Network Working Group
Internet-Draft
Obsoletes: RFC 6844 (if approved)
Intended status: Standards Track
Expires: August 27, 2018

P. Hallam-Baker
R. Stradling
Comodo Group, Inc
J. Hoffman-Andrews
Let's Encrypt

February 23, 2018

DNS Certification Authority Authorization (CAA) Resource Record
draft-hoffman-andrews-caa-simplification-03

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. This document defines the syntax of the CAA record and rules for processing CAA records by certificate issuers.

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 https://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 August 27, 2018.

Copyright Notice

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

The Certification Authority Authorization (CAA) DNS Resource Record allows a DNS domain name holder to specify the Certification Authorities (CAs) authorized to issue certificates for that domain. Publication of CAA Resource Records allows a public Certification Authority 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) [RFC6698], 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 an 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 typically be 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 Certification 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. Definitions

2.1. 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].

2.2. 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.

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: A DNS Domain Name.

Domain Name: A DNS Domain Name as specified in [RFC1034].

Domain Name System (DNS): The Internet naming system specified in [RFC1034] and [RFC1035].

DNS Security (DNSSEC): Extensions to the DNS that provide authentication services as specified in [RFC4033], [RFC4034], [RFC4035], [RFC5155], and revisions.

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

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 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 particular entry in the DNS including the owner name, class, type, time to live, and data, as defined in [RFC1034] and [RFC2181].

Resource Record Set (RRSet): A set of Resource Records of a particular owner name, class, and type. The time to live on all RRs with an RRSet is always the same, but the data may be different among RRs in the RRSet.

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.

3. 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 to '1', indicates that the corresponding property 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 relevant CAA Resource Record set contains unknown property tags that have the Critical bit set.

The following property tags are defined:

issue <Issuer Domain Name> [; <name>=<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.

issuewild <Issuer Domain Name> [; <name>=<value>]* : The issuewild 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 wildcard 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 Incident Object Description Exchange Format (IODEF) format is used [RFC5070].

The following example is a DNS zone file (see [RFC1035]) that informs CAs that certificates are not to be issued except by the holder of the domain name 'ca.example.net' or an authorized agent thereof. This 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 of additional parameters is a sequence of name-value pairs as defined in Section 5.2. The semantics of such parameters is left to site policy and is outside the scope of this document.

The critical flag is intended to permit future versions of CAA to introduce new semantics that MUST be understood for correct processing of the record, preventing conforming CAs that do not recognize the new semantics 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 at certificate issue. 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.

4. Certification Authority Processing

Before issuing a certificate, a compliant CA MUST check for publication of a relevant CAA Resource Record set. If such a record set exists, a CA MUST NOT issue a certificate unless the CA determines that either (1) the certificate request is consistent with the applicable CAA Resource Record set or (2) an exception specified in the relevant Certificate Policy or Certification Practices Statement applies.

A certificate request MAY specify more than one domain name and MAY specify wildcard domains. Issuers MUST verify authorization for all the domains and wildcard domains specified in the request.

The search for a CAA record climbs the DNS name tree from the specified label up to but not including the DNS root '.' until CAA records are found.

Given a request for a specific domain name X, or a request for a wildcard domain name *.X, the relevant record set RelevantCAASet(X) is determined as follows:

Let CAA(X) be the record set returned by performing a CAA record query for the domain name X, according to the lookup algorithm specified in RFC 1034 section 4.3.2 (in particular chasing aliases). Let Parent(X) be the domain name produced by removing the leftmost label of X.

```
RelevantCAASet(domain):
  for domain is not ".":
    if CAA(domain) is not Empty:
      return CAA(domain)
    domain = Parent(domain)
  return Empty
```

For example, processing CAA for the domain name "X.Y.Z" where there are no CAA records at any level in the tree RelevantCAASet would have the following steps:

```
CAA("X.Y.Z.") = Empty; domain = Parent("X.Y.Z.") = "Y.Z."
CAA("Y.Z.")   = Empty; domain = Parent("Y.Z.")   = "Z."
CAA("Z.")     = Empty; domain = Parent("Z.")     = "."
return Empty
```

Processing CAA for the domain name "A.B.C" where there is a CAA record "issue example.com" at "B.C" would terminate early upon finding the CAA record:

```
CAA("A.B.C.") = Empty; domain = Parent("A.B.C.") = "B.C."
CAA("B.C.")   = "issue example.com"
return "issue example.com"
```

4.1. 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 relevant CAA Resource Record set, irrespective of whether the corresponding DNS records are signed.

DNSSEC provides a proof of non-existence for both DNS domains and RR set within domains. DNSSEC verification thus enables an issuer to determine if the answer to a CAA record query is empty because the RR set is empty or if it is non-empty but the response has been suppressed.

Use of DNSSEC allows an issuer to acquire and archive a 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.

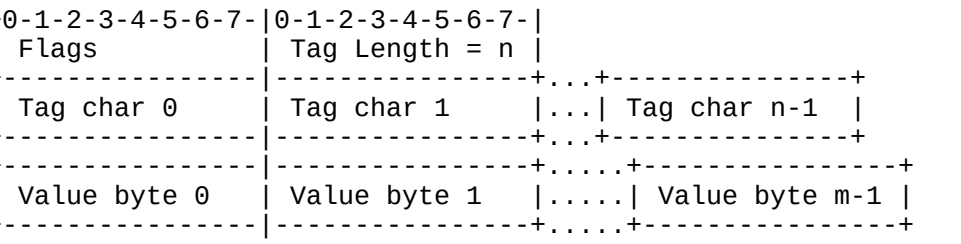
5. Mechanism

5.1. 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:



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 field:

Bit 0, Issuer Critical Flag: If the value is set to '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 [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 US-ASCII characters.

Tag values MAY contain US-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) US-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.

5.1.1. Canonical Presentation Format

The canonical presentation format of the CAA record is:

CAA <flags> <tag> <value>

Where:

Flags: Is an unsigned integer between 0 and 255.

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

Value: Is the <character-string> encoding of the value field as specified in [RFC1035], Section 5.1.

