key Encryption certs) Section: 4.4.1

Figure 3

3.1. Application Layer Design

An EST client SHOULD have its own client certificate suitable for TLS client authentication (e.g., the digitalSignature bit is set). The client certificate, if available, is used when authenticating to the EST server. This certificate MAY also be used by the client with other certificate consuming protocols. If a client does not have a certificate, then the client MUST have HTTP Basic or Digest

authentication credentials (see Section 3.2.3). HTTP authentication provides a bootstrap for clients that have not yet been issued an initial certificate. EST clients obtaining a certificates for other protocol purposes are RECOMMENDED to first obtain an appropriate digitalSignature certificate for use when authenticating to the EST server.

The client also SHOULD also have a CA certificate that will be used to authenticate the EST server.

An EST client MUST be capable of generating and parsing simple CMC messages (see Section 4.4). Generating and parsing full CMC messages is optional (see Section 4.5). The client MUST also be able to request CA certificates from the EST server and parse the returned "bag" of certificates (see Section 4.3). Requesting CSR attributes and parsing the returned list of attributes is optional (see Section 4.7).

3.2. HTTP Layer Design

HTTP is used to transport EST requests and responses. Specific URIs are provisioned for handling each type of request as described in Section 3.2.2. HTTP is also used for client authentication services when TLS client authentication is not available due to lack of a client certificate suitable for use by TLS, as detailed in Section 3.2.3. HTTP message types are used to convey EST requests and responses as specified in Figure 5.

3.2.1. HTTP headers for control

This document profiles the HTTP content-type header (as defined in [RFC2046], but see Figure 5 for specific values) to indicate the message type for EST messages and to specify EST control messages. The HTTP Status value is used to communicate success or failure of control messages. Support for the HTTP username/password methods is profiled for when a client does not have a suitable client certificate.

CMC does not provide specific messages for certificate renewal and certificate rekey. This profile defines the renewal and rekey behavior of both the client and server. It does so by specifying the HTTP control mechanisms employed by the client and server without requiring a new CMC message type.

Various media types as indicated in the HTTP content-type header are used to transport EST messages. Valid media types are specified in Section 3.2.4.

3.2.2. HTTP URIs for control

This profile supports four operations indicated by specific URIs:

Operations and their corresponding URIs:

Operation	Operation Path	Details
Distribution of CA certificates	/CACerts	Section 4.3
Enrollment of new clients	/simpleEnroll	Section 4.4
Re-Enrollment of existing clients	/simpleReEnroll	Section 4.4.1
Full CMC (optional)	/fullCMC	Section 4.5
Server-side Key Generation (optional)	/serverKeyGen	Section 4.6
Request CSR attributes (optional)	/CSRAttrs	Section 4.7

Figure 4

An HTTP base path common for all of an EST server's requests is defined in the form of an path-absolute ([RFC3986], section 3.3). The operation path (Figure 4) is appended to the base path to form the URI used with HTTP GET or POST to perform the desired EST operation.

An example:

With a base path of "/arbitrary/path" and an operation path of "/CACerts", the EST client would combine them to form an absolute path of "/arbitrary/path/CACerts". Thus, to retrieve the CA's certificates, the EST client would use the following HTTP request:

```
GET /arbitrary/path/CACerts HTTP/1.1
```

Likewise, to request a new certificate enrollment in this example scheme, the EST client would use the following request:

```
POST /arbitrary/path/simpleEnroll HTTP/1.1
```

The mechanisms by which the EST server interacts with an HTTPS server

to handle GET and POST operations at these URIs is outside the scope of this document. The use of distinct operation paths simplifies implementation for servers that do not perform client authentication when distributing "CACerts" responses.

EST clients are to be provided with the URL of the EST server and the base path. The means by which clients acquire the URL and base path are outside the scope of this document. Whether the URL and base path are provided securely determines the authorization scheme required to perform operations. (See Section 4.1.)

An EST server MAY provide additional, non-EST services on other URIs.

An EST server MAY use multiple base paths in order to provide service for multiple CAs. Each CA would use a distinct base path, but operations are otherwise the same as specified for an EST server operating on behalf of only one CA.

3.2.3. HTTP-Based Client Authentication

An EST server MAY fallback to using HTTP-based client authentication if TLS client authentication (Section 3.3.1.2) is not possible.

Basic and Digest authentication MUST only be performed over TLS 1.1 [RFC4346] (or later). As specified in CMC: Transport Protocols [RFC5273] the server "MUST NOT assume client support for any type of HTTP authentication such as cookies, Basic authentication, or Digest authentication". Clients intended for deployments where password authentication is advantageous SHOULD support the Basic and Digest authentication mechanism. Servers MAY provide configuration mechanisms for administrators to enable Basic [RFC2616] and Digest [RFC2617] authentication methods.

Servers that support Basic and Digest authentication methods MAY reject requests using the HTTP defined WWW-Authenticate response-header ([RFC2616], Section 14.47). At that point the client SHOULD repeat the request, including the appropriate Authorization Request Header ([RFC2617], Section 3.2.2) if the client is capable of using the Basic or Digest authentication. If the client is not capable then the client MUST terminate the connection.

Clients MAY set the username to the empty string ("") if they wish to present a "one-time password" or "PIN" that is not associated with a username.

Support for HTTP-based client authentication has security ramifications as discussed in Section 6. The client MUST NOT respond to this request unless the client has authenticated the EST server

(as per Section 4.1).

3.2.4. Message types

This document uses existing media types for the messages as specified by Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP [RFC2585] and The application/pkcs10 Media Type [RFC5967] and CMC [RFC5272]. To support distribution of multiple application/pkcs7-mime's for the CA certificate chain the [RFC2046] multipart/mixed media type is used.

The message type is specified in the HTTP Content-Type header. The use herein is consistent with [RFC5273], with clarifications made concerning transfer encoding.

For reference the messages and their corresponding MIME and media types are:

Message type	Request type Response type Source(s) of types	Request section Response section
CA certificate request	N/A application/pkcs7-mime RFC 5751	Section 4.3 Section 4.3.1
Cert enroll/renew	application/pkcs10 application/pkcs7-mime RFC 5967, RFC 5751	Section 4.4/4.4.1 Section 4.4.2
Full CMC	application/pkcs7-mime application/pkcs7-mime RFC 5751	Section 4.5.1 Section 4.5.2
Server-side Key Generation	application/pkcs10 multipart/mixed (application/pkcs7-mime & application/pkcs8) RFC 5967, RFC 5751	Section 4.6.1 Section 4.6.2
Request CSR attributes	N/A application/csrattrs (Specified in this RFC)	Section 4.7.1 Section 4.7.2

Figure 5

3.3. TLS Layer Design

TLS provides communications security for the layers above it. Specifically, the integrity and authentication services it provides are leveraged to supply proof-of-identity and to allow authorization decisions to be made. TLS client authentication is the preferred method for identifying EST clients. In lieu of that, HTTP authentication protected by TLS encryption is also acceptable. Additionally, TLS channel binding information may be optionally inserted into a certificate request in order to provide the EST server with assurance that the authenticated TLS client entity has possession of the private key for the certificate being requested.

HTTP 1.1 [RFC2616] and above support persistent connections. As given in Section 8.1 of that RFC persistent connections may be used to reduce network and processing load associated with multiple HTTP requests. EST does not require persistent HTTP connections and their use is out of scope of this specification.

3.3.1. TLS for transport security

HTTPS is defined in HTTP Over TLS [RFC2818] and is a specification of how HTTP messages are carried over TLS. HTTPS (e.g., HTTP over TLS) MUST be used. TLS 'session resumption' SHOULD be supported.

3.3.1.1. TLS-Based Server Authentication

The EST client MUST authenticate the EST server by validating the TLS server certificate the server presented during the TLS 1.1 [RFC4346] (or later) exchange-defined Server Certificate message or the client MUST independently validate the response contents. Validation is performed as given in [RFC5280] and [RFC6125].

There are multiple methods of validation depending on the current state of the client:

Method 1) If the client has a store of trust anchors, which may be in the form of certificates, for authenticating TLS connections the client MAY validate the TLS server certificate using the standard HTTPS logic of checking the server's identity as presented in the server's Certificate message against the URI provisioned for the EST server (see HTTP Over TLS [RFC2818], Section 3.1 "Server Identity" and [RFC6125]). This method makes it possible for clients with a store of trust anchors to securely obtain the CA certificate by leveraging the HTTPS security model. The EST server URI SHOULD be made available to the client in a secure fashion so that the client only obtains EST functions from a desired server.

Method 2) If the client already has one or more trust anchors associated with this EST server, the client MUST validate the EST server certificate using these trust anchors. The EST server URI MAY be made available to the client in an insecure fashion. The EST server certificate MUST contain the id-kp-cmcRA [CMC RFC5272bis] extended key usage extension.

Method 3) If the client does not yet have a trust anchor associated with this EST server then the client MAY provisionally accept the TLS connection, but the HTTP content data MUST be accepted manually as described in Section 4.3. HTTP authentication requests MUST NOT be responded to since the server is unauthenticated (only the content data is accepted manually).

Methods 1 and 2 are essentially validation as given in [RFC5280]. Method 1 is as described in [RFC6125], Section 6.6.1 "Match Found". Method 2 is described in [RFC6125] as "No Match Found, Pinned Certificate". Method 3 is described in [RFC6125], Section 6.6.4 as "Fallback" and describes the process of "pinning" the received certificate.

If one of these validation methods succeeds, the CA certificate(s) are stored and "pinned" for future use. If none of these validation methods succeeds the client MUST reject the EST server response and SHOULD log and/or inform the end user.

If Method 1 was used to authenticate the EST server then subsequent connections to the EST server also use Method 1. If Method 2 was used to authenticate the EST server then subsequent connections to the EST server also use Method 2. If Method 3 was used to manually authenticate the EST server then the EST client SHOULD "pin" the CA certificates received from a /CACerts (Section 4.3) operation and Method 2 is used for subsequent connections.

3.3.1.2. TLS-Based Client Authentication

Clients SHOULD support [RFC4346]-defined (or later) Certificate request (section 7.4.4). As required by [RFC4346], the client certificate needs to indicate support for digital signatures. The client SHOULD support this method in order to leverage /simpleReEnroll using client authentication by existing certificate. If a client does not support TLS client authentication, then it MUST support HTTP-based client authentication. (Section 3.2.3)

3.4. Proof-of-Possession

As defined in Section 2.1 of CMC [RFC5272], Proof-of-possession (POP) "refers to a value that can be used to prove that the private key

corresponding to the public key is in the possession and can be used by an end-entity."

The signed enrollment request provides a "Signature"-based proof-of-possession. The mechanism described in Section 3.5 strengthens this by optionally including "Direct"-based proof-of-possession by including TLS session specific information within the data covered by the enrollment request signature (thus linking the enrollment request to the authenticated end-point of the TLS connection).

3.5. Linking Identity and POP information

This specification provides an optional method of linking identity and proof-of-possession by including information specific to the current authenticated TLS session within the signed certification request. This proves to the server that the authenticated TLS client has possession of the private key associated with the certification request and that the client was able to sign the certification request after the TLS session was established. This is an alternative to the [RFC5272] Section 6.3-defined "Linking Identity and POP information" method available if full CMC messages are used.

The client generating the request SHOULD obtain the tls-unique value as defined in Channel Bindings for TLS [RFC5929] from the TLS subsystem. The tls-unique value is encoded as specified in Section 4 of Base64 [RFC4648] and the resulting string is placed in the certification request challenge-password field. If tls-unique information is not embedded within the certification request the challenge-password field MUST be empty.

The tls-unique specification includes a synchronization issue as described in Channel Bindings for TLS [RFC5929] section 3.1. This problem is avoided for EST implementations. If the tls-unique value is used it MUST be from the first TLS handshake. EST client and servers use their tls-unique implementation specific synchronization methods to obtain this first tls-unique value.

If identity linking is used then TLS renegotiation MUST use "secure_renegotiation" [RFC5746] (thus maintaining the binding). Mandating secure renegotiation secures this method of avoiding the synchronization issues encountered when using the most recent tls-unique value (which is defined as the the value from the most recent TLS handshake).

The EST server MUST verify the tls-unique information embedded within the certification request and MUST reject requests with invalid tls-unique information. The EST server MAY be configured to accept requests from authenticated clients that do not include the tls-

unique information.

The tls-unique value is encoded into the certification request by the client but back-end infrastructure elements that process the request after the EST server might not have access to the initial TLS session. Such infrastructure elements validate the source of the certification request to determine if POP checks have already been performed. For example if the EST client authentication results in an authenticated client identity of an EST server RA that is known to independently verify the proof-of-possession then the back-end infrastructure does not need to perform proof-of-possession checks a second time. If the EST server forwards a request to a back-end process it SHOULD communicate the authentication results. This communication might use the CMC "RA POP Witness Control" in a CMC Full PKI Request message or other mechanisms which are out-of-scope of this document.

[[EDNOTE: A specific error code (TBD) is returned indicating this additional linkage might be useful. This would be similar to the "WWW-Authenticate response-header" control message. Alternatively simply rejecting the request with an informative text message would work in many use cases.]]

4. Protocol Exchange Details

Before processing a request, an EST server determines if the client is authorized to receive the requested services. Likewise, the client must make a determination if it will accept services from the EST server. Those determinations are described in the next two sections. Assuming that both sides of the exchange are authorized, then the actual operations are as described in the sections following.

4.1. Server Authorization

The client MUST check the EST server authorization before accepting the server's response. The presented certificate MUST be an end-entity certificate such as a CMC Registration Authority (RA) certificate.

There are multiple methods for checking authorization corresponding to the method of server authentication used (these authorization methods align with the authentication methods described in Section 3.3.1.1):

Method 1) If the client authenticated the EST server using the client's TLS trust anchors store, then the client MUST have obtained the EST server's URI in a secure fashion. The client MUST check the URI "against the server's identity as presented in the server's Certificate message" (Section 3.1 "Server Identity" [RFC2818] and [RFC6125]). The securely configured URI provides the authorization statement and the server's authenticated identity confirms it is the authorized server.

Method 2) If the previous check fails or is not applicable, or if the EST server's URI was made available to the client in an insecure fashion, then the EST server certificate MUST contain the id-kp-cmcRA [CMC RFC5272bis] extended key usage extension. The client MUST further verify the server's authorization by checking that the [RFC5280]-defined certificate policy extension sequence contains the 'RA Authorization' policy OID. The RA Authorization policy OID is defined as: id-cmc [[EDNOTE: TBD, perhaps 35]]. The RA Authorization policy information MUST NOT contain any optional qualifiers.

Method 3) If fallback logic was invoked to accept the certificate manually, then that authentication implies authorization of the EST server.

4.2. Client Authorization

When the EST server receives a CMC Simple PKI Request or rekey/renew message, the decision to issue a certificates is always a matter of local policy. Thus the CA can use any data it wishes in making that determination. The EST protocol exchange provides the EST server access to the TLS client certificate in addition to any HTTP user authentication credentials to help in that determination. The communication channel between the TLS server implementation and the EST software implementation is out-of-scope of this document.

If the client authentication is incomplete (for example if the client certificate is self-signed or issued by an unknown PKI or if the client offered an unknown username/password during HTTP authentication) the server MUST extract the certificate request for manual authorization by the administrator.

4.3. Distribution of CA certificates

The EST Client MAY request trust anchor information of the CA (in the form of certificates) by sending an HTTPS GET message to the EST server with an operations path of "/CACerts". Clients SHOULD request an up-to-date response before stored information has expired in order to maintain continuity of trust.

The EST server SHOULD NOT require client authentication or authorization to reply to this request.

The client MUST authenticate the EST server as specified in Section 3.3.1 and check the server's authorization as given in Section 4.1. If the TLS authentication and authorization is not successful then the client MAY continue the TLS handshake to completion and proceed with the /CACerts request. If the EST client continues with an unauthenticated connection the EST client MUST extract the CA certificate from the response (Section 4.3.1) and engage the end-user to authorize the CA certificate using out-of-band pre-configuration data such as a CA certificate "fingerprint" (e.g., a SHA-1, SHA-256, SHA-512, or MD5 hash on the whole CA certificate). In this case it is incumbent on the end user to properly verify the fingerprint or to provide valid out-of-band data necessary to verify the fingerprint.

4.3.1. Distribution of CA certificates response

The EST server MUST respond to the client HTTPS GET message with CA trust anchor information, in the form of certificates within the CMC Simple PKI Response. The response is conveyed within an HTTP response.

The EST server MUST include the current CA certificate in the response. The EST server MUST include any additional certificates the client would need to build a chain to the root certificate. For example if the EST server is configured to use a subordinate CA when signing new client requests then the appropriate subordinate CA certificates to chain to the root must be included in the response.

Additional certificates MAY be included. If support for the CMP root certificate update mechanism is provided by the CA then the server MUST include the three "Root CA Key Update" certificates OldWithOld, OldWithNew, and NewWithOld. These are defined in Section 4.4 of CMP [RFC4210].

The client can always find the current self-signed CA certificate by examining the certificates received. The NewWithNew certificate is self-signed and has the latest NotAfter date.