5.2. 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]).

```
issuevalue = space [domain] space [";" *(space parameter) space]

domain = label *("." label) label = (ALPHA / DIGIT) *( *("-") (ALPHA
/ DIGIT))

space = *(SP / HTAB)

parameter = tag "=" value

tag = 1*(ALPHA / DIGIT)

value = *VCHAR
```

For consistency with other aspects of DNS administration, domain name values are specified in letter-digit-hyphen Label (LDH-Label) form.

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 polices, billed to certain accounts, or issued under specific trust anchors.

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

5.3. CAA issuewild Property

The issuewild property has the same syntax and semantics as the issue property except that issuewild properties only grant authorization to issue certificates that specify a wildcard domain and issuewild properties take precedence over issue properties when specified. Specifically:

issuewild properties MUST be ignored when processing a request for a domain that is not a wildcard domain.

If at least one issuewild property is specified in the relevant CAA record set, all issue properties MUST be ignored when processing a request for a domain that is a wildcard domain.

5.4. 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].

6. 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 [RFC6698] describes a mechanism for avoiding reliance on mis-issued certificates.

6.1. 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.

6.2. 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.

6.3. 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 an issuer that was not authorized to issue for that domain name.

Where possible, issuers SHOULD perform DNSSEC validation to detect missing or modified CAA record sets.

In cases where DNSSEC is not deployed in a corresponding domain, an issuer 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.

6.4. Denial of Service

Introduction of a malformed or malicious CAA RR could in theory enable a Denial-of-Service (DoS) 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 DoS 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 DoS attacks to any meaningful degree.

6.5. Abuse of the Critical Flag

A Certification Authority could make use of the critical flag to trick customers into publishing records that 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. The customer will thus discover that they had been deceived.

7. Deployment Considerations

7.1. Blocked Queries or Responses

Some middleboxes, in particular anti-DDoS appliances, may be configured to drop DNS packets of unknown types, or may start dropping such packets when they consider themselves under attack. This generally manifests as a timed-out DNS query, or a SERVFAIL at a local recursive resolver.

For deployability of CAA and future DNS record types, middleboxes SHOULD block DNS packets by volume and size rather than by query type.

7.2. Rejected Queries and Malformed Responses

Some authoritative nameservers respond with REJECTED or NOTIMP when queried for a resource record type they do not recognize. At least one authoritative resolver produces a malformed response (with the QR bit set to 0) when queried for unknown resource record types. Per RFC 1034, the correct response for unknown resource record types is NOERROR.

7.3. Bogus DNSSEC Responses

CAA queries are unusual in DNS, because a signed, empty response is different from a bogus response. A signed, empty response indicates that there is definitely no CAA policy set at a given label. A bogus response may mean either a misconfigured zone, or an attacker tampering with records. DNSSEC implementations may have bugs with signatures on empty responses that go unnoticed, because for more common resource record types like A and AAAA, the difference to an end user between empty and bogus is irrelevant; they both mean a site is unavailable.

In particular, at least two authoritative resolvers that implement live signing had bugs when returning empty resource record sets for DNSSEC-signed zones, in combination with mixed-case queries. Mixed-case queries, also known as DNS 0x20, are used by some recursive resolvers to increase resilience against DNS poisoning attacks. DNSSEC-signing authoritative resolvers are expected to copy the same capitalization from the query into their ANSWER section, but sign the response as if they had use all lowercase. In particular, PowerDNS versions prior to 4.0.4 had this bug.

8. IANA Considerations

8.1. Registration of the CAA Resource Record Type

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	[RFC6844]

8.2. Certification Authority Restriction Properties

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

Tag	Meaning	Reference
issue	Authorization Entry by Domain	[RFC6844]
issuewild	Authorization Entry by Wildcard Domain	[RFC6844]
iodef	Report incident by IODEF report	[RFC6844]
auth	Reserved	Unpublished draft
path	Reserved	Unpublished draft
policy	Reserved	Unpublished draft

Addition of tag identifiers requires a public specification and Expert Review as set out in [RFC6195], Section 3.1.1.

The tag space is designed to be sufficiently large that exhausting the possible tag space need not be a concern. The scope of Expert Review SHOULD be limited to the question of whether the specification provided is sufficiently clear to permit implementation and to avoid unnecessary duplication of functionality.

8.3. Certification Authority Restriction Flags

IANA has created the "Certification Authority Restriction Flags" registry with the following initial values:

Flag	Meaning	Reference
0	Issuer Critical Flag	[RFC6844]
1-7	Reserved>	[RFC6844]

Assignment of new flags follows the RFC Required policy set out in [RFC5226], Section 4.1.

9. 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, James Manager, Chris Palmer, Scott Schmit, Sean Turner, and Ben Wilson.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC5070] Danyliw, R., Meijer, J., and Y. Demchenko, "The Incident Object Description Exchange Format", RFC 5070, DOI 10.17487/RFC5070, December 2007, <https://www.rfc-editor.org/info/rfc5070>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6195] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", RFC 6195, DOI 10.17487/RFC6195, March 2011, <https://www.rfc-editor.org/info/rfc6195>.

[RFC6546] Trammell, B., "Transport of Real-time Inter-network Defense (RID) Messages over HTTP/TLS", RFC 6546, DOI 10.17487/RFC6546, April 2012, <https://www.rfc-editor.org/info/rfc6546>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6844] Hallam-Baker, P. and R. Stradling, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 6844, DOI 10.17487/RFC6844, January 2013, <https://www.rfc-editor.org/info/rfc6844>.

10.2. 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, DOI 10.17487/RFC3647, November 2003, <https://www.rfc-editor.org/info/rfc3647>.

Authors' Addresses

Phillip Hallam-Baker
Comodo Group, Inc

Email: philliph@comodo.com

Rob Stradling
Comodo Group, Inc

Email: rob.stradling@comodo.com

Jacob Hoffman-Andrews
Let's Encrypt

Email: jsha@letsencrypt.org

INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: 3 October 2018

R. Housley
Vigil Security

3 April 2018

Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS) <draft-housley-cms-mix-with-psk-04.txt>

Abstract

The invention of a large-scale quantum computer would pose a serious challenge for the cryptographic algorithms that are widely deployed today. The Cryptographic Message Syntax (CMS) supports key transport and key agreement algorithms that could be broken by the invention of such a quantum computer. By storing communications that are protected with the CMS today, someone could decrypt them in the future when a large-scale quantum computer becomes available. Once quantum-secure key management algorithms are available, the CMS will be extended to support them, if current syntax the does not accommodated them. In the near-term, this document describes a mechanism to protect today's communication from the future invention of a large-scale quantum computer by mixing the output of key transport and key agreement algorithms with a pre-shared key.

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."

Copyright Notice

Table of Contents

1. Introduction

The invention of a large-scale quantum computer would pose a serious challenge for the cryptographic algorithms that are widely deployed today. It is an open question whether or not it is feasible to build a large-scale quantum computer, and if so, when that might happen. However, if such a quantum computer is invented, many of the cryptographic algorithms and the security protocols that use them would become vulnerable.

The Cryptographic Message Syntax (CMS) [RFC5652][RFC5803] supports key transport and key agreement algorithms that could be broken by the invention of a large-scale quantum computer [C2PQ]. These algorithms include RSA [RFC4055], Diffie-Hellman [RFC2631], and Elliptic Curve Diffie-Hellman. As a result, an adversary that stores CMS-protected communications today, could decrypt those communications in the future when a large-scale quantum computer becomes available.

Once quantum-secure key management algorithms are available, the CMS

will be extended to support them, if current syntax the does not accommodated them.

In the near-term, this document describes a mechanism to protect today's communication from the future invention of a large-scale quantum computer by mixing the output of existing key transport and key agreement algorithms with a pre-shared key (PSK). Secure communication can be achieved today by mixing with a strong PSK with the output of an existing key transport algorithm, like RSA, or an existing key agreement algorithm, like Diffie-Hellman [RFC2631] or Elliptic Curve Diffie-Hellman [RFC5753]. A security solution that is believed to be quantum resistant can be achieved by using a PSK with sufficient entropy along with a quantum resistant key derivation function (KDF), like HKDF [RFC5869], and a quantum resistant encryption algorithm, like 256-bit AES [AES]. In this way, today's CMS-protected communication can be invulnerable to an attacker with a large-scale quantum computer.

Note that the CMS also supports key management techniques based on symmetric key-encryption keys and passwords, but they are not discussed in this document because they are already quantum resistant. The symmetric key-encryption key technique is quantum resistant when used with an adequate key size. The password technique is quantum resistant when used with a quantum-resistant key derivation function and a sufficiently large password.

1.1. Terminology

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. ASN.1

CMS values are generated using ASN.1 [X680], which uses the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X690].

1.3. Version Numbers

The major data structures include a version number as the first item in the data structure. The version number is intended to avoid ASN.1 decode errors. Some implementations do not check the version number prior to attempting a decode, and then if a decode error occurs, the version number is checked as part of the error handling routine. This is a reasonable approach; it places error processing outside of

the fast path. This approach is also forgiving when an incorrect version number is used by the sender.

Whenever the structure is updated, a higher version number will be assigned. However, to ensure maximum interoperability, the higher version number is only used when the new syntax feature is employed. That is, the lowest version number that supports the generated syntax is used.

2. Overview

The CMS enveloped-data content type [RFC5652] and the CMS authenticated-enveloped-data content type [RFC5083] support both key transport and key agreement public-key algorithms to establish the key used to encrypt the content. In both cases, the sender randomly generates the key, and then all recipients obtain that key. For enveloped-data, a content-encryption key is established. For authenticated-enveloped-data, a content-authenticated-encryption key is established. All recipients use the sender-generated key for decryption.

This specification defines two quantum-resistant ways to establish these keys. In both cases, a PSK MUST be distributed to the sender and all of the recipients by some out-of-band means that does not make it vulnerable to the future invention of a large-scale quantum computer, and an identifier MUST be assigned to the PSK.

The content-encryption key or content-authenticated-encryption key is established by following these steps:

1. The content-encryption key or the content-authenticated-encryption key is generated at random.

2. The key-derivation key is generated at random.

3. The key-encryption key is established for each recipient. The details depend on the key management algorithm used:

   key transport: the key-derivation key is encrypted in the recipient’s public key, then the key derivation function (KDF) is used to mix the pre-shared key (PSK) and the key-derivation key to produce the key-encryption key; or

   key agreement: the recipient’s public key and the sender’s private key are used to generate a pairwise symmetric key, then the key derivation function (KDF) is used to mix the pre-shared key (PSK) and the pairwise symmetric key to produce the key-encryption key.

4. The key-encryption key is used to encrypt the content-encryption key or content-authenticated-encryption key.

As specified in Section 6.2.5 of [RFC5652], recipient information for additional key management techniques are represented in the OtherRecipientInfo type. Two key management techniques are specified in this document. Each of these is identified by a unique ASN.1 object identifier.

The first key management technique, called keyTransPSK, see Section 3, uses a key transport algorithm to transfer the key-derivation key from the sender to the recipient, and then the key-derivation key is mixed with the PSK using a KDF. The output of the KDF is the key-encryption key for the encryption of the content-encryption key or content-authenticated-encryption key.

The second key management technique, called keyAgreePSK, see Section 4, uses a key agreement algorithm to establish a pairwise key-encryption key, which is used to encrypt the key-derivation key, and the then key-derivation key is mixed with the PSK using a KDF. The output of the KDF is the key-encryption key for the encryption of the content-encryption key or content-authenticated-encryption key.

3. KeyTransPSKRecipientInfo

Per-recipient information using keyTransPSK is represented in the KeyTransPSKRecipientInfo type, and its use is indicated by the id-ori-keyTransPSK object identifier. Each instance of KeyTransPSKRecipientInfo establishes the content-encryption key or content-authenticated-encryption key for one or more recipients that have access to the same PSK.