The NewWithNew certificate is the certificate that is extracted and authorized using out-of-band information as described in Section 4.3. When out-of-band validation occurs each of the other three certificates MUST be validated using normal [RFC5280] certificate path validation (using the NewWithNew certificate as the trust anchor) before they can be used to build certificate paths during peer certificate validation.

The response format is the CMC Simple PKI Response as defined in [RFC5272]. The HTTP content-type of "application/pkcs7-mime" MUST be specified. The CMC Simple PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFBMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1tdGF0ZTEh
Simplified example of Base64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryble4iZgMUvuIgwEjQwpD
8J40hHvLhlo=
-----END PKCS7-----
```

4.4. Simple Enrollment of Clients

The EST client MAY request a certificate from the EST server by HTTPS POSTing using the operation path value of "/simpleEnroll".

When HTTPS POSTing to the 'SimpleEnroll' location the client MUST include a CMC Simple PKI Request as specified in CMC Section 3.1 (i.e., a PKCS#10 Certification Request). Consistent with [RFC6403] the certification request "signature MUST be generated using the private key corresponding to the public key in the CertificationRequestInfo, for both signature and key establishment certification requests". The signature provides proof-of-possession of the private key to the EST server.

The HTTP content-type of "application/pkcs10" MUST be specified. The format of the request is as specified in Section 6.4 of [RFC4945].

The server MUST check client authorization as specified in Section 4.2. The EST server MUST check the tls-unique value as described in Section 3.5 but depending on policy MAY accept a request without the encoded tls-unique value. The EST server applies whatever authorization or policy logic it chooses in determining if the certificate should be issued.

The optional client signature certificate MAY be an existing certificate issued by the CA the EST server is providing services for or it MAY be from any other PKI the EST server indicated as acceptable during the TLS handshake.

The client MAY request an additional certificate even when using an existing certificate in the TLS client authentication. For example the client can use an existing signature certificate to request a key encryption certificate.

The client MUST authenticate the EST server as specified in Section 3.3.1.1.

4.4.1. Simple Re-Enrollment of Clients

The EST client MAY request renew/rekey of its certificate from the EST server by HTTPS POSTing using the operation path value of `"/simpleReEnroll"`.

The certificate request is the same format as for the `"simpleEnroll"` path extension with the same HTTP content-type.

The server MUST check client authorization as specified in Section 4.2. The EST server MUST check the `tls-unique` value as described in Section 3.5 but depending on policy MAY accept a request without the encoded `tls-unique` value. The server applies whatever authorization or policy logic it chooses in determining if the certificate should be renewed/rekeyed. The optional client signature certificate MAY be an existing certificate issued by the CA the EST server is providing services for or it MAY be from any other PKI the EST server indicated as acceptable during the TLS handshake. When attempting to renew or rekey the client SHOULD use an existing certificate for TLS client authentication (Section 3.3.1.2). The certificate being re-enrolled MAY be different than the certificate used for EST client authentication.

The EST server MUST handle enrollment requests submitted to the `"simpleReEnroll"` URI as a renewal or rekey request. (This explicit method of indicating a re-enroll request is an alternative to the `/fullCMC` method specified in Section 2 of [RFC5272] wherein the `"renewal and rekey requests look the same as any certification request, except that the identity proof is supplied by existing certificates from a trusted CA"`).

The request `Subject/SubjectAltName` field(s) MUST contain the identity of the certificate being re-enrolled. The `ChangeSubjectName` attribute, as defined in [RFC6402] MAY be included in the certificate request. The EST server MUST verify that that authenticated client is authorized to perform the inferred re-enroll operation.

If the public key information in the certification request is the same as the currently issued certificate the EST server performs a renew operation. If the public key information is different than the currently issued certificate then the EST server performs a rekey operation. The specifics of these operations are out of scope of this profile.

The client MUST authenticate the EST server as specified in Section 3.3.1.1. The EST client is RECOMMENDED to have obtained the current CA certificates using Section 4.3 to ensure it can validate the EST server certificate.

4.4.2. Simple Enroll and Re-Enroll Response

If the enrollment is successful the server response MUST have an HTTP 200 response code with a content-type of "application/pkcs7-mime". The response data is a degenerate certs- only CMC Simple PKI Response containing only the certificate issued. The CMC Simple PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryble4iZgMUvuIgwEjQwpD
8J40hHvLhlo=
-----END PKCS7-----
```

When rejecting a request the server MUST specify either an HTTP 4xx/ 401 error, or an HTTP 5xx error. A CMC PKI Response with an HTTP content-type of "application/pkcs7-mime" MAY be included in the response data for any error response. If the content-type is not set the response data MUST be a plain text human-readable error message. A client MAY elect not to parse a CMC error response in favor of a generic error message.

If the server responds with an HTTP 202 this indicates that the request has been accepted for processing but that a response is not yet available. The server MUST include a Retry-After header as defined for HTTP 503 responses and MAY include informative human-readable content. The client MUST wait at least the specified 'retry-after' time before repeating the same request. The client repeats the initial enrollment request after the appropriate 'retry-after' interval has expired. The client SHOULD log or inform the end user of this event. The server is responsible for maintaining all state necessary to recognize and handle retry operations as the client is stateless in this regard (it simply sends the same request repeatedly until it receives a different response code).

All other return codes are handled as specified in HTTP.

If the EST client has not obtained the current CA certificates using Section 4.3 then it may not be able to validate the certificate received.

4.5. Full CMC

The EST client MAY request a certificate from the EST server by HTTPS POSTing using the operation path value of "/fullCMC".

The client MUST authenticate the server as specified in Server

Authentication (Section 3.3.1.1), if method 3 is used, then the Publish Trust Anchors control within the HTTP content must be accepted manually as noted in Section 4.3. While use of TLS is not optional within EST, since a full CMC message inately provides security, a TLS NULL cipher suite may be used while making this request.

The server SHOULD authenticate the client as specified in Section 3.3.1. The server MAY depend on CMC client authentication methods instead.

4.5.1. Full CMC Request

When HTTPS POSTing to the "fullCMC" location the client MUST include a valid CMC message. The HTTP content-type MUST be set to "application/pkcs7-mime" as specified in [RFC5273].

4.5.2. Full CMC Response

The server responds with the client's newly issued certificate or provides an error response.

If the enrollment is successful the server response MUST have an HTTP 200 response code with a content-type of "application/pkcs7-mime" as specified in [RFC5273]. The response data includes either the CMC Simple PKI Response or the CMC Full PKI Response.

When rejecting a request the server MAY specify either an HTTP 4xx/401 error or an HTTP 5xx error. A CMC response with content-type of "application/pkcs7-mime" MUST be included in the response data for any error response. The client MUST parse the CMC response to determine the current status.

All other return codes are handled as specified in Section 4.4.2 or HTTP [RFC2616].

The CMC PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFBMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of CMC Full PKI Response
ED8rf3UDF6HjloiV3jBnpetx4UjZH/BlmD9HMqofVEryble4iZgMUvuIgwEjQwpD
8J40hHvLh1o=
-----END PKCS7-----
```

4.6. Server-side Key Generation

[[EDNOTE: This section is references [draft-ietf-pkix-cmc-serverkeygeneration-00] which has not yet been published.]]

The EST client MAY request a "private" key and associated certificate from the EST server by HTTPS POSTING using the operation path value of `"/serverKeyGen"`.

The client MUST authenticate the server as specified in Section 3.3.1.1. The EST client is RECOMMENDED to have obtained the current CA certificates using Section 4.3 to ensure it can validate the EST server certificate.

The EST server MUST authenticate the client as specified in Section 3.3.1. The EST server applies whatever authorization or policy logic it chooses to determine if the "private" key and certificate should be distributed. The server SHOULD use TLS-Based Client Authentication for authorization purposes. The server SHOULD respond to repeated requests from the same client with the same "private" key and certificate but MAY respond with a renewed or rekeyed "private" key and certificate. Clients that wish multiple "private" keys and certificates MUST specify a keyUsage in the certificate request which the server will use to intuit the type of key to be generated.

Proper random number and key generation and storage is a server implementation responsibility. The keypair and certificate are transferred over the TLS session; the EST server MUST verify that the current ciphersuite is acceptable for securing the key data.

4.6.1. Server-side Key Generation Request

The certificate request is HTTPS POSTed and is the same format as for the `"/simpleEnroll"` path extension with the same content-type.

The public key values of the certificate request and the request signature MUST be ignored by the server.

4.6.2. Server-side Key Generation Response

If the request is successful the server response MUST have an HTTP 200 response code with a content-type of `"multipart/mixed"` consisting of two parts. The first part is the "private" key data and the second part is the certificate data.

The first submessage is an `"application/pkcs8"` consisting of the

Base64 encoded DER-encoded PrivatekeyInfo sandwiched between the PEM headers as described in [RFC5958]:

```
-----BEGIN PRIVATE KEY-----
MIIBhDCB7gIBADBQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of DER-encoded PrivateKeyInfo
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J40hHvLh1o=
-----END PRIVATE KEY-----
```

The second submessage is an "application/pkcs7-mime" and exactly matches the certificate response to /simpleEnroll. The server response MUST use the same SubjectPublicKeyInfo as requested or the request MUST be denied.

When rejecting a request the server MUST specify either an HTTP 4xx/401 error, or an HTTP 5xx error. If the content-type is not set the response data MUST be a plain text human-readable error message.

4.7. CSR Attributes

The CA MAY want to include client-provided attributes in certificates that it issues and some of these attributes may describe information that is not available to the CA. For this reason, the EST client MAY request a set of attributes from the EST server to include in its certification request.

4.7.1. CSR Attributes Request

The EST Client MAY request a list of CA-desired CSR attributes from the CA by sending an HTTPS GET message to the EST server with an operations path of "/CSRAttrs". Clients SHOULD request such a list if they have no a priori knowledge of what attributes are desired by the CA in an enrollment request or when dictated by policy.

4.7.2. CSR Attributes Response

The server MUST reply to the client's HTTPS GET message with a (set of) attribute(s). Responses to attribute request messages MUST be encoded as content type "application/csrattrs" and conveyed within an HTTP response.

The syntax for application/csrattrs body is as follows:

Csrattrs ::= SEQUENCE SIZE (0..MAX) OF OBJECT IDENTIFIER { }

A robust application SHOULD output Distinguished Encoding Rules (DER)

([X.690]) but MAY use Basic Encoding Rules (BER) ([X.680]). Data produced by DER or BER is 8-bit. When the transport for the application/csrattrs is limited to 7-bit data, a suitable transfer encoding MUST be applied in MIME-compatible transports. The base64 encoding (section 4 of [RFC4648]) SHOULD be used with application/csrattrs, although any 7-bit transfer encoding may work.

Servers include zero or more object identifiers that they wish the client to include in their certification request. When the server encodes csrattrs as an empty SEQUENCE OF it means that the server has no attributes it wants in client certification requests.

For example, if a CA wishes to have a certification request contain the MAC address [RFC2397] of a device and the pseudonym [X.520] and friendly name [RFC2925] of the holder of the private analog to the public key in the certification request, it takes the following object identifiers:

- o macAddress: 1.3.6.1.1.1.1.22
- o pseudonym: 2.5.4.65
- o friendlyName: 1.2.840.113549.1.9.20

and encodes them into an ASN.1 SEQUENCE to produce:

```
30 19 06 07 2b 06 01 01 01 01 16 06 03 55 04 41 06 09 2a 86 48 86
f7 0d 01 09 14
```

and then base64 encodes the resulting ASN.1 SEQUENCE to produce:

```
MBkGBysGAQEBARYGAlUEQQYJKoZIhvcNAQkU
```

The resulting response would look like this:

```
Content-Type: application/csrattrs; name=attributes
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=attributes
```

```
MBkGBysGAQEBARYGAlUEQQYJKoZIhvcNAQkU
```

5. IANA Considerations

(This section is incomplete)

The following aspects should be registered with IANA Considerations:

The RA Authorization certificate policy extension OID as discussed in Section 4.1 requires registration with IANA.

[[EDNOTE: The URLs specified in Section 1 probably do not need to be registered with IANA.]]

IANA SHALL update the Application Media Types registry with the following filled-in template from [RFC4288].

The media subtype for Attributes in a CertificationRequest is application/csrattrs.

Type name: application

Subtype name: csrattrs

Required parameters: None

Optional parameters: None

Encoding considerations: binary;

Security Considerations:

Clients request a list of attributes that servers wish to be in certification requests. The request/response SHOULD be done in a TLS-protected tunnel.

Interoperability considerations: None

Published specification: This memo.

Applications which use this media type:

Enrollment over Secure Transport (EST)

Additional information:

Magic number(s): None

File extension: None

Macintosh File Type Code(s):

Person & email address to contact for further information:

Dan Harkins <dharkins@arubanetworks.com>

Restrictions on usage: None

Author: Dan Harkins <dharkins@arubanetworks.com>

Intended usage: COMMON

Change controller: The IESG

6. Security Considerations

(This section is incomplete)

"Badges? We ain't got no badges. We don't need no badges! I don't have to show you any stinkin' badges!" -- The Treasure of the Sierra Madre.

As described in CMC Section 6.7, "For keys that can be used as signature keys, signing the certification request with the private key serves as a POP on that key pair". The inclusion of tls-unique within the certification request links the the proof-of-possession to the TLS proof-of-identity.

As given in Section 3.3.1.2 clients use an existing certificate for TLS client authentication. If a certificate with appropriate key usage is not available the client MAY generate one. If a self-signed certificate with appropriate key usage is used the server SHOULD require HTTP-based client authentication according to server policy as described in Section 3.3.1.2 and Section 4.2. The server MAY fallback on manual authorization by the server administrator.

Clients authenticate EST servers by means of TLS authentication. If a client does not possess a root certificate suitable for validating an EST server certificate, it MAY rely upon other trusted root certificates it has (such as those found in its HTTPS store). The client then is able to retrieve additional root certificates as given in Section 4.3. Alternatively, a server certificate MAY be authenticated manually as specified in Section 3.3.1.1 #3.

As noted in Section 3.3.1.1 servers use an existing certificate for TLS server authentication. When the server certificate is issued by a mutually trusted PKI hierarchy, validation proceeds as specified in Section 4.1. In this situation the client has validated the server as being a valid responder for the URI configured but can not directly verify that the responder is authorized as an RA within the to-be-enrolled PKI hierarchy. A client may thus be enticed to expose username/password or certificate enrollment requests to an unauthorized server (if the server presents a valid HTTPS certificate

for an erroneous URL that the client has been tricked into using). Proof-of-identity and Proof-of-possession checks by the CA prevent an illegitimate RA from leveraging such misconfigured clients to act as a man-in-the-middle during client authenticated operations but it is possible for such illegitimate RAs to send the client doctored messages or erroneous CA certificate lists. If the illegitimate RA has successfully phished a username/password or PIN from the client it might try to use these values to enroll its own keypair with the real PKI hierarchy. EST servers identified with an externally issued server certificate SHOULD require HTTPS-based client authentication (Section 3.3.1.2). Similarly EST clients SHOULD use an existing client certificate to identify themselves and otherwise prevent "private data" (obviously including passwords but also including private identity information) from being exposed during the enrollment exchange a weak server authorization method is used.

Section 3.2.3 allows clients to optionally authenticate using HTTP-based authentication in place of TLS-based authentication. HTTP-based authentication MUST NOT take place unless performed over a TLS-protected link.

The server-side key generation method allows keys to be transported over the TLS connection to the client. The distribution of "private" key material is inherently risky and servers are NOT RECOMMENDED to support this operation by default. Clients are NOT RECOMMENDED to request this service unless there is a compelling operational benefit such as the use of [BGPsec RPKI].

Regarding the CSR attributes that the CA may list for inclusion in an enrollment request, there are no real inherent security issues with the content being conveyed but an adversary who is able to interpose herself into the conversation could exclude attributes that a server may want, include attributes that a server may not want, and render meaningless other attributes that a server may want.

[[EDNOTE: need final reference for BGPsec RPKI]]

Support for Basic authentication as specified in HTTP [RFC2617] allows the server access to the client's cleartext password. This provides integration with legacy username/password databases but requires exposing the plaintext password to the EST server. Use of a PIN or one-time-password can help mitigate concerns but EST clients are RECOMMENDED to use such credentials only once to obtain an appropriate client certificate to be used during future interactions with the EST server.

7. References

7.1. Normative References

- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.
- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC4945] Korver, B., "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX", RFC 4945, August 2007.

- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.
- [RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, June 2008.
- [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.
- [RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, February 2010.
- [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, July 2010.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.
- [RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, August 2010.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, November 2011.
- [X.680] ITU-T Recommendation, "ITU-T Recommendation X.680 Abstract Syntax Notation One (ASN.1): Specification of basic notation", November 2008, <<http://www.itu.int/rec/T-REC-X.680-200811-I/en>>.
- [X.690] ITU-T Recommendation, "ITU-T Recommendation X.690 ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", November 2008, <<http://www.itu.int/rec/T-REC-X.690-200811-I/en>>.

7.2. Informative References

- [IDevID] IEEE Std, "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/>>

standard/802.1AR-2009.html>.

- [RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, August 1998.
- [RFC2925] White, K., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 2925, September 2000.
- [RFC6403] Ziegler, L., Turner, S., and M. Peck, "Suite B Profile of Certificate Management over CMS", RFC 6403, November 2011.
- [X.520] ITU-T Recommendation, "ITU-T Recommendation X.520 The Directory: Selected attribute types", November 2008, <<http://www.itu.int/rec/T-REC-X.520-200811-I/en>>.

Appendix A. Server Discovery

(informative)

Clients can use DNS-SD or similar discovery algorithms to determine the EST server URI. In such cases it is expected that method 2 (Section 3.3.1.1) be used during server authentication because the first method is insecure if the discovery mechanism is insecure.

If the user interaction in the third method is acceptable it is expected that the user would also supply the URI instead of using a discovery protocol.

Appendix B. External TLS concentrator

(informative)

In some deployments it may be beneficial to use a TLS concentrator to offload the TLS processing from the server.

The TLS server should not reject the connection based on PKIX validation of the client certificate. Instead the client certificate is passed to the EST server layer for verification and authorization. This allows support of external TLS concentrators that might provide an independent TLS implementation.

The TLS concentrator does validate the TLS Section 7.4.8 'Certificate Verify'.

In such a deployment the TLS client authentication result must be

forwarded to the EST server layer. For example a TLS concentrator might insert the client certificate into the HTTP header (first removing any existing client certificates, possibly inserted by a nefarious client, from the HTTP headers) before forwarding the HTTP connection to the EST server.

The EST server MUST be specifically configured by the administrator to accept this mechanism.

Appendix C. CGI Server implementation

(informative)

In some deployments it may be beneficial to use a HTTPS server that runs the EST server as a CGI application.

The HTTPS server should not reject the connection based on PKIX validation of the client certificate. Instead the client certificate is passed to the EST server layer for verification and authorization. This allows support of external HTTPS servers that might provide an independent TLS implementation.

In such a deployment the TLS client authentication result must be forwarded to the EST server layer. For example an HTTPS server might insert the client certificate into the environment variables before forwarding the HTTP data to the EST server.

Appendix D. Operational Scenario Example Messages

(informative)

This section expands on the Operational Scenario Overviews by providing detailed examples of the messages at each TLS layer. Figures are informative sections of TLSv1

D.1. Obtaining CA Certificates

The following is an example of a valid /CACerts exchange.

During the initial TLS handshake the client can ignore the optional server generated "certificate request" and can instead proceed with the HTTP GET request:

```

GET /CACerts HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*

```

In response the server provides the current CA certificate:

```

<= Recv header, 38 bytes (0x26)
Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

```

```

<= Recv data, 1111 bytes (0x457)
-----BEGIN PKCS7-----.MIIDEQYJKoZIhvcNAQcCoIIDAjCCAv4CAQExADALBg
kqhkiG9w0BBwGgggGkMIIC.4DCCAcigAwIBAgIJAOjxMzcXhE5wMA0GCSqGSIb3D
QEBBQUAMBCxFTATBgNVBAMT.DGVzdEV4YW1wbGVDQTAEFw0xMjA3MDQxODM5Mjda
Fw0xMzA3MDQxODM5MjdaMBcx.FTATBgNVBAMTDGVzdEV4YW1wbGVDQTCCASIwDQY
JKoZIhvcNAQEBBQADggEPADCC.AQoCggEBALQ7SjZSt6qrnBzUnBNj9z4oxYkvMA
Vh00IOVRkNhZ/2kDGsds0ne7cw.W33kYlxPba4psdLMixCT/O8ZQMpgA+QFKtwb9
VPE8EFUgGzxSYHQHjhJsbg0BVaN.Ya38vjKMjvosuSXUHWkvU57SInSkMr3/aNtS
T8qFfeC6Vuf/G/GLHGuHQKAY/DSO.206MjaMNmWYRVQQVERGookRA4GBF/YE+G/C
SlTsCQNE0KyBFz8JWIkgyY2gYkxb7.wWMvvhaU/Esp+2DG92v9Dhs2MRgrR+WPs7
Y6CYOLD5Mr5lEdkHg27IxxSAoRrI6D.fnVVEQGCj7QrrsUgfXFVYv6cCWFfhMcCA
wEAAAmVMC0wDAYDVR0TBAUwAwEB/zAd.BgNVHQ4EFgQUhH9KxW5TsJkgL7kg2kxJ
yy5tD/MwDQYJKoZIhvcNAQEFBQADggEB.AD+vzdZo292XFb2vXojdKD57Gv4tKvM
hvXRdVINntzkY/0AyFCfHJ4BwndgtMh4t.rvBD8+8dL+W3jFPjCScCUQ/JEnFuMn
b5+kivLeqOnUshETasFPBz2Xq4ClSHDno9.CW0csjPPw08Tn4dSrzdBSq1NdXB2z
9N0paVnbbp0lqQGhXSOaEvcbZcDuGiW7Di3.gV++remokuPph/s6XoZffzc7ZVzf
Job6tS4RwNz0lsutPybXiRWivOz7+QeCOT87.nTGlkQH/+RImUyJ2jefjAW/GDFT
Pzek6cZnabAtsg32n0Pv0j0/1RTNSdYGxPIVA.2f9fhMqMz+vm3w4CFNkGZnOhAD
EA.-----END PKCS7-----.

```

D.2. Previously Installed Signature Certificate

The following is an example of a valid /simpleEnroll exchange. During this exchange the EST client uses an existing certificate issued by a trusted 3rd party PKI to obtain an initial certificate from the EST server.

During the initial TLS handshake the server generated "certificate request" includes both the distinguished name of the CA the EST server provides services for ("estExampleCA") and it includes the distinguished name of a trusted 3rd party CA ("estEXTERNALCA"):

```

0d 00 00 3d 03 01 02 40 00 37 00 1a 30 18 31 16 ...=...@.7..0.1.
30 14 06 03 55 04 03 13 0d 65 73 74 45 58 54 45 0...U....estEXTE
52 4e 41 4c 43 41 00 19 30 17 31 15 30 13 06 03 RNALCA..0.1.0...
55 04 03 13 0c 65 73 74 45 78 61 6d 70 6c 65 43 U....estExampleC
41                                     A

```

Which decodes as:

```

Acceptable client certificate CA names
/CN=estEXTERNALCA
/CN=estExampleCA

```

The EST client provides a certificate issued by "estEXTERNALCA" in the certificate response and the TLS handshake proceeds to completion. The EST server accepts the EST client certificate for authentication and accepts the EST client's POSTed certificate request:

```

POST /simpleEnroll HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/x-est-pkcs10
Content-Length: 952

```

=> Send data, 952 bytes (0x3b8)

```

-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGAlUE
AxMccmVxIGJ5IGNsaWVudCBpbjBkZWlvdjEHN0.ZXAgNjEYMBYGA1UEBRMPUElEold
pZGdldCBTtjo2MIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEAWyYI+
aYezyx+kW0GVUbmMKLf2BUd8BgGyKKIJYxms6SH.Bv5S4ktcpYbEPR9iCmp96vK6a
Ar57ArZtMmi0Y6eLX4c+njJnYhUeTivnfyfMM5d.hNVWyzKbJagm5f+RLTMfp0y0
ykqrfZlhFhcNrRzF6mJeaORTHBehMdu8RXcbmy5R.s+vjnUC4Fe3/oLHtXePyYv1
qq1kk0XDrw/+lx0y4Px5tiyb84iPnQOXjG2tuStM+.iEvfpNanwU0+3GDjl3sjx0
+gTKvblp6Diw9NSaqIAKupcgWsA0JlyYkgPiJnXFKL.vy6rXoOyx3wAbGKLrKCxT
l+RH3oNXf3UCH70ad758QIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBADwpafWU
BsOJ2g2oyHQ7Ksw6MwvimjhB7GhjweCcceTSLInUMk10.4E0TfNgaWcoQengMVZr
IcbOb+sa69BWNb/WYIUlfEtJIV23/g3n/y3JltMNw/q+R.200t0bNAViiJHQHmlF
6dt93tkRrTzXnhV70Ijnff08G7P9HfnXQH4Eiv3zOB6Pak.JoL7QlWQ+w5vHpPo6
WGH5n2iE+Ql76F0HykGegaR402+ae0WlGLHEvcN9wiFQVKh.KUHteU10SEPiJlqf
QW+hciLleX2CwuZY5MqKb4qqyDTs4HSQCBCl8jR2cXsGDuN4.PcMpp+9A1/UPuGD
jhwPt/K3y6aV8zUEh8Ws=-----END CERTIFICATE REQUEST-----.

```

The EST server uses the trusted 3rd party CA issued certificate to perform additional authorization and issues a certificate to the client:

```

<= Recv header, 38 bytes (0x26)
Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

<= Recv data, 1200 bytes (0x4b0)
-----BEGIN PKCS7-----.MIIDUQYJKoZIhvcNAQcCoIIDQjCCAz4CAQExADALBg
kqhkiG9w0BBwGgggMkMIID.IDCCAgigAwIBAgIBBjANBgkqhkiG9w0BAQUFADAXM
RUwEwYDVQQDEwxc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM3WhcNMTMwNzA0
MTgzOTM3WjBBMSUwIwYDVQQDE.ExxyZXEGYnkgY2xpZW50IGluIGRlbW8gc3RlcCA
2MRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjYwggEiMA0GCSqGSib3DQEBAQUAA4
IBDwAwggEKAoIBAQCfjIj5ph7.PLH6RbQZVRswot/YFR3wGAbKSQgljGazpIcG/
lLiSlylhsSlH2IKan3q8rpoCvns.Ctm0yaLRjp4tfhz6eMmdiFR5OK+d/J8wz12E
1XDLmps1qCbl/5EtMx+nTLTKSqt9.nWEWFw2tHMXqY15o5FMcF6Ex27xFdxubL1G
z6+OdQLgV7f+gseld4/Ji/WqqWSTR.cOvD/6XHTLg/Hm2LJvziI+dA5eMba25K0z
6IS9+k0CfBTT7cYOOXeyPHT6BMq9uW.noOLD01JqogAq6lyBawDQmXJiSA+ImdcU
ou/Lqteg7LHfABsYousoLFOX5Efegld./dQIfvRoPvnxAgMBAAGjTTBLMAkGAlUd
EwQCMAAwHQYDVR0OBBYEFJv4oLLeNxNK.OmMQDDuJyNR+zaVPMB8GAlUdIwQYMBa
AFIR/SsVuU7I5IC+5INpMScsubQ/zMA0G.CSqGSib3DQEBBQUAA4IBAQCmdomfdR
9vi4VUYdF+eym7F8qVUG/1jtjfaxmrzKeZ.7LQ1F758RtwG9CDu2GPHNPjjeM+DJ
RQZN999eLs3Qd/DIJCNimaqdDqmkeBFC5hq.LZOxbKhSmhlR7YKjIZuyI299rOaI
W54ULyz8k0zw6Rl/0lMJTsDFGJM+9yDeaARE.n3vtKnUDGHsVU3fYpDENaqUunoU
MZfuEdejfHhU7lVbJIloSJbnRwBFkPr/RQ3/5.FymcrBD9RpAM5MsQIn0BONil/o
JM+LjOJqyZLbBxz6P3w/OiJGYJNfFT8YudLfjZ.LDX8A8FFcReapNELC4QxE4OrA
hn3sQUT207ndIsit4kJoQAxA==.-----END PKCS7-----.

```

D.3. Username/Password Distributed Out-of-Band

The following is an example of a valid /simpleEnroll exchange. During this exchange the EST client uses an out-of-band distributed username/password to authenticate itself to the EST server.

During the intial TLS handshake the client can ignore the optional server generated "certificate request" and can instead proceed with the HTTP POST request:

```

POST /simpleEnroll HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenSSL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/x-est-pkcs10
Content-Length: 952

```

```

=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbjBkZWlviHN0.ZXAgMjEYMBYGA1UEBRMPUElEold
pZGdldCBTTjoyMIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEaz9lXz9
Mowul0x0W5v1k7GKlsNy7mAgmkz/wZDimBDXez.QZCb8lr08iTD3tI0NH2xpkY3b
uqFjdtQTzCmANLyNWtRlsC5GjN/EMlJSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
/zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxOPaqeR.+Ktkv1+9mlvvvdKIs5Hm4Sp
O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+F//0BzkFSqWsmUomyJy
WCfLCuX9grs1CNlLxw0gcMprdTxLxjcl8z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
kzTUIkeHtcp6FVf13D+TwIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBAJH7Etuy
B/oQgQeals08mD2U3lFfQ/uYqjNxxZpZJSzVLGMASv9a.pNzaWdfqPdIs+ZZ+gAQ
QkVcXjdbqY3pAf/EeWk+KnuAUjOIPKu3ZBPVbWbXu/Ie7.FlekQ7TLkFNkHSxHRu
2/bPIByBLRVfWNVXd3wPq+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPpy0OwG/Xf
QrKBB3tzpbjCy+cwOUAJFPOTO+86RUjf9Wh+yoMl82vlg8O.FyEaaA/PMpl3aEcT
BlRZmPx4e7FLwGIhbgE7/6K0nF99xdGd7JYPHasbcWszxD0Z.oPYm+44g0gOnhlj
OWpRiKXcnngSSutRILaw=.-----END CERTIFICATE REQUEST-----.
== Info: upload completely sent off: 952 out of 952 bytes
== Info: HTTP 1.1 or later with persistent connection, pipelining
supported

```

The EST server accepts this request but since a client certificate was not provided for authentication/authorization the EST server responds with the WWW-authenticate header:

```

<= Recv header, 27 bytes (0x1b)
HTTP/1.1 401 Unauthorized
<= Recv header, 75 bytes (0x4b)
WWW-Authenticate: Digest qop="auth", realm="estrealm", nonce="13
41427174"

```

The EST client repeats the request, this time including the requested Authorization header:


```

== Info: SSL connection using AES256-SHA
== Info: Server certificate:
== Info:  subject: CN=127.0.0.1
== Info:  start date: 2012-07-04 18:39:27 GMT
== Info:  expire date: 2013-07-04 18:39:27 GMT
== Info:  common name: 127.0.0.1 (matched)
== Info:  issuer: CN=estExampleCA
== Info:  SSL certificate verify ok.
== Info: Server auth using Digest with user 'estuser'
=> Send header, 416 bytes (0x1a0)
POST /simpleEnroll HTTP/1.1
Authorization: Digest username="estuser", realm="estrealm", nonc
e="1341427174", uri="/simpleEnroll", cnonce="ODc00Tk2", nc=00000
001, qop="auth", response="48a2b671ccb6596adfef039e134b7d5d"
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/x-est-pkcs10
Content-Length: 952

=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbIBkZWlviHN0.ZXAgMjEYMBYGA1UEBRMPUElEOld
pZGdlldCBTTjoyMIIBIjANBgkqhkiG9w0BAQEF.AAOCQAQ8AMIIBCgKCAQEAz9lXz9
Mowu1Ox0W5v1k7GKlsNy7mAgmkz/wZDIImBDXez.QZCb8lrO8iTD3tI0NH2xpkY3b
uqFjdtQTzCmANLyNWtRlsC5GjN/EMlJSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
/zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxOPAqER.+Ktkv1+9mlvvsdKIs5Hm4Sp
O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+F//0BzkFSqWsmUomyJy
WCfLCuX9grs1CNlLxw0gcMprdTxlXjcl8z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
kzTUIkeHtcp6FVF13D+TwIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBAJH7Etuy
B/oQgQeals08mD2U31FfQ/uYqjNxxZpZJSzVLGMASv9a.pNzaWdfqPdIs+ZZ+gAQ
QkVcXjdbqY3pAf/EeWk+KnuAUjoIPKu3ZBPVbWbXu/Ie7.FlekQ7TLkFNkHSxHRu
2/bPIByBLRVfWNVXd3WpQ+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPyOOWG/Xf
QrKBB3tzpbjCy+cwOUAJFPOTO+86RUjf9Wh+yoMl82vlg8O.FyEaaa/PMpl3aEcT
BlRZmPx4e7FLwGIhbgE7/6K0nF99xdGd7JYPHasbcWszxD0Z.oPYm+44g0gOnhlj
OWpRiKXcnngSSutRILaw=.-----END CERTIFICATE REQUEST-----.

```

The ESTserver uses the username/password to perform authentication/ authorization and responds with the issued certificate:

```

<= Recv header, 38 bytes (0x26)
0000: Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