The id-ori-keyTransPSK object identifier is:

```
id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }

id-ori-keyTransPSK OBJECT IDENTIFIER ::= { id-ori 1 }
```

The KeyTransPSKRecipientInfo type is:

```
KeyTransPSKRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  pskid PreSharedKeyIdentifier,
  kdfAlgorithm KeyDerivationAlgorithmIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  ktris KeyTransRecipientInfos,
  encryptedKey EncryptedKey }

PreSharedKeyIdentifier ::= OCTET STRING

KeyTransRecipientInfos ::= SEQUENCE OF KeyTransRecipientInfo
```

The fields of the KeyTransPSKRecipientInfo type have the following meanings:

version is the syntax version number. The version MUST be 0. The CMSVersion type is described in Section 10.2.5 of [RFC5652].

pskid is the identifier of the PSK used by the sender. The identifier is an OCTET STRING, and it need not be human readable.

kdfAlgorithm identifies the key-derivation algorithm, and any associated parameters, used by the sender to mix the key-derivation key and the PSK to generate the key-encryption key. The KeyDerivationAlgorithmIdentifier is described in Section 10.1.6 of [RFC5652].

keyEncryptionAlgorithm identifies a key-encryption algorithm used to encrypt the content-encryption key. The KeyEncryptionAlgorithmIdentifier is described in Section 10.1.3 of [RFC5652].

ktris contains one KeyTransRecipientInfo type for each recipient; it uses a key transport algorithm to establish the key-derivation key. KeyTransRecipientInfo is described in Section 6.2.1 of [RFC5652].

encryptedKey is the result of encrypting the content-encryption key or the content-authenticated-encryption key with the key-encryption key. EncryptedKey is an OCTET STRING.

4. KeyAgreePSKRecipientInfo

Per-recipient information using keyAgreePSK is represented in the KeyAgreePSKRecipientInfo type, and its use is indicated by the id-ori-keyAgreePSK object identifier. Each instance of KeyAgreePSKRecipientInfo establishes the content-encryption key or content-authenticated-encryption key for one or more recipients that have access to the same PSK.

The id-ori-keyAgreePSK object identifier is:

```
id-ori-keyAgreePSK OBJECT IDENTIFIER ::= { id-ori 2 }
```

The KeyAgreePSKRecipientInfo type is:

```
KeyAgreePSKRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  pskid PreSharedKeyIdentifier,
  originator [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  kdfAlgorithm KeyDerivationAlgorithmIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys }
```

The fields of the KeyAgreePSKRecipientInfo type have the following meanings:

version is the syntax version number. The version MUST be 0. The CMSVersion type is described in Section 10.2.5 of [RFC5652].

pskid is the identifier of the PSK used by the sender. The identifier is an OCTET STRING, and it need not be human readable.

originator is a CHOICE with three alternatives specifying the sender's key agreement public key. Implementations MUST support all three alternatives for specifying the sender's public key. The sender uses the corresponding private key and the recipient's public key to generate a pairwise key. A key derivation function (KDF) is used to mix the PSK and the pairwise key to produce a key-encryption key. The OriginatorIdentifierOrKey type is described in Section 6.2.2 of [RFC5652].

ukm is optional. With some key agreement algorithms, the sender provides a User Keying Material (UKM) to ensure that a different key is generated each time the same two parties generate a pairwise key. Implementations MUST accept a KeyAgreePSKRecipientInfo SEQUENCE that includes a ukm field. Implementations that do not support key agreement algorithms that

make use of UKMs MUST gracefully handle the presence of UKMs. The UserKeyingMaterial type is described in Section 10.2.6 of [RFC5652].

kdfAlgorithm identifies the key-derivation algorithm, and any associated parameters, used by the sender to mix the pairwise key and the PSK. The KeyDerivationAlgorithmIdentifier is described in Section 10.1.6 of [RFC5652].

keyEncryptionAlgorithm identifies a key-encryption algorithm used to encrypt the content-encryption key or the content-authenticated-encryption key. The KeyEncryptionAlgorithmIdentifier type is described in Section 10.1.3 of [RFC5652].

recipientEncryptedKeys includes a recipient identifier and encrypted key for one or more recipients. The KeyAgreeRecipientIdentifier is a CHOICE with two alternatives specifying the recipient's certificate, and thereby the recipient's public key, that was used by the sender to generate a pairwise key. The encryptedKey is the result of encrypting the content-encryption key or the content-authenticated-encryption key with the key-encryption key. EncryptedKey is an OCTET STRING. The RecipientEncryptedKeys type is defined in Section 6.2.2 of [RFC5652].

5. ASN.1 Module

This section contains the ASN.1 module for the two key management techniques defined in this document. This module imports types from other ASN.1 modules that are defined in [RFC5911] and [RFC5912].

```
CMSORIforPSK-2017
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-cms-ori-psk-2017(TBD0) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS All

IMPORTS

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

```
   OTHER-RECIPIENT, OtherRecipientInfo, CMSVersion,
   KeyTransRecipientInfo, OriginatorIdentifierOrKey,
   UserKeyingMaterial, RecipientEncryptedKeys, EncryptedKey,
   KeyDerivationAlgorithmIdentifier, KeyEncryptionAlgorithmIdentifier
     FROM CryptographicMessageSyntax-2009  -- [RFC5911]
       { iso(1) member-body(2) us(840) rsadsi(113549)
         pkcs(1) pkcs-9(9) smime(16) modules(0)
         id-mod-cms-2004-02(41) } ;

   --
   -- OtherRecipientInfo Types (ori-)
   --

   SupportedOtherRecipInfo OTHER-RECIPIENT ::= {
     ori-keyTransPSK |
     ori-keyAgreePSK,
     ... }

   --
   -- Key Transport with Pre-Shared Key
   --

   ori-keyTransPSK OTHER-RECIPIENT ::= {
     KeyTransPSKRecipientInfo IDENTIFIED BY id-ori-keyTransPSK }

   id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }

   id-ori-keyTransPSK OBJECT IDENTIFIER ::= { id-ori 1 }

   KeyTransPSKRecipientInfo ::= SEQUENCE {
     version CMSVersion,  -- always set to 0
     pskid PreSharedKeyIdentifier,
     kdfAlgorithm KeyDerivationAlgorithmIdentifier,
     keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
     ktris KeyTransRecipientInfos,
     encryptedKey EncryptedKey }

   PreSharedKeyIdentifier ::= OCTET STRING

   KeyTransRecipientInfos ::= SEQUENCE OF KeyTransRecipientInfo
```

```
--
-- Key Agreement with Pre-Shared Key
--

ori-keyAgreePSK ORI-TYPE ::= {
  KeyAgreePSKRecipientInfo IDENTIFIED BY id-ori-keyAgreePSK }

id-ori-keyAgreePSK OBJECT IDENTIFIER ::= { id-ori 2 }

KeyAgreePSKRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  pskid PreSharedKeyIdentifier,
  originator [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  kdfAlgorithm KeyDerivationAlgorithmIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys }

END
```

6. Security Considerations

Implementations must protect the pre-shared key (PSK), key transport private key, the agreement private key, the key-derivation key, and the key-encryption key. Compromise of the PSK will make the encrypted content vulnerable to the future invention of a large-scale quantum computer. Compromise of the PSK and either the key transport private key or the agreement private key may result in the disclosure of all contents protected with that combination of keying material. Compromise of the PSK and the key-derivation key may result in disclosure of all contents protected with that combination of keying material. Compromise of the key-encryption key may result in the disclosure of all content-encryption keys or content-authenticated-encryption keys that were protected with that keying materail, which in turn may result in the disclosure of the content.

A large-scale quantum computer will essentially negate the security provided by the key transport algorithm or the key agreement algorithm, which means that the attacker with a large-scale quantum computer can discover the key-derivation key. In addition a large-scale quantum computer effectively cuts the security provided by a symmetric key algorithm in half. Therefore, the PSK needs at least 256 bits of entropy to provide 128 bits of security. To match that same level of security, the key derivation function needs to be quantum-resistant and produce a key-encryption key that is at least 256 bits in length. Similarly, the content-encryption key or content-authenticated-encryption key needs to be at least 256 bits in length.

Implementations must randomly generate key-derivation keys as well as the content-encryption keys or content-authenticated-encryption keys. Also, the generation of public/private key pairs for the key transport and key agreement algorithms rely on a random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area.

When using a PSK with a key transport or a key agreement algorithm, a key-encryption key is produced to encrypt the content-encryption key or content-authenticated-encryption key. If the key-encryption algorithm is different than the algorithm used to protect the content, then the effective security is determined by the weaker of the two algorithms. If, for example, content is encrypted with 256-bit AES, and the key is wrapped with 128-bit AES, then at most 128 bits of protection is provided. Implementers must ensure that the key-encryption algorithm is as strong or stronger than the content-encryption algorithm or content-authenticated-encryption algorithm.

Implementers SHOULD NOT mix quantum-resistant key management algorithms with their non-quantum-resistant counterparts. For example, the same content should not be protected with KeyTransRecipientInfo and KeyTransPSKRecipientInfo. Likewise, the same content should not be protected with KeyAgreeRecipientInfo and KeyAgreePSKRecipientInfo. Doing so would make the content vulnerable to the future invention of a large-scale quantum computer.

Implementers should not send the same content in different messages, one using a quantum-resistant key management algorithm and the other using a non-quantum-resistant key management algorithm, even if the content-encryption key is generated independently. Doing so may allow an eavesdropper to correlate the messages, making the content vulnerable to the future invention of a large-scale quantum computer.

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptoanalysis techniques are developed and computing performance improves, the work factor to break a particular cryptographic algorithm will be reduced. Therefore, cryptographic algorithm implementations should be modular, allowing new algorithms to be readily inserted. That is, implementors should be prepared for the set of supported algorithms to change over time.

7. IANA Considerations

One object identifier for the ASN.1 module in the Section 5 was assigned in the SMI Security for S/MIME Module Identifiers (1.2.840.113549.1.9.16.0) [IANA-MOD] registry:

```
id-mod-cms-ori-psk-2017 OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
   pkcs-9(9) smime(16) mod(0) TBD0 }
```

One object identifier for an arc to assign Other Recipient Info Identifiers was assigned in the SMI Security for S/MIME Mail Security (1.2.840.113549.1.9.16) [IANA-SMIME] registry:

```
id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }
```

This assignment created the new SMI Security for Other Recipient Info Identifiers (1.2.840.113549.1.9.16.TBD1) [IANA-ORI] registry with the following two entries with references to this document:

```
id-ori-keyTransPSK OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
   pkcs-9(9) smime(16) id-ori(TBD1) 1 }