<= Recv data, 1200 bytes (0x4b0)
-----BEGIN PKCS7-----.MIIDUQYJKoZIhvcNAQcCoIIDQjCCAz4CAQExADALBg
kqhkiG9w0BBwGgggMkMIID.IDCCAgigAwIBAgIBAjANBgkqhkiG9w0BAQUFADAXM
RUWewYDVQQDEwxlc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM0WhcNMTMwNzA0
MTgzOTM0WjBBMSUwIwYDVQQD.ExxyZXEgYnkgY2xpZW50IGluIGRlbW8gc3RlcCA
yMRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjIwggEiMA0GCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQDP2VfP0yJc.6U7HRbm/WTsYqWw3LuYCCaTP/BkMiYEND7NBk
JvyWs7yJMpe0jQ0fbGmRjdu6oWN.2lBPMKYA0vIla1HWwLkaM38QzULIKs7/NkyK
DzflFc1M/zvw3+MlbsTNLFv/Mrk7.MomhFtngeBmbg0Nqkx8xwHiOoF0/Gg8Cp5H
4pOS/X72bw++x0oizkebhKk7ZaiUc./DkEJd27nOV5voAKKhtml3ZykcxPpkcLQb
U5/4X//QHOQVKpayZSibInJYJ8sK5f.2CuzUI2UvHDSBwmtlPEvGNzXzPTdmYEK
rSqrn+ZQr+2/1HaTz5a4/dqTNNQiR4e.lynoVUWXcP5PAgMBAAGjTTBLMAkGAlUd
EwQCMAAwHQYDVR0OBBYEFChDQpKEfG9c.e4JaMf8438tb2XOIMB8GAlUdIwQYMBa
AFIR/SsVuU7I5IC+5INpMScsubQ/zMA0G.CSqGSIb3DQEBAQUAA4IBAQAn42mIVG
piaY4yqFD0F8KyUhKsdNnyKeeISQxP//lp.quIieJzdWSc7bhWZNldSzNswCod8B
4eJToQejLSNb8JBDC849z0tcuyHgN6N/p8z.IwI+hAlfXS9q02OECyFes4Jmzc7r
erE5jtOdGsEDBIscw/A+Kv86wv6BKbagMslQ.5lAJyPsL6iBhm7LPFrErJgH2kWN
jDKFH9CcVFjXvgriMrLPFeqQWOpj/2XF+4m+c.f9QP5tSjieHJRlhnYk2tldofE7
iV4pJ07Mmf3yBf753VSUVybqWiMcd0Lm7oghSX.E2GAXrsU1N+Nlodn+gJ2wmXTu
AC2aHt9VPRViov4RRRTvoQAxA==.-----END PKCS7-----.

```

D.4. Re-Enrollment

The following is an example of a valid /simpleReEnroll exchange. During this exchange the EST client authenticates itself using an existing certificate issued by the CA the EST server provides services for.

Initially this exchange is identical to enrollment using an externally issued certificate for client authentication since the server is not yet aware of the client's intention. As in that example the EST server the server generated "certificate request" includes both the distinguished name of the CA the EST server provides services for ("estExampleCA") and it includes the distinguished name of a trusted 3rd party CA ("estEXTERNALCA").

```

0d 00 00 3d 03 01 02 40 00 37 00 1a 30 18 31 16 ...=...@.7..0.1.
30 14 06 03 55 04 03 13 0d 65 73 74 45 58 54 45 0...U....estEXTE
52 4e 41 4c 43 41 00 19 30 17 31 15 30 13 06 03 RNALCA..0.1.0...
55 04 03 13 0c 65 73 74 45 78 61 6d 70 6c 65 43 U....estExampleC
41                                         A

```

In text format this is:

```

Acceptable client certificate CA names
/CN=estEXTERNALCA
/CN=estExampleCA

```

The EST client provides a certificate issued by "estExampleCA" in the certificate response and the TLS handshake proceeds to completion. The EST server accepts the EST client certificate for authentication and accepts the EST client's POSTed certificate request.

The rest of the protocol traffic is effectively identical to a normal enrollment.

D.5. Server Key Generation

The following is an example of a valid /serverKeyGen exchange. During this exchange the EST client authenticates itself using an existing certificate issued by the CA the EST server provides services for.

The initial TLS handshake is identical to the enrollment example handshake. The HTTP POSTed message is:

```

POST /serverKeyGen HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/x-est-pkcs10
Content-Length: 968

```

```

=> Send data, 968 bytes (0x3c8)
-----BEGIN CERTIFICATE REQUEST-----.MIICKzCCAXsCAQAwTjEyMDAGAlUE
AxMpc2VydmVyS2V5R2VuIHJlcSBieSBjbGll.bnQgaW4gZGVtbyBzdGVwIDUxGDA
WBgnVBAUTDlBJRDpXaWRnZXQgU046NTCCASIw.DQYJKoZIhvcNAQEBBQADggEPAD
CCAQoCggEBAMnlUlq0ag/fDAVhLgrXEAD6WtZw.Y2rVGev5saWirer2n0OzghB59
uJByxPo0DYBYqZRuoRF0FTL1ZZTmaZxivge0ecA.ZcoR46jwSBocEMT1jkwFyAER
t9Q2EwdnJLIPo/Ib2PLJNb4Jo8NNKmxgtg55BgIVi.vkIB+rMtLeYRUVLORUaBAqX
FmtXRDceVFIEY24iUQw6vESGJKpArht592aT8lyaP.24bZovuG19dd5xtTX3j37K
x49SlkUvLSpD6ZavIFAZn7Yv19LBKHvRIemybUo294.QeLb/VYP10+EathV/igiX
1DHq1UZCZp5SdyUXUwZPatFboNwEVR0R3MJwVECAwEA.AaAAMA0GCSqGSib3DQEB
BQUAA4IBAQAqhHezK5/tvbXleHO/aTBVY091414NM+WA.wJcnS2UaJYScPBq1YK/
giJ+dqAtFE+5ukAj56t7HnooI4EFo9r8jqCHewx7iLZYh.JDxo4hW0sAvHV+Iziy
jkhJNdHBIqGM7Gd5f/2VJLEPQPmwnOL5P+204eQC/QeEYc.bAmfhOS8b/ZH09/9T
PeaeQpjspjOui/100OuLE8KvU3FM0sXMYt1Va0A0jxz1+5k.EiEJo+ltXsQwdP0H
csoTNBN+j3K18omJQS0e91X8v0xkMWYhUtonXD0YZ6SO/B9c.AE6GTADHA/xpSvA
cqlWa+FHxjwEMXdmViHvMUywo31fDZ/TUvCPX.-----END CERTIFICATE REQUE
ST-----.

```

After processing the request the EST server response is:

```

<= Recv header, 17 bytes (0x11)
HTTP/1.1 200 OK
<= Recv header, 16 bytes (0x10)
Status: 200 OK
<= Recv header, 67 bytes (0x43)
Content-Type: multipart/mixed ; boundary=estServerExampleBoundar
y
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

```

```

<= Recv data, 3234 bytes (0xca2)
This is the preamble. It is to be ignored, though it is a handy
place for estServer to include an explanatory note including con
tact or support information.--estServerExampleBoundary.Content-
Type=application/pkcs8.-----BEGIN PRIVATE KEY-----.MIIEvQIBADAN
BgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQC0781l7tri0yii.Mb9ZZYch8ze
izXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM.tzxn41+9tI
sVDkAe4FyzN0hLd/zawggj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blv.aMUKDSJhV
bQ+z/GlW1TRx3iWi5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx.Igbx9vG0
mftIIxM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhg1FU8.DQiQEki
nn66GPMtmlSNgitxFxWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhk.g0gMIQ

```

TXAgMBAECgEgEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu.Y/0sc
VbLbu46WzysolIfJFyUC+zFJnbMCCOPjGbI/4NWkEqc9TA1Kz+wDo+hf5bf0.ypFr
EmikHk8R3fkpnvKi69ldw0iYnqcFVhq7VtGrSmJcy6Hckwbk7EBoUZGL0wtp.xl0
6XlhksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTHlH5M4daFbqyRWD7UiPKt.Q3
hdwlrpyVZg8ZbBFp0Ej4f9GdRaq88SIKMKCDu3t9ibn/vlkEte+PxhuwyW+d0o.h
kKSEW0yLKCzQm5tjSPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv.
vOJnNWP7oMzyJ4/eFqdE9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G
.gFd93diyRPSvBNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NEEnZOe
c.jPMOV/IcGlaLzTqQssL3zo/0gQKBgQDB40lpg3EBggtJ/+dlkLHUW8c7Pe3UyL
kS.VxVsyQwioYt8xMeCWuPvPNFcojCw53KN/YSpCVjpttKGsPtLibMlKYKgasEgg
cvl.Vb5OfTA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0gO38HDWq7EtqSi9Q0JSM
g3YC.QNcOORptVwKBgQCHrCafaYWDhA1l/+g2U9x6Yd56iff43rCbnV+2EQCVaq
i49xC.w4AH+Bs0mdlgT5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQH
ek92D7.wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJdQYckg
QKBgFXS.zSBzGgwz1/F7AaDZK49mlwPnhyeBb00qHwBx/Li7lrZlmWef+nSF9Juh
/Y77B5/J.UPd09vgGgS00nRk0LIRP2s5OU5IQgQTVLvf8a1UmbVgI+KX511Yi5yM
ztEwRcjEX.VM9ejXeXN0I57pvqG/xCOK3Kl2eYLh4TO9/E8WjjAoGAA1mqUV4Hnf
4yvFlrydMp.fpvWekiIRE33iEbYZNATYhsl7uxwn760pqVifkq2DSrZeYm4+lw9
jwWMtUoPzpg.CJYMoG1846nhiZrbbJ5b5twoLV6GRmkK/CfOxPXNzCtSoQA86HHq
7rRdhXSau/bY.EXc91tnhLjFzZxdBgrrd+f4k=.-----END PRIVATE KEY-----
--estServerExampleBoundary.Content-Type: application/pkcs7-mime.
.-----BEGIN PKCS7-----.MIIDPAYJKoZIhvcNAQcCoIIDLTCCAykCAQExADALB
gkqhkiG9w0BBWggGMPMIID.CzCCAFogAwIBAgIBBTANBgkqhkiG9w0BAQUFADAX
MRUwEwYDVQQDEwxc3RFeGft.cGxlQ0EwHhcNMjIwNzA0MTgzOTM2WhcNMjIwNzA
0MTgzOTM2WjAsMSowKAYDVQQDEyFzZXJ2ZXJzaWRLIGtleSBnZW5lcmF0ZWQgcm
VzcG9uc2UwggeiMA0GCSCqGSIB3.DQEBAQUAA4IBDwAwggEKAoIBAQC078117tri0
yiiMb9ZZYch8zeizXrjMPF/Rxoz.2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+
DsSMtzxn4l+9tIsVDkAe4FyzN0hL.d/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImD
V3blvaMUKDSJhVbQ+z/G1W1TRx3iW.i5CMHYb+lpJXPTJz/GuWr/b/+EfQwz2Zlw
Gcj4DxIgbx9vG0mftIIXM4TUX28KBb.aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGG
AhglFU8DQiQEkin66GPMtm1SNgitxF.xWouFqpsax5MWn/i52TfEaF2PNTThOuzK
tilweJhkg0gMIQTXAgMBAAGjTTLMAkG.A1UdEwQCMAAwHQYDVR0OBBYEFlylcQN
0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY.MBAAFIR/SsVuU7I5IC+5INpMScsubQ
/zMA0GCSCqGSIB3DQEBBQUAA4IBAQBtIeM.DB9Pkw1GGe7zqyUWVD8y99zowwV6A
rAOXWX+JO0bihgMtZaUfvPCX/LhZVEKDAki.W5orjAEvIu10b6l38ZzX2oyJgkYy
Mmbb14lZTsRyjiqFw9j1PXxwgZvhwcaCF4b7.eDUUBQIEZg3AnkQrEwnHR5oVIN5
8qo0P7PSKC3V13H6DlQh3y7w87nN12923/wk0.v/bS3lv7lDX3HdmbQDlr2KPtBs
JGF4jMdstT7FTx32ZFkObycbk7WJ4LHytnJDci.4iXf+B0S3D6ZbflcXj80/W+jC
GvU0+4SV3cgEXFE5VQvXd8x40W4h0dTSkQCdPOS.nPj4Dl/PsLqX3lDboQAxAA==
.-----END PKCS7-------estServerExampleBoundary--.This is the ep
ilogue. It is also to be ignored..

In text format this is:

HTTP/1.1 200 OK

Status: 200 OK

Content-Type: multipart/mixed ; boundary=estServerExampleBoundary

This is the preamble. It is to be ignored, though it is a handy place for estServer to include an explanatory note including contact or support information.

--estServerExampleBoundary

Content-Type=application/pkcs8

-----BEGIN PRIVATE KEY-----

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBAcwggSjAgEAAoIBAQC078117tri0yii
Mb9ZZYch8zeizXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM
tzxn4l+9tIsVDkAe4FyzN0hLd/zawgj6kUoCi3mxZnb2rWaRYAmM5w4lImDV3blv
aMUKDSJhVbQ+z/GlW1TRx3iWi5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx
Igbx9vG0mftIIXM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhglFU8
DQiQEkin66GPMtmLSNgitxWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhk
g0gMIQTXAgMBAECggEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu
Y/0scVbLbu46WzysoIfJFyUC+zFJnbMCCOPjGbI/4NWkEqc9TAlKz+wDo+hf5bf0
ypFrEmikHk8R3fKpnrKi69ldw0iYnqcFVhq7VtGrSmJcy6Hckwbk7EBoUZGL0wtp
xl06XlhksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTHlH5M4daFbqyRWD7UiPKt
Q3hdwlrpyVZg8ZbBfP0Ej4f9GdRaq88SIKMKCDu3t9ibn/vlkEte+PxhuwyW+d0o
hkKSEW0yLKCzQm5tujsPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv
v0JnNWP7oMzyJ4/eFqdE9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G
gFd93diyRPScVbNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NenZOec
jPMOV/IcGlaLzTqQssL3zo/0gQKBgQDB40lpg3EBggtJ/+dlkLHUW8c7Pe3UyLkS
VxVsyQwioYt8xMeCWuPvPNFcoJcW53KN/YSpCVjpttKGsPtLibMlKYKgasEqgcVl
Vb5OfTA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0g038HDWq7EtqSi9Q0JSMg3YC
QNC0ORptVwKBgQChRcafaYWDhA1l/+g2U9x6Yd56ifF43rCbnV+2EQCvaqi49xC
w4AH+BsoMDlgT5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQHeK92D7
wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJJdqYckgQKBgFXS
zSBzGgwz1/F7AaDZK49mlwPnhyeBb0OqHwbX/Li7lrZlmWef+nSF9Juh/Y77B5/J
UPd09vgGgS00nRk0LIRP2s5OU5IQgQTVLvf8a1UmbVgI+KX511Yi5yMztEwRcjEX
VM9ejXeXN0I57pvqg/xCOK3Kl2eYLh4TO9/E8WjjAoGAAlmqUV4Hnf4yvFlrydMp
fpvoWekiiRE33iEbYZNATYhs17uxwn760pgVifkq2DSrZeYm4+lw9jwWmtUoPzpg
CJYMoG1846nhiZrbbJ5b5twoLV6GRmkK/CfOxPXNzCtSoQA86HHq7rRdhXSau/bY
EXc91tnhLjFzZxdBgrd+f4k=

-----END PRIVATE KEY-----

--estServerExampleBoundary

Content-Type: application/pkcs7-mime

-----BEGIN PKCS7-----

MIIDPAYJKoZIhvcNAQcCoIIDLTCCAYkCAQExADALBgkqhkiG9w0BBwGgggMPMIID
CzCCAFogAwIBAgIBBTANBgkqhkiG9w0BAQUFADAXMRUwEwYDVQQDEwxlc3RFeGFt
cGxlQ0EwHhcNMjIwNzA0MTgzOTM2WhcNMjIwNzA0MTgzOTM2WjAsMSowKAYDVQQD
EyFzZXJ2ZXJzaWRlIGtleSBnZW5lcmF0ZWQgcGVzZG9uc2UwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQC078117tri0yiiMb9ZZYch8zeizXrjMPF/Rxoz
2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSMtzxn4l+9tIsVDkAe4FyzN0hL
d/zawgj6kUoCi3mxZnb2rWaRYAmM5w4lImDV3blvaMUKDSJhVbQ+z/GlW1TRx3iW
i5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4DxIgbx9vG0mftIIXM4TUX28KBb
aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhglFU8DQiQEkin66GPMtmLSNgitxW
ouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhkg0gMIQTXAgMBAAGjTtBLMAK

A1UdEwQCMAAwHQYDVR0OBByEFlylcQN0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY
MBaAFIR/SsVuU7I5IC+5INpMSsubQ/zMA0GCSqGSIb3DQEBAQUAA4IBAQBtIeM
DB9Pkw1GGe7zqvUWVD8y99zowwV6ArAOXWX+J00bihgMtZaUfvPCX/LhZVEKDAki
W5orjAEvIu10b6l38ZzX2oyJgkYyMmbb14lzTsRyjiqFw9j1PXxwgZvhwcaCF4b7
eDUUBQIeZg3AnkQrEwnHR5oVIN58qo0P7PSKC3Vl3H6DlQh3y7w87nN12923/wk0
v/bS3lv7lDX3HdmbQDlr2KPtBsJGF4jMdstT7FTx32ZFKObycbK7WJ4LHytNJDci
4iXf+B0S3D6ZbflcXj80/W+jCGvU0+4SV3cgEXFE5VQvXd8x40W4h0dTSkQCDPOS
nPj4Dl/PsLqX3lDboQAxA==

-----END PKCS7-----

--estServerExampleBoundary--

This is the epilogue. It is also to be ignored.

Authors' Addresses

Max Pritikin (editor)
Cisco Systems, Inc.
510 McCarthy Drive
Milpitas, CA 95035
USA

Email: pritikin@cisco.com

Peter E. Yee (editor)
AKAYLA, Inc.
7150 Moorland Drive
Clarksville, MD 21029
USA

Email: peter@akayla.com

Dan Harkins (editor)
Aruba Networks
1322 Crossman Avenue
Sunnyvale, CA 94089-1113
USA

Email: dharkins@arubanetworks.com

INTERNET-DRAFT
Intended Status: Proposed Standard
Obsoletes: 2560, 6277 (if approved)
Expires: January 29, 2013

S. Santesson
(3xA Security)
July 28, 2012

X.509 Internet Public Key Infrastructure
Online Certificate Status Protocol - OCSP
draft-ietf-pkix-rfc2560bis-05

Abstract

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 document obsoletes RFC 2560 and RFC 6277.

Status of this Memo

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

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

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

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

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

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1.	4
2. Protocol Overview	5
2.1 Request	5
2.2 Response	5
2.3 Exception Cases	7
2.4 Semantics of thisUpdate, nextUpdate and producedAt	7
2.5 Response Pre-production	8
2.6 OCSF Signature Authority Delegation	8
2.7 CA Key Compromise	8
3. Functional Requirements	8
3.1 Certificate Content	8
3.2 Signed Response Acceptance Requirements	9
4. Detailed Protocol	9
4.1 Requests	9
4.1.1 Request Syntax	9
4.1.2 Notes on the Request Syntax	10
4.2 Response Syntax	11
4.2.1 ASN.1 Specification of the OCSF Response	11
4.2.2 Notes on OCSF Responses	14
4.2.2.1 Time	14
4.2.2.2 Authorized Responders	14
4.2.2.2.1 Revocation Checking of an Authorized Responder	15
4.2.2.3 Basic Response	16
4.3 Mandatory and Optional Cryptographic Algorithms	16
4.4 Extensions	17
4.4.1 Nonce	17
4.4.2 CRL References	17
4.4.3 Acceptable Response Types	18
4.4.4 Archive Cutoff	18
4.4.5 CRL Entry Extensions	19
4.4.6 Service Locator	19
4.4.7 Preferred Signature Algorithms	19
4.4.7.1 Extension Syntax	20
4.4.7.2 Responder Signature Algorithm Selection	21
4.4.7.2.1 Dynamic Response	21
4.4.7.2.2 Static Response	22

5.	Security Considerations	23
5.1	Preferred Signature Algorithms	23
5.1.1	Use of insecure algorithms	23
5.1.2	Man in the Middle Downgrade Attack	24
5.1.3	Denial of Service Attack	24
6	IANA Considerations	25
7.	References	25
7.1.	Normative References	25
7.2.	Informative References	25
7.	Acknowledgement	27
Appendix A.	27
A.1	OCSP over HTTP	27
A.1.1	Request	27
A.1.2	Response	27
Appendix B.	ASN.1 Modules	28
B.1.	OCSP in ASN.1	28
B.2.	Preferred Signature Algorithms ASN.1	31
B.2.1.	ASN.1 Module	31
B.2.2.	1988 ASN.1 Module	32
Appendix C.	MIME registrations	32
C.2	application/ocsp-response	33
Authors' Addresses	36

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:

- o Section 4.1.1 specifies the ASN.1 syntax for the nonce extension, which was missing in RFC 2560.
- o 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].
- o Section 2.3 extends the use of the "unauthorized" error response, as specified in [RFC5019].
- o Section 4.2.1 and 4.2.2.3 states that a response may include revocation status information for certificates that were not included in the request, as permitted in [RFC5019].
- o Section 4.3 changes set of cryptographic algorithms that clients must support and the set of cryptographic algorithms that clients should support as specified in [RFC6277].
- o Section 4.2.2.2 has been updated to clarify when a responder is considered an Authorized Responder.

An overview of the protocol is provided in section 2. Functional requirements are specified in section 4. Details of the protocol are in section 5. We cover security issues with the protocol in section 6. Appendix A defines OCSP over HTTP, appendix B accumulates ASN.1 syntactic elements and appendix C specifies the mime types for the messages.

- 1.1. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [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 a certificate (cf. [RFC5280], Section 3.3). Examples include high-value funds transfer or large stock trades.

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

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

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 the prior conditions are 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 requester
- a CA Designated Responder (Authorized Responder) 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
- name of the responder
- responses for each of the certificates in a request
- optional extensions
- signature algorithm OID
- signature computed across 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 the certificate is not revoked, but 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 positive statement about issuance, validity, etc.

The "revoked" state indicates that the certificate has been revoked (either permanently or temporarily (on hold)).

The "unknown" state indicates that the responder doesn't know about the certificate being requested.

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

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

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

2.5 Response Pre-production

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

2.6 OCSF 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 OCSF signing authority by issuing a certificate containing a unique value for `extendedKeyUsage` in the OCSF signer's certificate. This certificate MUST be issued directly to the responder by the cognizant CA.

2.7 CA Key Compromise

If an OCSF 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 OCSF clients a well-known point of information access, CAs SHALL provide the capability to include the `AuthorityInfoAccess` extension (defined in [RFC5280], section 4.2.2.1) in certificates that can be checked using OCSF. Alternatively, the `accessLocation` for the OCSF provider may be configured locally at the OCSF client.

CAs that support an OCSF service, either hosted locally or provided by an Authorized Responder, MUST provide for the inclusion of a value for a `uniformResourceIndicator` (URI) `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 OCSF responder and may contain other transport dependent information (e.g. a URL).

3.2 Signed Response Acceptance Requirements

Prior to accepting a signed response as valid, OCSF clients SHALL confirm that:

1. The certificate identified in a received response corresponds to that which 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 sign the response.
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. Detailed 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, CRLReason

4.1 Requests

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 Request Syntax

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

TBSPRequest      ::=      SEQUENCE {
```

```

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

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

Version ::= INTEGER { v1(0) }

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

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