id-ori-keyAgreePSK OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
   pkcs-9(9) smime(16) id-ori(TBD1) 2 }
```

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, November 2007.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[X680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, 2015.

[X690] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 2015.

9. Informative References

[AES] National Institute of Standards and Technology, FIPS Pub 197: Advanced Encryption Standard (AES), 26 November 2001.

[C2PQ] Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", work-in-progress, draft-hoffman-c2pq-03, February 2018.

[IANA-MOD] https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#security-smime-0.

[IANA-SMIME] https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#security-smime.

[IANA-ORI] https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#security-smime-13.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC3560] Housley, R., "Use of the RSAES-OAEP Key Transport Algorithm in Cryptographic Message Syntax (CMS)", RFC 3560, July 2003.

[RFC4086] D. Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.

[RFC5753] Turner, S., and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, January 2010.

[RFC5869] Krawczyk, H., and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.

[RFC5911] Hoffman, P., and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, June 2010.

[RFC5912] Hoffman, P., and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)" RFC 5912, June 2010.

Acknowledgements

Many thanks to Burt Kaliski, Panos Kampanakis, Jim Schaad, Sean Turner, and Daniel Van Geest for their review and insightful comments. They have greatly improved the design and implementation guidance.

Author’s Address

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
EMail: housley@vigilsec.com

LAMPS WG
Internet-Draft
Updates: 3370 (if approved)
Intended status: Standards Track
Expires: March 19, 2020

P. Kampanakis
Cisco Systems
Q. Dang
NIST

September 16, 2019

Use of the SHAKE One-way Hash Functions in the Cryptographic Message Syntax (CMS)
draft-ietf-lamps-cms-shakes-18

Abstract

This document updates the "Cryptographic Message Syntax Algorithms" (RFC3370) and describes the conventions for using the SHAKE family of hash functions in the Cryptographic Message Syntax as one-way hash functions with the RSA Probabilistic signature and ECDSA signature algorithms. The conventions for the associated signer public keys in CMS are also described.

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 https://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 March 19, 2020.

Copyright Notice

Table of Contents

1. Change Log

[EDNOTE: Remove this section before publication.]

- draft-ietf-lamps-cms-shake-18:
  - Minor ASN.1 changes.
- draft-ietf-lamps-cms-shake-17:
  - Minor updates for EDNOTE accuracy.
- draft-ietf-lamps-cms-shake-16:
  - Minor nits.
  - Using bytes instead of bits for consistency.
- draft-ietf-lamps-cms-shake-15:
  - Minor editorial nits.

- draft-ietf-lamps-cms-shake-14:
  - Fixing error with incorrect preimage resistance bits for SHA128 and SHA256.
- draft-ietf-lamps-cms-shake-13:
  - Addressing comments from Dan M.'s secdir review.
  - Addressing comment from Scott B.'s opsdir review about references in the abstract.
- draft-ietf-lamps-cms-shake-12:
  - Nits identified by Roman, Barry L. in ballot position review.
- draft-ietf-lamps-cms-shake-11:
  - Minor nits.
  - Nits identified by Roman in AD Review.
- draft-ietf-lamps-cms-shake-10:
  - Updated IANA considerations section to request for OID assignments.
- draft-ietf-lamps-cms-shake-09:
  - Fixed minor text nit.
  - Updates in Sec Considerations section.
- draft-ietf-lamps-cms-shake-08:
  - id-shake128-len and id-shake256-len were replaced with id-sha128 with 32 bytes output length and id-shake256 with 64 bytes output length.
  - Fixed a discrepancy between section 3 and 4.4 about the KMAC OIDs that have parameters as optional.
- draft-ietf-lamps-cms-shake-07:
  - Small nit from Russ while in WGLC.
- draft-ietf-lamps-cms-shake-06:

    * Incorporated Eric's suggestion from WGLC.

- draft-ietf-lamps-cms-shake-05:

    * Added informative references.

    * Updated ASN.1 so it compiles.

    * Updated IANA considerations.

- draft-ietf-lamps-cms-shake-04:

    * Added RFC8174 reference and text.

    * Explicitly explained why RSASSA-PSS-params are omitted in section 4.2.1.

    * Simplified Public Keys section by removing redundant info from RFCs.

- draft-ietf-lamps-cms-shake-03:

    * Removed paragraph suggesting KMAC to be used in generating k in Deterministic ECDSA. That should be RFC6979-bis.

    * Removed paragraph from Security Considerations that talks about randomness of k because we are using deterministic ECDSA.

    * Completed ASN.1 module and fixed KMAC ASN.1 based on Jim's feedback.

    * Text fixes.

- draft-ietf-lamps-cms-shake-02:

    * Updates based on suggestions and clarifications by Jim.

    * Started ASN.1 module.

- draft-ietf-lamps-cms-shake-01:

    * Significant reorganization of the sections to simplify the introduction, the new OIDs and their use in CMS.

    * Added new OIDs for RSASSA-PSS that hardcodes hash, salt and MGF, according the WG consensus.

    * Updated Public Key section to use the new RSASSA-PSS OIDs and clarify the algorithm identifier usage.

    * Removed the no longer used SHAKE OIDs from section 3.1.

  o draft-ietf-lamps-cms-shake-00:

    * Various updates to title and section names.

    * Content changes filling in text and references.

  o draft-dang-lamps-cms-shakes-hash-00:

    * Initial version

2. Introduction

The "Cryptographic Message Syntax (CMS)" [RFC5652] is used to digitally sign, digest, authenticate, or encrypt arbitrary message contents. "Cryptographic Message Syntax (CMS) Algorithms" [RFC3370] defines the use of common cryptographic algorithms with CMS. This specification updates RFC3370 and describes the use of the SHAKE128 and SHAKE256 specified in [SHA3] as new hash functions in CMS. In addition, it describes the use of these functions with the RSASSA-PSS signature algorithm [RFC8017] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [X9.62] with the CMS signed-data content type.

In the SHA-3 family, two extendable-output functions (SHAKEs), SHAKE128 and SHAKE256, are defined. Four other hash function instances, SHA3-224, SHA3-256, SHA3-384, and SHA3-512, are also defined but are out of scope for this document. A SHAKE is a variable length hash function defined as SHAKE(M, d) where the output is a d-bits-long digest of message M. The corresponding collision and second-preimage-resistance strengths for SHAKE128 are min(d/2,128) and min(d,128) bits, respectively (Appendix A.1 [SHA3]). And the corresponding collision and second-preimage-resistance strengths for SHAKE256 are min(d/2,256) and min(d,256) bits, respectively. In this specification we use d=256 (for SHAKE128) and d=512 (for SHAKE256).

A SHAKE can be used in CMS as the message digest function (to hash the message to be signed) in RSASSA-PSS and ECDSA, message authentication code and as the mask generation function (MGF) in RSASSA-PSS. This specification describes the identifiers for SHAKEs to be used in CMS and their meaning.

2.1. Terminology

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Identifiers

This section identifies eight new object identifiers (OIDs) for using SHAKE128 and SHAKE256 in CMS.

Two object identifiers for SHAKE128 and SHAKE256 hash functions are defined in [shake-nist-oids] and we include them here for convenience.

```
id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
     country(16) us(840) organization(1) gov(101) csor(3)
     nistAlgorithm(4) 2 11 }

id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
     country(16) us(840) organization(1) gov(101) csor(3)
     nistAlgorithm(4) 2 12 }
```

In this specification, when using the id-shake128 or id-shake256 algorithm identifiers, the parameters MUST be absent. That is, the identifier SHALL be a SEQUENCE of one component, the OID.

[I-D.ietf-lamps-pkix-shake] [EDNOTE: Update reference with the RFC when it is published.] defines two identifiers for RSASSA-PSS signatures using SHAKEs which we include here for convenience.

```
id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6) 30 }

id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6) 31 }
```

The same RSASSA-PSS algorithm identifiers can be used for identifying public keys and signatures.

[I-D.ietf-lamps-pkix-shake] [EDNOTE: Update reference with the RFC when it is published.] also defines two algorithm identifiers of ECDSA signatures using SHAKEs which we include here for convenience.

```
id-ecdsa-with-shake128 OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6) 32 }

id-ecdsa-with-shake256 OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6) 33 }
```

The parameters for the four RSASSA-PSS and ECDSA identifiers MUST be absent. That is, each identifier SHALL be a SEQUENCE of one component, the OID.

Two object identifiers for KMACs using SHAKE128 and SHAKE256 as defined in by the National Institute of Standards and Technology (NIST) in [shake-nist-oids] and we include them here for convenience.

```
id-KmacWithSHAKE128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 2 19 }

id-KmacWithSHAKE256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 2 20 }
```

The parameters for id-KmacWithSHAKE128 and id-KmacWithSHAKE256 are OPTIONAL.

Section 4.1, Section 4.2.1, Section 4.2.2 and Section 4.4 specify the required output length for each use of SHAKE128 or SHAKE256 in message digests, RSASSA-PSS, ECDSA and KMAC.

4. Use in CMS

4.1. Message Digests

The id-shake128 and id-shake256 OIDs (Section 3) can be used as the digest algorithm identifiers located in the SignedData, SignerInfo, DigestedData, and the AuthenticatedData digestAlgorithm fields in CMS [RFC5652]. The OID encoding MUST omit the parameters field and the output length of SHAKE128 or SHAKE256 as the message digest MUST be 32 or 64 bytes, respectively.

The digest values are located in the DigestedData field and the Message Digest authenticated attribute included in the signedAttributes of the SignedData signerInfo. In addition, digest values are input to signature algorithms. The digest algorithm MUST be the same as the message hash algorithms used in signatures.

4.2. Signatures

In CMS, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of SignedData content type and countersignature attribute. Signature values are located in the SignerInfo signature field of SignedData content type and countersignature attribute.

Conforming implementations that process RSASSA-PSS and ECDSA with SHAKE signatures when processing CMS data MUST recognize the corresponding OIDs specified in Section 3.

When using RSASSA-PSS or ECDSA with SHAKEs, the RSA modulus or ECDSA curve order SHOULD be chosen in line with the SHAKE output length. Refer to Section 6 for more details.

4.2.1. RSASSA-PSS Signatures

The RSASSA-PSS algorithm is defined in [RFC8017]. When id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 specified in Section 3 is used, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256. [RFC4055] defines RSASSA-PSS-params that are used to define the algorithms and inputs to the algorithm. This specification does not use parameters because the hash, mask generation algorithm, trailer and salt are embedded in the OID definition.

The hash algorithm to hash a message being signed and the hash algorithm as the mask generation function used in RSASSA-PSS MUST be the same: both SHAKE128 or both SHAKE256. The output length of the hash algorithm which hashes the message SHALL be 32 (for SHAKE128) or 64 bytes (for SHAKE256).

The mask generation function takes an octet string of variable length and a desired output length as input, and outputs an octet string of the desired length. In RSASSA-PSS with SHAKEs, the SHAKEs MUST be used natively as the MGF function, instead of the MGF1 algorithm that uses the hash function in multiple iterations as specified in Section B.2.1 of [RFC8017]. In other words, the MGF is defined as the SHAKE128 or SHAKE256 with input being the mgfSeed for id-RSASSA-PSS- SHAKE128 and id-RSASSA-PSS-SHAKE256, respectively. The mgfSeed is the seed from which mask is generated, an octet string [RFC8017]. As explained in Step 9 of section 9.1.1 of [RFC8017], the output length of the MGF is emLen - hLen - 1 bytes. emLen is the maximum message length ceil((n-1)/8), where n is the RSA modulus in bits. hLen is 32 and 64-bytes for id-RSASSA-PSS-SHAKE128 and id-RSASSA-PSS-SHAKE256, respectively. Thus when SHAKE is used as the MGF, the

SHAKE output length maskLen is (8*emLen - 264) or (8*emLen - 520) bits, respectively. For example, when RSA modulus n is 2048, the output length of SHAKE128 or SHAKE256 as the MGF will be 1784 or 1528-bits when id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 is used, respectively.

The RSASSA-PSS saltLength MUST be 32 bytes for id-RSASSA-PSS-SHAKE128 or 64 bytes for id-RSASSA-PSS-SHAKE256. Finally, the trailerField MUST be 1, which represents the trailer field with hexadecimal value 0xBC [RFC8017].

4.2.2. ECDSA Signatures

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in [X9.62]. When the id-ecdsa-with-shake128 or id-ecdsa-with-shake256 (specified in Section 3) algorithm identifier appears, the respective SHAKE function is used as the hash. The encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, the OID id-ecdsa-with-shake128 or id-ecdsa-with-shake256.

For simplicity and compliance with the ECDSA standard specification, the output length of the hash function must be explicitly determined. The output length for SHAKE128 or SHAKE256 used in ECDSA MUST be 32 or 64 bytes, respectively.

Conforming CA implementations that generate ECDSA with SHAKE signatures in certificates or CRLs SHOULD generate such signatures with a deterministically generated, non-random k in accordance with all the requirements specified in [RFC6979]. They MAY also generate such signatures in accordance with all other recommendations in [X9.62] or [SEC1] if they have a stated policy that requires conformance to those standards. Those standards have not specified SHAKE128 and SHAKE256 as hash algorithm options. However, SHAKE128 and SHAKE256 with output length being 32 and 64 octets, respectively can be used instead of 256 and 512-bit output hash algorithms such as SHA256 and SHA512.

4.3. Public Keys

In CMS, the signer's public key algorithm identifiers are located in the OriginatorPublicKey's algorithm attribute. The conventions and encoding for RSASSA-PSS and ECDSA public keys algorithm identifiers are as specified in Section 2.3 of [RFC3279], Section 3.1 of [RFC4055] and Section 2.1 of [RFC5480].

Traditionally, the rsaEncryption object identifier is used to identify RSA public keys. The rsaEncryption object identifier

continues to identify the public key when the RSA private key owner does not wish to limit the use of the public key exclusively to RSASSA-PSS with SHAKEs. When the RSA private key owner wishes to limit the use of the public key exclusively to RSASSA-PSS, the AlgorithmIdentifier for RSASSA-PSS defined in Section 3 SHOULD be used as the algorithm attribute in the OriginatorPublicKey sequence. Conforming client implementations that process RSASSA-PSS with SHAKE public keys in CMS message MUST recognize the corresponding OIDs in Section 3.