```

issuerNameHash is the hash of the Issuer's distinguished name. The hash shall be calculated over the DER encoding of the issuer's name field in the certificate being checked. 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. The hash algorithm used for both these hashes, is identified in hashAlgorithm. serialNumber is the serial number of the certificate for which status is being requested.

4.1.2 Notes on the Request Syntax

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, `responseBytes` are 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 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 OCSF Responses

4.2.2.1 Time

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. Responses where the `nextUpdate` value is not set are equivalent to a CRL with no time for `nextUpdate` (see Section 2.4).

The `producedAt` time is the time at which this response was signed.

4.2.2.2 Authorized Responders

TBD ----- UPDATE THIS SECTION -----

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 either sign the OCSF responses itself or it MUST explicitly designate this authority to another entity. OCSF signing delegation SHALL be designated by the inclusion of `id-kp-OCSPSigning` in an `extendedKeyUsage` certificate extension included in the OCSF response signer's certificate. This certificate MUST be issued directly by the CA that issued the certificate in question.

The CA SHOULD use the same issuing key to issue a delegation certificate as was used to sign the certificate being checked for revocation. Systems relying on OCSF responses MUST recognize a delegation certificate as being issued by the CA that issued the certificate in question if the delegation certificate and the certificate being checked for revocation was signed by the same key. Systems relying on OCSF responses MAY recognize a delegation certificate as being issued by the CA that issued the certificate in question if the delegation certificate and the certificate being checked for revocation was issued by the same CA using different issuing keys.

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

Systems or applications that rely on OCSF responses MUST be capable of detecting and enforcing use of the `id-ad-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 fails to 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-ad-ocspSigning in an ExtendedKeyUsage 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 should be NULL. CAs issuing such a certificate should realize that a compromise of the responder's key is as serious as the compromise of a CA key used to sign CRLs, at least for the validity period of this certificate. CA's may choose to issue this type of certificate with a very short lifetime and renew it frequently.

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

- A CA may specify how the responder's certificate be checked for revocation. This can be done using CRL Distribution Points if the check should be done using CRLs or CRL Distribution Points, or Authority Information Access if the check should be done in some other way. Details for specifying either of these two mechanisms are available in [RFC5280].

- 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:

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

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

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

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

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

4.3 Mandatory and Optional Cryptographic Algorithms

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

4.4 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 OCSF Responder, and any extensions which 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 OCSF responder to indicate the CRL on which a revoked or onHold certificate is found. This can be useful where OCSF is used between repositories, and also as an auditing mechanism. The CRL may be specified by a URL (the URL at which the CRL is available), a number (CRL number) or a time (the time at which the relevant CRL was created). 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 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 OCSF server may be operated in a mode whereby the server receives a request and routes it to the OCSF server which 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 it's algorithm preferences, there is always a risk that a server choosing a non-mandatory algorithm, will generate a response that the client may not support.

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

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

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

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

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

This section describes:

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

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 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 4.4.7.1 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 method has the highest precedence:

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

A responder SHOULD always apply the lowest numbered selection mechanism that 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.

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

The use of 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.

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. This criteria may not hold in long term archival applications however in which the status of a certificate is being queried for a date in the distant past, long after the signing algorithm has ceased being considered trustworthy.

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 OCSF 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 OCSF 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 OCSF responder MUST NOT sign OCSF Responses using weak algorithms even if requested by the client. In addition, the client can reject OCSF 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 introduces a slightly increased attack surface for Denial-of-Service attacks where the client request is altered to require algorithms that are not supported by the server. Denial-of-Service considerations from RFC 4732 [RFC4732] are relevant for this document.

6 IANA Considerations

<IANA considerations text>

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3279] 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.
- [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 (1994) | ISO/IEC 8825-1:1995, Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

7.2. Informative References

- [RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

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

7. Acknowledgement

TBD

Appendix A.

A.1 OCSP over HTTP

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

A.1.1 Request

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

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

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

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

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

A.1.2 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 C.1 includes an ASN.1 module that conforms to the 1998 version of ASN.1 for all syntax elements of OCSP other than the preferred signature algorithms extension. An alternative to this module that conforms to the 2002 version of ASN.1 may be found in Section 4 of [RFC5912]. Appendix C.2 includes two ASN.1 modules for the preferred signature algorithms extension, one that conforms to the 1998 version of ASN.1 and one that conforms to the 2002 version of ASN.1.

B.1. OCSP in ASN.1

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

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

```
BEGIN
```

```
IMPORTS
```

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

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

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

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

```
Signature ::= SEQUENCE {
    signatureAlgorithm    AlgorithmIdentifier,
    signature             BIT STRING,
```

```
certs                [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }

Version ::= INTEGER { v1(0) }

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

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

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

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

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

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

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

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

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

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

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

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

UnknownInfo ::= NULL

ArchiveCutoff ::= GeneralizedTime

AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER

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

-- Object Identifiers

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

END
```

B.2. Preferred Signature Algorithms ASN.1

B.2.1. ASN.1 Module

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

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

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

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

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

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

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

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

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

PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm

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

END
```

B.2.2. 1988 ASN.1 Module

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

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL;
IMPORTS

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

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

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

PreferredSignatureAlgorithms ::= SEQUENCE OF PreferredSignatureAlgorithm

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

END
```

Appendix 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 Draft on 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:
Ambarish Malpani <ambarish@valicert.com>

Intended usage: COMMON

Author/Change controller:
Ambarish Malpani <ambarish@valicert.com>

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 Draft on 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:

Ambarish Malpani <ambarish@valicert.com>

Intended usage: COMMON

Author/Change controller:

Ambarish Malpani <ambarish@valicert.com>

Authors' Addresses

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

INTERNET-DRAFT
Intended Status: Proposed Standard
Updates: 5280 (if approved)
Expires: December 28, 2012

P. Yee
AKAYLA
June 26, 2012

Updates to the Internet X.509 Public Key Infrastructure
Certificate and Certificate Revocation List (CRL) Profile
<draft-ietf-pkix-rfc5280-clarifications-05.txt>

Status of this Memo

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

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

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

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

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

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Abstract

This document updates RFC 5280, the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. This document changes the set of acceptable encoding methods for the explicitText field of the user notice policy qualifier and clarifies the rules for converting internationalized domain name labels to ASCII. This document also provides some clarifications on the use of self-signed certificates, trust anchors, and some updated security considerations.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Update to RFC 5280, Section 3.2: Certification Paths and Trust	3
3. Update to RFC 5280, Section 4.2.1.4: Certificate Policies	3
4. Update to RFC 5280, Section 6.2: Using the Path Validation Algorithm	4
5. Update to RFC 5280, Section 7.3: Internationalized Domain Names in Distinguished Names	6
6. Security Considerations	7
7. IANA Considerations	8
8. References	8
8.1. Normative References	8
8.2. Informative References	8
9. Acknowledgements	8
Author's Address	8

1. Introduction

This document updates the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC5280].

This document makes a recommendation that self-signed certificates used to convey trust anchor data be marked as CA certificates, which is not always current practice.

The acceptable and unacceptable encodings for the explicitText field of the user notice policy qualifier are updated to bring them in line with existing practice.

The use of self-signed certificates as trust anchors in Section 6.2 is clarified. While it is optional to use additional information in these certificates in the path validation process, [RFC5937] is noted

as providing guidance in that regard.

The Section 7.3 rules for ASCII encoding of Internationalized Domain Names (IDN) as Distinguished Names are aligned with the rules in Section 7.2 which govern IDN encoding as GeneralNames.

In light of some observed attacks, the Security Considerations now give added depth to the consequences of CA key compromise. This section additionally notes that collision resistance is not a required property of one-way hash functions when used to generate key identifiers.

1.1. Terminology

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

2. Update to RFC 5280, Section 3.2: Certification Paths and Trust

Add the following paragraph to the end of RFC 5280, Section 3.2:

In some cases, a self-signed certificate that does not contain a BasicConstraints extension (and thus is implicitly an EE certificate) is used to convey a public key. Such a certificate (i.e., a self-signed certificate not marked as a CA certificate) is not compatible with the path validation rules described in Section 6, or the definitions in the paragraph above. This document RECOMMENDS that self-signed certificates used to convey trust anchor data be marked (see Section 4.2.1.9) as CA certificates, but acknowledges that this convention is often not adopted in practice. Other uses of self-signed EE certificates are outside the scope of this specification.

3. Update to RFC 5280, Section 4.2.1.4: Certificate Policies

RFC 5280, Section 4.2.1.4, the tenth paragraph says:

An explicitText field includes the textual statement directly in the certificate. The explicitText field is a string with a maximum size of 200 characters. Conforming CAs SHOULD use the UTF8String encoding for explicitText, but MAY use IA5String. Conforming CAs MUST NOT encode explicitText as VisibleString or BMPString. The explicitText string SHOULD NOT include any control characters (e.g., U+0000 to U+001F and U+007F to U+009F). When the UTF8String encoding is used, all character sequences SHOULD be normalized according to Unicode normalization form C (NFC) [NFC].

This paragraph is replaced with:

An explicitText field includes the textual statement directly in the certificate. The explicitText field is a string with a maximum size of 200 characters. Conforming CAs SHOULD use the UTF8String encoding for explicitText, but MAY use VisibleString or BMPString. Conforming CAs MUST NOT encode explicitText as IA5String. The explicitText string SHOULD NOT include any control characters (e.g., U+0000 to U+001F and U+007F to U+009F). When the UTF8String or BMPString encoding is used, all character sequences SHOULD be normalized according to Unicode normalization form C (NFC) [NFC].

4. Update to RFC 5280, Section 6.2: Using the Path Validation Algorithm

RFC 5280, Section 6.2, the third paragraph says:

Where a CA distributes self-signed certificates to specify trust anchor information, certificate extensions can be used to specify recommended inputs to path validation. For example, a policy constraints extension could be included in the self-signed certificate to indicate that paths beginning with this trust anchor should be trusted only for the specified policies. Similarly, a name constraints extension could be included to indicate that paths beginning with this trust anchor should be trusted only for the specified name spaces. The path validation algorithm presented in Section 6.1 does not assume that trust anchor information is provided in self-signed certificates and does not specify processing rules for additional information included in such certificates. Implementations that use self-signed certificates to specify trust anchor information are free to process or ignore such information.

This paragraph is replaced with:

Where a CA distributes self-signed certificates to specify trust anchor information, certificate extensions can be used to specify recommended inputs to path validation. For example, a policy constraints extension could be included in the self-signed certificate to indicate that paths beginning with this trust anchor should be trusted only for the specified policies. Similarly, a name constraints extension could be included to indicate that paths beginning with this trust anchor should be trusted only for the specified name spaces. The path validation algorithm presented in Section 6.1 does not assume that trust anchor information is provided in self-signed certificates and does not specify processing rules for additional information included in such certificates. However, [RFC5937] provides an example of how additional information included in self-signed certificates may be used to initialize the path validation inputs. Implementations that use self-signed certificates to specify trust anchor information are free to make

| use of any additional information that is included in the
| certificates, or to ignore such information.

5. Update to RFC 5280, Section 7.3: Internationalized Domain Names in Distinguished Names

RFC 5280, Section 7.3, the first paragraph says:

Domain Names may also be represented as distinguished names using domain components in the subject field, the issuer field, the subjectAltName extension, or the issuerAltName extension. As with the dNSName in the GeneralName type, the value of this attribute is defined as an IA5String. Each domainComponent attribute represents a single label. To represent a label from an IDN in the distinguished name, the implementation MUST perform the "ToASCII" label conversion specified in Section 4.1 of RFC 3490. The label SHALL be considered a "stored string". That is, the AllowUnassigned flag SHALL NOT be set.

This paragraph is replaced with:

Domain Names may also be represented as distinguished names using domain components in the subject field, the issuer field, the subjectAltName extension, or the issuerAltName extension. As with the dNSName in the GeneralName type, the value of this attribute is defined as an IA5String. Each domainComponent attribute represents a single label. To represent a label from an IDN in the distinguished name, the implementation MUST perform the "ToASCII" label conversion specified in Section 4.1 of RFC 3490 with the UseSTD3ASCIIRules flag set. The label SHALL be considered a "stored string". That is, the AllowUnassigned flag SHALL NOT be set. The conversion process is the same as is performed in step 4 in Section 7.2.

6. Security Considerations

This document modifies the Security Considerations section of RFC 5280 as follows. The fifth paragraph of the Security Considerations section of RFC 5280 says:

The protection afforded private keys is a critical security factor. On a small scale, failure of users to protect their private keys will permit an attacker to masquerade as them or decrypt their personal information. On a larger scale, compromise of a CA's private signing key may have a catastrophic effect. If an attacker obtains the private key unnoticed, the attacker may issue bogus certificates and CRLs. Existence of bogus certificates and CRLs will undermine confidence in the system. If such a compromise is detected, all certificates issued to the compromised CA MUST be revoked, preventing services between its users and users of other CAs. Rebuilding after such a compromise will be problematic, so CAs are advised to implement a combination of strong technical measures (e.g., tamper-resistant cryptographic modules) and appropriate management procedures (e.g., separation of duties) to avoid such an incident.

This paragraph is replaced with:

The protection afforded private keys is a critical security factor. On a small scale, failure of users to protect their private keys will permit an attacker to masquerade as them or decrypt their personal information. On a larger scale, compromise of a CA's private signing key may have a catastrophic effect.

If an attacker obtains the private key of a CA unnoticed, the attacker may issue bogus certificates and CRLs. Even if an attacker is unable to obtain a copy of a CA's private key, the attacker may be able to issue bogus certificates and CRLs by making unauthorized use of the CA's workstation or of an RA's workstation. Such an attack may be the result of an attacker obtaining unauthorized access to the workstation, either locally or remotely, or may be the result of inappropriate activity by an insider. Existence of bogus certificates and CRLs will undermine confidence in the system. Among many other possible attacks, the attacker may issue bogus certificates that have the same subject names as legitimate certificates in order impersonate legitimate certificate subjects. This could include bogus CA certificates in which the subject names in the bogus certificates match the names under which legitimate CAs issue certificates and CRLs. This would allow the attacker to issue bogus certificates and CRLs that have the same issuer names, and possibly the same serial numbers, as certificates and CRLs issued by legitimate CAs.

The following text is added to the end of the Security Considerations section of 5280:

One-way hash functions are commonly used to generate key identifier values (AKI and SKI), e.g., as described in Sections 4.1.1 and 4.1.2. However, none of the security properties of such functions are required for this context.

7. IANA Considerations

This document has no actions for IANA.

8. References

8.1. Normative References

- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.

8.2. Informative References

- [RFC5937] Ashmore, S. and C. Wallace, "Using Trust Anchor Constraints during Certification Path Processing", RFC 5937, August 2010.
- [X.680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.
- [NFC] Davis, M. and M. Duerst, "Unicode Standard Annex #15: Unicode Normalization Forms", October 2006, <<http://www.unicode.org/reports/tr15/>>.

9. Acknowledgements

David Cooper is acknowledged for his fine work in editing versions 00 through 04 of this document.

Author's Address

Peter E. Yee
AKAYLA
7150 Moorland Drive
Clarksville, MD 21029
USA

EMail: peter@akayla.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2013

S. Josefsson
SJD AB
S. Leonard
Penango, Inc.
July 17, 2012

Text Encodings of PKIX and CMS Structures
draft-josefsson-pkix-textual-01

Abstract

This document describes and discuss the text encodings of Public-Key Infrastructure using X.509 (PKIX) Certificates, PKIX Certificate Revocation Lists (CRLs), PKCS #10 Certification Request Syntax, PKCS #7 structures, Cryptographic Message Syntax (CMS), PKCS #8 Private-Key Information Syntax, and Attribute Certificates. The text encodings are well-known, are implemented by several applications and libraries, and are widely deployed. This document is intended to articulate the de-facto rules that existing implementations operate by, and to give recommendations that will promote interoperability going forward.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. General Considerations	4
3. ABNF	4
4. Text Encoding of PKIX Certificates	5
4.1. Encoding	5
4.2. Explanatory Text	6
4.3. File Extension	7
5. Text Encoding of PKIX CRLs	7
6. Text Encoding of PKCS #10 Certification Request Syntax	7
7. Text Encoding of PKCS #7 Cryptographic Message Syntax	8
8. Text Encoding of Cryptographic Message Syntax	9
9. Text Encoding of PKCS #8 Private Key Info, and One Asymmetric Key	9
10. Text Encoding of PKCS #8 Encrypted Private Key Info	9
11. Text Encoding of Attribute Certificates	10
12. Non-Conforming Examples	10
13. Security Considerations	12
14. IANA Considerations	12
15. Acknowledgements	12
16. References	13
16.1. Normative References	13
16.2. Informative References	13
Editorial Comments	
Authors' Addresses	14

1. Introduction

Several security-related standards used on the Internet define data formats that are normally encoded using Distinguished Encoding Rules (DER) [CCITT.X690.2002], which is a binary data format. This document is about text encodings of some of these formats:

1. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC5280], for both Certificates and Certificate Revocation Lists (CRLs).
2. PKCS #10: Certification Request Syntax [RFC2986].
3. PKCS #7: Cryptographic Message Syntax [RFC2315].
4. Cryptographic Message Syntax [RFC5652].
5. PKCS #8: Private-Key Information Syntax [RFC5208] and One Asymmetric Key (in Asymmetric Key Package [RFC5958]).
6. An Internet Attribute Certificate Profile for Authorization [RFC5755].

A disadvantage of a binary data format is that it cannot be interchanged in textual transports, such as e-mail or text documents. One advantage with text encodings is that they are easy to modify using common text editors; for example, a user may concatenate several certificates to form a certificate chain with copy-and-paste operations.

The tradition within the RFC series can be traced back to PEM [RFC1421], based on a proposal by M. Rose in Message Encapsulation [RFC0934]. Originally called "PEM encapsulation mechanism", "encapsulated PEM message", or (arguably) "PEM printable encoding", today the format is sometimes referred to as "PEM encoding". Variations include OpenPGP ASCII Armor and OpenSSH Key File Format.

For reasons that basically boil down to non-coordination (or gross inattention), many PKIX and CMS libraries implement a text encoding that is similar to--but not identical with--PEM encoding. This Internet-Draft calls this format "PKIX text encoding", articulates the de-facto rules that most implementations operate by, and provides recommendations that will promote interoperability going forward. Peter Gutmann's X.509 Style Guide [X509SG] contains a section "base64 Encoding" that describes the formats and contains suggestions similar to what is in this document.

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

2. General Considerations

PKIX text encoding begins with a line starting with "-----BEGIN" and ends with a line starting with "-----END". Between these lines, or "encapsulation boundaries", are base64 [RFC4648]-encoded data. Data before the "-----BEGIN" and after the "-----END" encapsulation boundaries are permitted and MUST NOT cause parsers to malfunction. Furthermore, parsers MUST ignore whitespace and other non-alphabetic characters [DP1] and MUST handle different newline conventions.

The type of data encoded is labeled depending on the type label in the "-----BEGIN" line (pre-encapsulation boundary). For example, the line may be "-----BEGIN CERTIFICATE-----" to indicate that the content is a PKIX certificate (see further below). Generators MUST put the same label on the "-----END" line (post-encapsulation boundary) as the corresponding "-----BEGIN" line. Parsers MAY disregard the label on the "-----END" line instead of signaling an error if there is a label mismatch.

The label type implies that the encoded data follows the specified syntax. Parsers MUST handle non-conforming data gracefully. However, not all parsers or generators prior to this Internet-Draft behave consistently. A conforming parser MAY interpret the contents as another label type, but ought to be aware of the security implications discussed in the Security Considerations section.

Unlike PEM encoding, OpenPGP ASCII armor, and OpenSSH key file format, PKIX text encoding does NOT define or permit attributes to be encoded alongside the PKIX or CMS data. Whitespace MAY appear between the pre-encapsulation boundary and the base64, but generators SHOULD NOT emit such whitespace.

Files MAY contain multiple instances of the text encoded representation. This is used, for example, when a file contains several certificates. Whether the instances are ordered or unordered depends on the context.

Generators MUST wrap the base64 encoded lines so that each line consists of exactly 64 characters except for the final line which will encode as much data is left (within the 64 character line boundary). Parsers MAY handle other line sizes. These requirements are consistent with PEM [RFC1421].

3. ABNF

The ABNF of the PKIX text encoding is:

```
pkixmsg      ::= preeb
               *eolWSP
               base64text
               posteb

preeb        ::= "-----BEGIN " label "-----" eol

posteb       ::= "-----END " label "-----" eol

base64char   ::= ALPHA / DIGIT / "+" / "/"

base64pad    ::= "="

base64line   ::= 1*base64char eol

base64finl   ::= *base64char *2base64pad eol ; implies that:
                                              ; ...AB= <CRLF> = <CRLF>
                                              ; is invalid. not sure
                                              ; if this is a good idea

base64text   ::= *base64line base64finl
; we could also use <enbinbody> from RFC 1421,
; which requires 16 groups of 4 chars, which means 64 chars
; exactly per line, except the final line

labelchar    ::= %x21-2C / %x2E-%7E ; any printable character,
                                   ; except hyphen

label        ::= labelchar *(labelchar / labelchar "-" / SP) labelchar

eol          ::= CRLF / CR / LF

eolWSP       ::= WSP / CR / LF ; compare with LWSP
```

Figure 1: ABNF

4. Text Encoding of PKIX Certificates

4.1. Encoding

PKIX certificates are encoded using the "CERTIFICATE" label. The encoded data MUST be a DER encoded ASN.1 "Certificate" structure as described in section 4 of [RFC5280].

```

-----BEGIN CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkJOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
A1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2VydgG1maWNhdGUgYXV0aG9y
aXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHedudVRMUyBjZXJ0aWZpY2F0
ZSBhdXR0b3JpdHkwHhcNMTEwNTIzMjAzODIxWhcNMTEwNTIzMjIyMDc0MTUxWjB9MQsw
CQYDVQQGEwJCRTEPMA0GA1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2Vy
dG1maWNhdGUgYXV0aG9yaXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHedu
dVRMUyBjZXJ0aWZpY2F0ZSBhdXR0b3JpdHkwWTATBgqhkJOPQIBBgqhkJOPQMB
BwNCAARS2I0jiuNn14Y2sSALCX3IybqiIJUvxUpj+oNfzngvj/Niyv2394BWNW4X
uQ4RTEiywK87WRcWMGgJB5kX/t2no0MwQTAPBgNVHRMBAf8EBTADAQH/MA8GA1Ud
DwEB/wQFAwMHBGAWHQYDVR00BBYEFPC0gf6YEr+1KLlkQAPLzB9mTigDMAoGCCqG
SM49BAMCA0gAMEUCIDGuwDlKPyG+hRf88MeyMQcqOFZD0TbVleF+UsAGQ4enAiEA
l4wOuDWkQa+upc8GftXE2C//4mKANBC6It0lgUaTIpo=
-----END CERTIFICATE-----

```

Figure 2: Certificate Example

Historically the label "X509 CERTIFICATE" and also, less common, "X.509 CERTIFICATE" have been used. Generators conforming to this document MUST generate "CERTIFICATE" labels and MUST NOT generate "X509 CERTIFICATE" or "X.509 CERTIFICATE" labels. Parsers are NOT RECOMMENDED to treat "X509 CERTIFICATE" or "X.509 CERTIFICATE" as equivalent to "CERTIFICATE", but a valid exception may be for backwards compatibility (potentially together with a warning).

4.2. Explanatory Text

Many tools are known to emit explanatory text before the BEGIN and after the END labels for PKIX certificates, more than any other type. If emitted, such text SHOULD be related to the certificate, such as providing a textual representation of key data elements in the certificate.

```

Subject: CN=Atlantis
Issuer: CN=Atlantis
Validity: from 7/9/2012 3:10:38 AM UTC to 7/9/2013 3:10:37 AM UTC
-----BEGIN CERTIFICATE-----
MIIBmTCCAUEgAwIBAgIBKjAJBgUrDgMCHQUAMBMxETAPBgNVBAMTCEF0bGFudG1z
MB4XDTEyMDcwOTAzMTAzOFoXDTEzMDcwOTAzMTAzNlowEzERMA8GA1UEAxMIQXR5
YW50aXNwXDNANBgkqhkiG9w0BAQEFAANLADBIAGIAu+BXo+miabDIHHx+yquqzqNh
Ryn/XtkJIIHVCyYtHvIX+S1x5ErgMoHhycpoxbErZmVR4GCq1S2diNmRFZCRtQID
AQABO4GJMIGMAwGA1UdEwEB/wQCMAAwIAYDVR0EAQH/BBYwFDAOMAAGCisGAQQB
gjcCARUDAgeAMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDAzA1BgNVHQEE
LjAsGBA0jOnSSuIHymnVryHAdywMoRUwEzERMA8GA1UEAxMIQXR5YW50aXNwXDNAN
BgkqhkiG9w0BAQFAANBAKi6HRBaNEL5R0n56nvfc1QNaXiDTl74uf+lojzA4lhVinc0
ILwpnZlzl4MlI9eCSHhVQBHEp2uQdXJB+d5Byg=
-----END CERTIFICATE-----

```

Figure 3: Certificate Example with Explanatory Text

4.3. File Extension

Although text encodings of PKIX structures can occur anywhere, many tools are known to offer an option to encode PKIX structures in this text encoding. To promote interoperability and to separate DER encodings from text encodings, This Internet-Draft RECOMMENDS that the extension ".crt" be used for this text encoding. Implementations should be aware that in spite of this recommendation, many tools still default to encode certificates in this text encoding with the extension ".cer".

5. Text Encoding of PKIX CRLs

PKIX CRLs are encoded using the "X509 CRL" label. The encoded data MUST be a DER encoded ASN.1 "CertificateList" structure as described in Section 5 of [RFC5280].

```
-----BEGIN X509 CRL-----
MIIB9DCCAV8CAQEwCwYJKoZIhvcNAQEFMIIBCDEXMBUGA1UEChMOVmVyaVNpZ24s
IEluYy4xHzAdBgNVBAsTF1Zlcm1TaWduIFRydXN0IE5ldHdvcmxRjBEBGVBAsT
PXd3dy52ZXJpc2lnbi5jb20vcnVwb3NpdG9yeS9SUEEgSW5jb3JwLiBieSBSZWYu
LEExJQUiUwTFRyZG90IFZhbGlkYXRlZDEm
MCQGA1UECzMdRGlnaXRhbCBJRCDDbGFzcyAxIC0gTmV0c2NhcnGUxGDAWBgNVBAMU
D1NpbW9uIEpvc2Vmc3NvbG90IEpvc2Vmc3NvbG90IEpvc2Vmc3NvbG90IEpvc2Vmc3
Lm9yZxcnMDYxMjI3MDgwMjM0MDIzNFowCwYJKoZIhvcNAQEFMCECEC4QNwPfRoWd
elUNplllhhTgXDTA2MTIyNzA4MDIzNFowCwYJKoZIhvcNAQEFMCECEC4QNwPfRoWd
Nbrq1Dn5IKL8nXLgPGChv1I/le1MNO9tlohGQxB5HnFUKRPAY82fR6Epor4aHgVy
b+5y+neKN9Kn2mPF4iiun+a4o26CjJ0pArojCLlp8T0yyi9Xxvyc/ezaZ98HiIyP
c3DGMNR+oUmSjKZ0jIhAYmeLxaPHfQwR
-----END X509 CRL-----
```

Figure 4: CRL Example

Historically the label "CRL" has rarely been used. Today it is not common and many popular tools do not understand the label. Therefore, this document standardizes "X509 CRL" in order to promote interoperability and backwards-compatibility. Generators conforming to this document MUST generate "X509 CRL" labels and MUST NOT generate "CRL" labels. Parsers are NOT RECOMMENDED to treat "CRL" as equivalent to "X509 CRL".

6. Text Encoding of PKCS #10 Certification Request Syntax

PKCS #10 Certification Requests are encoded using the "CERTIFICATE

REQUEST" label. The encoded data MUST be a DER encoded ASN.1 "CertificationRequest" structure as described in [RFC2986].

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQcCAQAwTjELMAkGA1UEBhMCU0UxJzAlBgNVBAoTHlNpbW9uIEpvc2Vm
c3NvbiBEYXRha29uc3VsdCBBCjEWMBQGA1UEAxMNam9zZWZzc29uLm9yZzBOMBAG
ByqGSM49AgEGBSuBBAAhAzoABLLPSkuXY0166MbxVJ3Mot5FCFuqQfn6dTs+9/CM
EOlSwVej77tj56kj9R/j9Q+LfysX8FO9I5p3oGIwYAYJKoZIhvcNAQkOMVMwUTAY
BgNVHREETAPgg1qb3NlZnNzb24ub3JnMAwGA1UdEwEB/wQCMAAwDwYDVR0PAQH/
BAUDAwegADAWBgNVHSUBAf8EDDAKBgggBgEFBQcDATAKBgggghkjOPQQDAGM/ADA8
AhxBvfhxPFfbBbsElNoFmCUczOFAPeUQVUw3ZP69AhwWXk3dgSUsKnuwL5g/ftAY
dEQc8B8jAcnuOrfU
-----END CERTIFICATE REQUEST-----
```

Figure 5: PKCS #10 Example

The label "NEW CERTIFICATE REQUEST" is also in wide use. Generators conforming to this document MUST generate "CERTIFICATE REQUEST" labels. Parsers MAY treat "NEW CERTIFICATE REQUEST" as equivalent to "CERTIFICATE REQUEST".

7. Text Encoding of PKCS #7 Cryptographic Message Syntax

PKCS #7 Cryptographic Message Syntax structures are encoded using the "PKCS7" label. The encoded data MUST be a DER encoded ASN.1 "ContentInfo" structure as described in [RFC2315].

```
-----BEGIN PKCS7-----
MIHjBgsqhkig9w0BCRABF6CB0zCB0AIBADFhol8CAQCgGwYJKoZIhvcNAQUMMA4E
CLfrI6dr0gUWAgITiDAjBgsqhkig9w0BCRADCTAUBgggghkiG9w0DBwQIZpECRWtz
u5kEGDCjerXY8odQ7EEErOmZJvAurk/j81IrozBSBgkqhkiG9w0BBwEwMwYlKoZI
hvcNAQkQAww8wJDAUBgggghkiG9w0DBwQI0tCBcU09nxEwDAYIKwYBBQUIAQIFAIAQ
OsYGYUFDahORnc1p4VbKEAQUM2Xo8PMHBoYdqEcsbTodlCFAZH4=
-----END PKCS7-----
```

Figure 6: PKCS #7 Example

The label "CERTIFICATE CHAIN" has been in use to denote a degenerative PKCS #7 structure that contains only a list of certificates. Several modern tools do not support this label. Generators MUST NOT generate the "CERTIFICATE CHAIN" label. Parsers are NOT RECOMMENDED to treat "CERTIFICATE CHAIN" as equivalent to "PKCS7".

PKCS #7 is an old standard that has long been superseded by CMS. Implementations SHOULD NOT generate PKCS #7 when CMS is an alternative.

8. Text Encoding of Cryptographic Message Syntax

Cryptographic Message Syntax structures are encoded using the "CMS" label. The encoded data MUST[mustshould2] be a DER encoded ASN.1 "ContentInfo" structure as described in [RFC5652].

```
-----BEGIN CMS-----
MIGDBgsqhkIG9w0BCRABCaB0MHICAQAwDQYLKoZIhvcNAQkQAwwXgYJKoZIhvcN
AQcBoFEET3icc87PK0nNK9ENqSxItVioSa0o0S/ISczMslZizkgsKk4tsQ0NlnUM
dvb05OXi5XLPLEtViMwvLVLwSE0sKlFIVHAqSk3MBkkBAJv0Fx0=
-----END CMS-----
```

Figure 7: CMS Example

CMS is the IETF successor to PKCS #7. Section 1.1.1 of RFC 5652 describes the changes since PKCS #7 v1.5. Implementations SHOULD generate CMS when it is an alternative, promoting interoperability and forwards-compatibility.

9. Text Encoding of PKCS #8 Private Key Info, and One Asymmetric Key

The PrivateKeyInfo structure of PKCS #8 Private Key Information Syntax, renamed to OneAsymmetricKey in [RFC5958], is encoded using the "PRIVATE KEY" label. The encoded data SHOULD be a DER encoded ASN.1 "PrivateKeyInfo" structure as described in PKCS #8, or the "OneAsymmetricKey" structure as described in [RFC5958]. The two are semantically identical, and can be distinguished by version number.

```
-----BEGIN PRIVATE KEY-----
MIGEAgEAMBAGByqGSM49AgEGBSuBBAKBG0wawIBAQQgVcB/UNPxalR9zDYAjQIf
jojUDIQuGnSJrFEEzZPT/92hRANCAASc7UJtgnF/abqWM60T3XNJEzBv5ez9TdwK
H0M6xpM2q+53wmsN/eYLdgtjgBd3DBmHtPilCkiFICXyaaA8z9LkJ
-----END PRIVATE KEY-----
```

Figure 8: PKCS #8 PrivateKeyInfo Example

10. Text Encoding of PKCS #8 Encrypted Private Key Info

The EncryptedPrivateKeyInfo structure of PKCS #8 Private Key Information Syntax, called the same in [RFC5958], is encoded using the "ENCRYPTED PRIVATE KEY" label. The encoded data SHOULD be a DER encoded ASN.1 "EncryptedPrivateKeyInfo" structure as described in PKCS #8 and [RFC5958].

```

-----BEGIN ENCRYPTED PRIVATE KEY-----
MIHNMEAGCSqGSIb3DQEFDTAzMBsGCSqGSIb3DQEFDDAOBAGhhICA6T/51QICCAAw
FAYIKoZIhvcNAwcECBCxDgviI59i9BIGIY3CAqlMNBgaSI5QiiWVNJ3IpflnEiEsW
Z0JIoHyRmKK/+cr9QPLnzxImm0TR9s4JrG3CilzTWvb0jIvbG3hu0zyFPraoMkap
8eRzWsIvC5SVel+CSjoS2mVS87cyj1D+txrmrXOVYDE+eTgMLbrLmsWh3QkCTRtF
QC7k0NNzUHTV9yGDwfqMbw==
-----END ENCRYPTED PRIVATE KEY-----

```

Figure 9: PKCS #8 EncryptedPrivateKeyInfo Example

11. Text Encoding of Attribute Certificates

Attribute certificates are encoded using the "ATTRIBUTE CERTIFICATE" label. The encoded data MUST be a DER encoded ASN.1 "AttributeCertificate" structure as described in [RFC5755].

```

-----BEGIN ATTRIBUTE CERTIFICATE-----
MIICKzCCAZQCAQEwgZeggZQwgYmkgyYYwgYmxCzAJBgNVBAYTA1VTMREwDwYDVQQI
DAh0ZXcgWW9yazEUMBIGA1UEBwwLU3RvbnkgQnJvb2sxZDZANBgNVBAoMBkNTRTU5
MjE6MDgGA1UEAwxU2NvdHQGU3RhbGxlc3R0YXN0YXN0ZXJlbnR0YXN0ZXJlbnR0
aWwMuc3VueXNiLmVkdQIGARWRgUUSoIGMMIGJpIGGMIGDMQswCQYDVQQGEwJVUzER
MA8GA1UECAwITmV3IFlvcmsxZDASBgNVBACMC1N0b255IEJyb29rMQ8wDQYDVQQK
DAZDU0U1OTIxOjA4BgNVBAMMMVNjb3R0IFN0YWxsZXIvZW1haWxBZGRyZXNzPjNz
dGFsbGVyQGljLnNlbnlzYi5lZHUwDQYJKoZIhvcNAQEFBQACBgEVq4FFSjAiGA8z
OTA3MDIwMTA1MDAwMfoYDzM5MTEwMTMxMDUwMDAwWjArMCKGA1UYSDEiMCCGHmh0
dHA6Ly9pZGVyYXN0b3R0YXN0ZXJlbnR0YXN0ZXJlbnR0YXN0ZXJlbnR0YXN0
M9axFPXXozEFcer06bj9MCBCCQLtAM7ZXcZjcxyva7xCBDmtZXPYUluHf5OcWPJz
5XPus/xS9wBgtlM3fldIKNyNO8RsMp6Ocx+PGLICc7zpZiGmCYLl641AEGPO/bsw
SmluaklaZIttePeTAHeJJ58izNJ5aR3Wcd3A5gLztQ==
-----END ATTRIBUTE CERTIFICATE-----

```

Figure 10: Attribute Certificate Example

12. Non-Conforming Examples

[DPncfex] This section contains examples for the non-recommended label variants described earlier in this document. As discussed earlier, supporting these are not required and sometimes discouraged. Still, they can be useful for interoperability testing and for easy reference.

```

-----BEGIN X509 CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
A1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2VydgG1maWNhdGUgYXV0aG9y
aXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdudVRMUyBjZXXJ0aWZpY2F0
ZSBhdXR0b3JpdHkwHhcNMTEwNTIzMjAzODIxWhcNMTEwNTIzMjIyMDc0MTUxWjB9MQsw
CQYDVQQGEwJCRTEPMA0GA1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2Vy
dG1maWNhdGUgYXV0aG9yaXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdu
dVRMUyBjZXXJ0aWZpY2F0ZSBhdXR0b3JpdHkwWTATBgqhkhjOPQIBBggqhkjOPQMB
BwNCAARS2I0jiuNn14Y2ssSALCX3IybqiIJUvxUpj+oNfzngvj/Niyv2394BWNW4X
uQ4RTEiywK87WRcWMGgJB5kX/t2no0MwQTAPBgNVHRMBAf8EBTADAQH/MA8GA1Ud
DwEB/wQFAwMHBGAWHQYDVR0OBBYEFPC0gf6YEr+1KL1kQAPLzB9mTigDMAoGCCqG
SM49BAMCA0gAMEUCIDGuwD1KPyG+hrf88MeyMQcqOFZD0TbVleF+UsAGQ4enAiEA
l4wOuDWKQa+upc8GftXE2C//4mKANBC6It0lgUaTIpo=
-----END X509 CERTIFICATE-----

```

Figure 11: Non-standard 'X509' Certificate Example

```

-----BEGIN X.509 CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
A1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2VydgG1maWNhdGUgYXV0aG9y
aXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdudVRMUyBjZXXJ0aWZpY2F0
ZSBhdXR0b3JpdHkwHhcNMTEwNTIzMjAzODIxWhcNMTEwNTIzMjIyMDc0MTUxWjB9MQsw
CQYDVQQGEwJCRTEPMA0GA1UEChMGR251VExTMSUwIwYDVQQLExxHbnVUTFMgY2Vy
dG1maWNhdGUgYXV0aG9yaXR5MQ8wDQYDVQQIEwZMZXV2ZW4xJTAjBgNVBAMTHEdu
dVRMUyBjZXXJ0aWZpY2F0ZSBhdXR0b3JpdHkwWTATBgqhkhjOPQIBBggqhkjOPQMB
BwNCAARS2I0jiuNn14Y2ssSALCX3IybqiIJUvxUpj+oNfzngvj/Niyv2394BWNW4X
uQ4RTEiywK87WRcWMGgJB5kX/t2no0MwQTAPBgNVHRMBAf8EBTADAQH/MA8GA1Ud
DwEB/wQFAwMHBGAWHQYDVR0OBBYEFPC0gf6YEr+1KL1kQAPLzB9mTigDMAoGCCqG
SM49BAMCA0gAMEUCIDGuwD1KPyG+hrf88MeyMQcqOFZD0TbVleF+UsAGQ4enAiEA
l4wOuDWKQa+upc8GftXE2C//4mKANBC6It0lgUaTIpo=
-----END X.509 CERTIFICATE-----

```

Figure 12: Non-standard 'X.509' Certificate Example

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBWDCCAQCcCAQAwTjELMAkGA1UEBhMCU0UxJzA1BgNVBAoTHlNpbW9uIEpvc2Vm
c3NvbiBEYXRha29uc3VsdCBBCQjEwMBQGA1UEAxMNam9zZWZzc29uLm9yZzBOMBAG
ByqGSM49AgEGBSuBBAAhAzoABLLPSkuXY0166MbxVJ3Mot5FCFuqQfn6dTs+9/CM
EOlSwVeJ77tj56kj9R/j9Q+LfysX8FO9I5p3oGIwYAYJKoZIhvcNAQkOMVMwUTAY
BgNVHREETAPgg1qb3N1ZnNzb24ub3JnMAwGA1UdEwEB/wQCMAAwDwYDVR0PAQH/
BAUDAwegADAwBgNVHSubAF8EDDAKBgggBgEFBQcDATAKBggqhkjOPQQDAgM/ADA8
AhxBvfHxPFfbBbsE1NoFmCUczOFApEuQVUw3ZP69AhwWXk3dgsUsKnuwL5g/ftAY
dEQc8B8jAcnuOrfU
-----END NEW CERTIFICATE REQUEST-----

```

Figure 13: Non-standard 'NEW' PKCS #10 Example


```
-----BEGIN CERTIFICATE CHAIN-----
MIHjBgsqhkIG9w0BCRABF6CB0zCB0AIBADFho18CAQCgGwYJKoZIhvcNAQUMMA4E
CLfrI6dr0gUWAgITiDAjBgsqhkIG9w0BCRADCTAUBggqhkIG9w0DBwQIZpECRWtz
u5kEGDCjerXY8odQ7EEEromZJvAurk/j8lIrozBSBgkqhkiG9w0BBwEwMwYLKoZI
hvcNAQkQAw8wJDAUBggqhkIG9w0DBwQI0tCBcU09nxEwDAYIKwYBBQUIAQIFAIAQ
OsYGYUFDaH0RNclp4VbKEAQUM2Xo8PMHBoYdqEcsbTodlCFAZH4=
-----END CERTIFICATE CHAIN-----
```

Figure 14: Non-standard 'CERTIFICATE CHAIN' Example

13. Security Considerations

Data in this format often originates from untrusted sources, thus parsers must be prepared to handle unexpected data without causing security vulnerabilities.

Ambiguities are introduced by having more than one canonical encoding of the same data. The first ambiguity is introduced by permitting the text encoded representation instead of the binary DER encoding, but further ambiguities arise when multiple labels are treated as similar. Variations of whitespace and non-base64 alphabetic characters can create further ambiguities. Implementations that rely on canonical representation or the ability to fingerprint a particular data format need to understand that this Internet-Draft does not define canonical encodings. If canonical encodings are desired, the encoded structure must be decoded and processed into a canonical form (namely, DER encoding). Data encoding ambiguities also create opportunities for side channels.

14. IANA Considerations

This document implies no IANA Considerations.

15. Acknowledgements

Peter Gutmann suggested to document labels for Attribute Certificates and PKCS #7 messages, and to add examples for the non-standard variants.

16. References

16.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC5208] Kaliski, B., "Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2", RFC 5208, May 2008.
- [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.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.
- [RFC5755] Farrell, S., Housley, R., and S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC 5755, January 2010.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.
- [CCITT.X690.2002] International International Telephone and Telegraph Consultative Committee, "ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER)", CCITT Recommendation X.690, July 2002.

16.2. Informative References

- [RFC0934] Rose, M. and E. Stefferud, "Proposed standard for message encapsulation", RFC 934, January 1985.
- [RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication

Procedures", RFC 1421, February 1993.

[RFC2015] Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, October 1996.

[X509SG] Gutmann, P., "X.509 Style Guide", WWW <http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt>, October 2000.

Editorial Comments

[DP1] S.L.: Non-alphabetic characters is too broad. Characters such as "+", "/", and "=" are valid base64; characters such as "-" and "_" are alternate base64 characters but are not used in this specification. In any event, any non-whitespace characters will cause existing implementations to fail.

[DPncfex] S.L.: The utility of this section is questionable. We can shorten up the RFC by removing this section.

[mustshould1] S.L.: SHOULD?

[mustshould2] S.L.: SHOULD?

Authors' Addresses

Simon Josefsson
SJD AB
Johan Olof Wallins Vaeg 13
Solna 171 64
SE

Email: simon@josefsson.org
URI: <http://josefsson.org/>

Sean Leonard
Penango, Inc.
1215 K Street
17th Floor
Sacramento, CA 95814
USA

Email: dev+iETF@seantek.com
URI: <http://www.penango.com/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2013

S. Leonard
Penango, Inc.
July 8, 2012

A Uniform Resource Name (URN) Namespace for Certificates
draft-seantek-certspec-00

Abstract

Digital certificates are used in many systems and protocols to identify and authenticate parties. This document describes a Uniform Resource Name (URN) namespace that identifies certificates. These URNs can be used when certificates need to be identified by value or reference. Applications can also use this specification in non-URI contexts.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 9, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

described in the Simplified BSD License.

1. Introduction

Digital certificates are used in many systems and protocols to identify and authenticate parties. Security considerations frequently require that the certificate must be identified with certainty, because selecting the wrong certificate will lead to validation errors (resulting in denial of service) or improper credential selection (resulting in unwanted disclosure or substitution attacks). The goal of this namespace is to provide a uniform syntax for identifying certificates with precision in Uniform Resource Identifiers (URIs), specifically Uniform Resource Names (URNs).

Using this syntax, any protocol or system that refers to a certificate in a textual format can unambiguously identify that certificate by value or reference. Implementers that parse these URNs can resolve them into actual certificates. Examples:

```
urn:cert:SHA-1:blf090a8e2d70353107454f9618347b18b321bf1
urn:cert:issuersn:CN=Atlantis;2A
```

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.

2. Motivation and Purpose

Although certificates have diverse applications, there has been traditionally no way to refer to a certificate uniformly and unambiguously by reference in text. Certificates that identify long public keys (e.g., 2048-bit RSA keys) and that contain required and recommended PKIX extensions can easily exceed many kilobytes in length, which are impractical for certain applications to store by value.

The purpose of this specification is to provide a uniform textual format for identifying individual certificates. When a resolver resolves a certificate specification, the resolver's output is either a single certificate or nothing. This specification is not designed or intended to provide a search tool or query language to match multiple certificates. Identifying a specific certificate by reference or value allows diverse applications to have a common

syntax. For example, applications can store certspecs as local or shared preferences, so that users can edit them without resorting to application-specific storage formats. When conveyed in protocol, a certspec can identify a specific certificate to a client or server using text-based formats such as YAML, XML, JSON, and others.

3. certspec Syntax

The Namespace Specific String (NSS) portion of a certspec has the following ABNF specification:

```
NSS = spec-type ":" spec-value ( '?' certattrs )
spec-type = scheme
certattrs = <URN chars>
hexOctet  = hexDigit hexDigit
hexDigit  =
    "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" /
    "a" / "b" / "c" / "d" / "e" / "f" /
    "A" / "B" / "C" / "D" / "E" / "F"
```

3.1. spec-type and spec-value

The spec-type identifies the certificate specification type. The acceptable characters for spec-type are the same as an URI scheme name Section 3.1 of [RFC3986]. spec-types are compared case-insensitively. The spec-value identifies the certificate specification value. The acceptable characters for spec-value depend on the spec-type.

3.2. certattrs

A certspec can include attributes that are associated with the identified certificate. These attributes do NOT affect certificate identification; the syntax is intended primarily to convey certificate metadata such as attributes found in PKCS #9, PKCS #11, PKCS #12, and particular implementations of cryptographic libraries. The characters of certattrs can be any valid URN character from [RFC2141] (effectively, any URI character from [RFC3986] except "[" and "]"). This Internet-Draft does not further define certattrs.

4. Certificate Specifications

A certificate specification (certspec) unambiguously identifies a single certificate. A certificate has exactly one canonical certspec per applicable definition (irrespective of case changes or other differences that do not affect lexical equivalence). This Internet-

Draft provides four cryptographic hash-based specs, two value-based specs, and two data-based specs.

4.1. Cryptographic Hash-Based Specifications

A cryptographic hash of a certificate uniquely identifies that certificate. Such a hash may also be called a "certificate fingerprint". In all certspecs in this specification *or* derived from this specification, the hash is computed over the octets of the DER encoding of the certificate, namely, the Certificate type of Section 4.1 of [RFC5280]. The DER encoding includes tag and length octets, so the first octet is always 30h (the tag for SEQUENCE), and the second octet is never 80h (the octet for indefinite length).

In certspecs in this specification, the spec-value is the hexadecimal encoding of the hash value octets. For example, a 256-bit SHA-256 hash is represented by exactly 32 hex octets, or 64 hex characters. The following ABNF defines proper spec-values:

```
spec-value-sha-1    = 20hexOctet
spec-value-sha-256  = 32hexOctet
spec-value-SHA-384  = 48hexOctet
spec-value-SHA-512  = 64hexOctet
```

Lexical equivalence of two certspecs that have the same spec-type SHALL be determined by converting the hexadecimal spec-values to octets and comparing exact equivalence of the octets. A strict implementation would reject values that contain non-hex digits, such as spaces, tabs, or hyphens. However, a lenient implementation MAY ignore non-hex characters. In any event, lexical equivalence is determined by converting the hex to octets, and then comparing the octets. If there are too few or too many hex characters, a conforming implementation MUST reject the certspec.

Conforming implementations to this Internet-Draft MUST recognize these hash-based certspecs, unless security considerations dictate otherwise. Acceptable reasons for refusing to process a certspec include a) the local policy prohibits use of the hash, or b) the hash has known cryptographic weaknesses, such as a preimage attacks, which weaken the cryptographic uniqueness guarantees of the hash.

4.1.1. SHA-1

The spec-type is "SHA-1". The hash is computed using SHA-1 [SHS].

4.1.2. SHA-256

The spec-type is "SHA-256". The hash is computed using SHA-256 [SHS].

4.1.3. SHA-384

The spec-type is "SHA-384". The hash is computed using SHA-384 [SHS].

4.1.4. SHA-512

The spec-type is "SHA-512". The hash is computed using SHA-512 [SHS].

4.1.5. [Discussion]

[[DPl: Hashes could be grouped into the spec-type "fp", such as: urn:cert:fp:HASHNAME:HASHVALUE. In such a case, a registry, such as one maintained by IANA, could be used to establish valid HASHNAMEs. In the formulation above, three characters ("fp:") are omitted, and the ability to use "arbitrary" hashes is curtailed. A new hash can be used, but must be procured through the IETF consensus process, which is a higher barrier for establishing new hashes than a registry entry, but offer less oversight. It is believed that omitting "fp:" fosters interoperability because it is better to identify certificates by a small number of widely-recognized, standard algorithms, especially if human consumption or production are foreseen. --S.L.]]

4.2. Value-Based Specifications

A certificate may be identified reflexively, by its constituent octets. For small-to-medium certificates, identifying the certificate by embedding it in the certspec will be computationally efficient and resistant to denial-of-service attacks (by being always available).

The octets of a certificate are the octets of the DER encoding of the certificate, namely, the Certificate type of Section 4.1 of RFC 5280.

Lexical equivalence of two certspecs that are value-based SHALL be determined by converting the spec-value to certificate octets, and comparing the octets for strict equivalence. Accordingly, it is possible for a base64 and a hex certspec to be lexically equivalent to each other.

A conforming implementation MUST implement base64 and hex specs.

4.2.1. base64

The spec-type is "base64". The spec-value is the base64 encoding (Section 4 of [RFC4648]) of the certificate octets. Like the data: URL [RFC2397], the characters '+' and '/' refer to values 62 and 63, respectively. Additionally, if the length of certificate octets is not a multiple of 3, it is expected that one or two trailing equal signs '=' will be present.

+', '/', and '=' have no reserved meaning in this spec-type. While the URN syntax rules [RFC2141] state that '/' should not be used in unencoded form, in this specification, '/' MAY be present in unencoded form in the base64 spec-type. In any case, a conforming implementation MUST be able to process "%"-encoded characters.

While a strict implementation would reject non-base64 characters, a lenient implementation MAY ignore non-base64 characters, such as CR, LF, whitespace, or the absence of trailing '='. As a result, two certspecs that have the same base64-encoded data but different stray non-base64 characters MAY be judged lexically equivalent. Similarly, [RFC2141] requires that non-reserved characters (in this case, alphanumerics) must not be "%"-encoded, but a lenient implementation MAY decode these "%"-encoded characters anyway. This specification neither recommends nor discourages such leniency, but implementors should weigh the benefits and risks as discussed further in the Security Considerations section.

4.2.2. hex

The spec-type is "hex". The spec-value is the hexadecimal encoding (Section 8 of [RFC4648]) of the certificate octets. Whether an implementation should process "%"-encoded characters or non-hex characters is subject to the same considerations as the equivalent characters in the base64 spec.

4.3. Data-Based Specifications

A certificate may be identified by data contained within it. The following specs reflect the traditional reliance of PKIX [RFC5280] and CMS [RFC5652] on a certificate's issuer name and serial number, or a certificate's subject key identifier. These specs provide textual representations for these identifiers.

4.3.1. issuersn: Issuer Name and Serial Number

The spec-type is "issuersn". The spec-value is given by the following ABNF:

```
spec-value-issuersn = dn SEMI serialNumber
serialNumber        = 1*hexOctet
```

<dn> is defined in [RFC4516], which is based on <distinguishedName> from [RFC4514] with the addition of "%"-encoding. <SEMI> is defined in [RFC4512]. RFC 4514 no longer separates relative distinguished names (RDNs) by semicolons, as required by its predecessor, RFC 2253. Accordingly, ';' is used to separate the issuer's DN from the subject's serial number. If ';' is present in the dn, it MUST be preceded by a backslash '\', which MUST be "%"-encoded.

Care should be taken in escaping and "%"-encoding the relevant characters. In particular, "?" is permitted in a distinguishedName, but is RESERVED by this specification and [RFC2141]. Any question marks in distinguished names MUST be "%"-encoded when placed in the spec-value.

<serialNumber> is the hexadecimal encoding of the certificate's serial number, with the exact same (DER encoded) contents octets of a CertificateSerialNumber ::= INTEGER as specified in Section 4.1 of [RFC5280]. If the serial number hex octets are malformed, the certspec is invalid.

A conforming implementation SHOULD implement this issuersn spec. If the implementation implements it, the implementation MUST process serial numbers up to the same length as required by Section 4.1.2.2 of [RFC5280] (20 octets), and MUST process distinguished name strings as required by [RFC4514], including the table of minimum AttributeType name strings that MUST be recognized.

Lexical equivalence of two issuersn certspecs SHALL be determined by comparing the integer values of the serialNumbers for exact equivalence, and comparing the distinguished names for a match. Distinguished names match if they satisfy the name matching requirements of [RFC5280]. An implementation MAY use attribute matching rules that are more restrictive or comprehensive than [RFC5280], provided that the applied rules do not contradict the LDAP definitions of the attributes.

4.3.2. ski: Subject Key Identifier

The spec-type is "ski". The spec-value is given by the following ABNF:

```
spec-value-ski = keyIdentifier
keyIdentifier   = 1*hexOctet
```

<keyIdentifier> is the hexadecimal encoding of the certificate's

subject key identifier, which is recorded in the certificate's Subject Key Identifier extension (Section 4.2.1.2 of [RFC5280]). A certificate that lacks a subject key identifier cannot be identified using this spec.

Lexical equivalence of two ski certspecs SHALL be determined by converting the hexadecimal spec-values to octets and comparing the exact equivalence of the octets.

A conforming implementation MAY implement this ski spec.

5. Registration Template

Namespace ID:
cert

Registration Information:
Version: 1
Date: 2012-07-08

Declared registrant of the namespace:
IETF

Declaration of syntactic structures:
The structure of the Namespace Specific String is provided above.

Relevant ancillary documentation:
Certificates are defined by [RFC5280] and [X.509].

Identifier uniqueness considerations:
The spec type is assigned by IANA through the IETF consensus process, so this process guarantees uniqueness of these identifiers. The uniqueness of the spec value depends on the spec type. For specs that identify cryptographic hashes, the cryptographic hash algorithm itself guarantees uniqueness. For specs that identify certificates by value, the inclusion of the certificate in the URN itself guarantees uniqueness. For specs that identify certificates by certificate data, the resolver's database of certificates and implementation of certification path validation Section 6 of [RFC5280] ensure uniqueness.

Identifier persistence considerations:
A certificate is a permanent digital artifact, irrespective of its origin. As the assignment process records mathematical or existential facts about the certificate, such as one of its

cryptographic hashes, the binding between the URN and the certificate resource is permanent. Changing even one bit of the certificate will alter its URN, will make the certificate unusable, or both.

Process of identifiers assignment:

Generating a certspec (cert URN) does not require that a registration authority be contacted.

Process for identifier resolution:

This Internet Draft does not specify a resolution service for certspecs. However, resolving certificate references to actual certificates is a common practice with a wide number of offline and online implementations.

Rules for Lexical Equivalence:

Certspecs (cert URNs) are lexically equivalent if they both have the same spec type (compared case-insensitively) and the same space value, and therefore impliedly point to the same certificate.

Comparison of spec values depends on the rules of the spec. Although extensions may be appended to a certspec, these extensions are guaranteed not to affect lexical equivalence.

Certspecs are semantically equivalent if they both resolve to the same certificate.

Conformance with URN Syntax:

The character '?' is reserved for future extensions to this specification. The URN of this namespace conforms to URN Syntax [RFC2141] and Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Validation mechanism:

Each spec defines the validation mechanism for its respective value. It may be appreciated that validation of the URN is a completely different process from the Certification Path Validation Algorithm (Section 6 of [RFC5280]), which determines whether the *certificate* is valid.

Scope:

Global.

6. Use of certspec outside URN

certspec is useful wherever a system may need to refer to a

certificate by value or by reference. Some implementations may wish to refer to a certificate without enabling all of the expressive power (and security considerations) of URIs. Accordingly, this section provides a uniform method for using a certspec outside of a URN. Examples:

```
SHA-1:blf090a8e2d70353107454f9618347b18b321bfl  
issuersn:CN=Atlantis;2A
```

To use certspec outside of a URI (URN) context, the prefix "urn:cert:" MAY be omitted. All other lexical rules remain in effect, including "%"-encoding. Care should be taken to process '?' in particular, since '?' separates the certspec from appended attributes. A conforming implementation of raw certspecs MUST permit the prefix "urn:cert:" in addition to the raw certspec, which starts with the spec-type. This specification guarantees that the cert-type "urn" is RESERVED and will never be used. However, implementors must take note that a raw certspec is not a valid URI, because cert-types are not registered URI schemes and do not have the same semantics as a URI.

7. IANA Considerations

This document requests the assignment of formal URN namespace ID "cert".

This document requests the creation of a registry to record specs. New specs shall be ratified by the IETF consensus process.

8. Best Practices

When producing a hash-based certspec, the producer has a wide choice of hashes. Nevertheless, this Internet-Draft RECOMMENDS that SHA-1 or SHA-256 be used to foster the greatest interoperability and human recognition, provided that the Security Considerations are heeded.

9. Security Considerations

Digital certificates are important building blocks for authentication, integrity, authorization, and (occasionally) confidentiality services. Accordingly, identifying digital certificates incorrectly can have significant security ramifications.

When using specs that are cryptographic hashes (fingerprints) for lookups and comparisons, the cryptographic hash algorithm MUST be

implemented properly and SHOULD have no known attack vectors. The registration of a particular algorithm spec in this namespace does NOT mean that it is acceptable or safe for every usage, even though this Internet-Draft requires that a conforming implementation MUST implement certain specs.

When using by-value specs, the implementation MUST be prepared to process URNs of arbitrary length. As of this writing, useful certificates rarely exceed 10KB, and most implementations are concerned with keeping certificate sizes down rather than up (for example, to fit in a single TCP packet). However, a pathological or malicious certificate could easily exceed these metrics. If an URN resolver cannot process a URN's full length, it MUST reject the certspec.

When using specs that depend on certificate data, the implementation MUST be prepared to deal with multiple found certificates that contain the same certificate data, but are not the same certificate. In such a case, the implementation MUST segregate these certificates so that it only resolves the URN to certificates that it considers valid or trustworthy (as discussed further below). If, despite this segregation, multiple valid or trustworthy certificates match the certspec, the certspec MUST be rejected, because a certspec is meant to identify exactly one certificate (not a family of certificates).

Apart from the mechanics of certspecs (cert URNs), certificates should not be used unless they have passed the Certification Path Validation Algorithm (Section 6 of [RFC5280]), or another algorithm that provides some guarantee of validity. For example, if a certificate database contains a set of certificates that it considers inherently trustworthy, then the inclusion of a certificate in that set makes it trustworthy, regardless of the results of the Certification Path Validation Algorithm. Such a database is frequently used for "Root CA" lists.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.
- [RFC3406] Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.
- [RFC4514] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006.
- [RFC4516] Smith, M. and T. Howes, "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, June 2006.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [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.
- [SHS] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.

10.2. Informative References

- [P11] RSA Laboratories, "PKCS #11 v2.20: Cryptographic Token Interface Standard", PKCS #11, June 2004.
- [P12] RSA Laboratories, "PKCS #12 v1.0: Personal Information Exchange Syntax Standard", PKCS #12, June 1999.
- [P9] RSA Laboratories, "PKCS #9 v2.0: Selected Object Classes and Attribute Types", PKCS #9, February 2000.
- [RFC2253] Wahl, M., Kille, S., and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.
- [RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, August 1998.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[X.509] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, ISO Standard 9594-8, November 2008.

Author's Address

Sean Leonard
Penango, Inc.
1215 K Street
17th Floor
Sacramento, CA 95814
USA

Email: dev+ietf@seantek.com
URI: <http://www.penango.com/>