Conforming implementations MUST specify and process the algorithms explicitly by using the OIDs specified in Section 3 when encoding ECDSA with SHAKE public keys in CMS messages.

The identifier parameters, as explained in Section 3, MUST be absent.

4.4. Message Authentication Codes

KMAC message authentication code (KMAC) is specified in [SP800-185]. In CMS, KMAC algorithm identifiers are located in the AuthenticatedData macAlgorithm field. The KMAC values are located in the AuthenticatedData mac field.

When the id-KmacWithSHAKE128 or id-KmacWithSHAKE256 OID is used as the MAC algorithm identifier, the parameters field is optional (absent or present). If absent, the SHAKE256 output length used in KMAC is 32 or 64 bytes, respectively, and the customization string is an empty string by default.

Conforming implementations that process KMACs with the SHAKEs when processing CMS data MUST recognize these identifiers.

When calculating the KMAC output, the variable N is 0xD2B282C2, S is an empty string, and L, the integer representing the requested output length in bits, is 256 or 512 for KmacWithSHAKE128 or KmacWithSHAKE256, respectively, in this specification.

5. IANA Considerations

One object identifier for the ASN.1 module in Appendix A was requested for the SMI Security for S/MIME Module Identifiers (1.2.840.113549.1.9.16.0) registry:

Decimal	Description	References
70	CMSAlgsForSHAKE-2019	[EDNOTE: THIS RFC]

6. Security Considerations

This document updates [RFC3370]. The security considerations section of that document applies to this specification as well.

NIST has defined appropriate use of the hash functions in terms of the algorithm strengths and expected time frames for secure use in Special Publications (SPs) [SP800-78-4] and [SP800-107]. These documents can be used as guides to choose appropriate key sizes for various security scenarios.

SHAKE128 with output length of 32 bytes offers 128-bits of collision and preimage resistance. Thus, SHAKE128 OIDs in this specification are RECOMMENDED with 2048 (112-bit security) or 3072-bit (128-bit security) RSA modulus or curves with group order of 256-bits (128-bit security). SHAKE256 with 64 bytes output length offers 256-bits of collision and preimage resistance. Thus, the SHAKE256 OIDs in this specification are RECOMMENDED with 4096-bit RSA modulus or higher or curves with group order of at least 512 bits such as NIST Curve P-521 (256-bit security). Note that we recommended 4096-bit RSA because we would need 15360-bit modulus for 256-bits of security which is impractical for today's technology.

When more than two parties share the same message-authentication key, data origin authentication is not provided. Any party that knows the message-authentication key can compute a valid MAC, therefore the content could originate from any one of the parties.

7. Acknowledgements

This document is based on Russ Housley's draft [I-D.housley-lamps-cms-sha3-hash]. It replaces SHA3 hash functions by SHAKE128 and SHAKE256 as the LAMPS WG agreed.

The authors would like to thank Russ Housley for his guidance and very valuable contributions with the ASN.1 module. Valuable feedback was also provided by Eric Rescorla.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, DOI 10.17487/RFC3370, August 2002, <https://www.rfc-editor.org/info/rfc3370>.

[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, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/info/rfc4055>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SHA3] National Institute of Standards and Technology, U.S. Department of Commerce, "SHA-3 Standard - Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, August 2015.

[SP800-185] National Institute of Standards and Technology, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. NIST SP 800-185", December 2016, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

8.2. Informative References

[I-D.housley-lamps-cms-sha3-hash] Housley, R., "Use of the SHA3 One-way Hash Functions in the Cryptographic Message Syntax (CMS)", draft-housley-lamps-cms-sha3-hash-00 (work in progress), March 2017.

[I-D.ietf-lamps-pkix-shake]
Kampanakis, P. and Q. Dang, "Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA using SHAKEs", draft-ietf-lamps-pkix-shake-15 (work in progress), July 2019.

[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, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, DOI 10.17487/RFC5753, January 2010, <https://www.rfc-editor.org/info/rfc5753>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/info/rfc6268>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", May 2009, <http://www.secg.org/sec1-v2.pdf>.

[shake-nist-oids]
National Institute of Standards and Technology, "Computer Security Objects Register", October 2017, <https://csrc.nist.gov/Projects/Computer-Security-Objects-Register/Algorithm-Registration>.

[SP800-107]
National Institute of Standards and Technology (NIST), "SP800-107: Recommendation for Applications Using Approved Hash Algorithms", May 2014, <https://csrc.nist.gov/csrc/media/publications/sp/800-107/rev-1/final/documents/draft_revised_sp800-107.pdf>.

[SP800-78-4]
National Institute of Standards and Technology (NIST), "SP800-78-4: Cryptographic Algorithms and Key Sizes for Personal Identity Verification", May 2014, <https://csrc.nist.gov/csrc/media/publications/sp/800-78/4/final/documents/sp800_78-4_revised_draft.pdf>.

[X9.62] American National Standard for Financial Services (ANSI), "X9.62-2005 Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA)", November 2005.

Appendix A. ASN.1 Module

This appendix includes the ASN.1 modules for SHAKEs in CMS. This module includes some ASN.1 from other standards for reference.

```
CMSAlgsForSHAKE-2019 { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)
     id-mod-cms-shakes-2019(70) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL;

IMPORTS

DIGEST-ALGORITHM, MAC-ALGORITHM, SMIME-CAPS
FROM AlgorithmInformation-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58) }

RSAPublicKey, rsaEncryption, id-ecPublicKey
FROM PKIXAlgs-2009 { iso(1) identified-organization(3) dod(6)
     internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-pkix1-algorithms2008-02(56) }

sa-rsassapssWithSHAKE128, sa-rsassapssWithSHAKE256,
```

```
   sa-ecdsaWithSHAKE128, sa-ecdsaWithSHAKE256
   FROM PKIXAlgsForSHAKE-2019 {
      iso(1) identified-organization(3) dod(6)
      internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-shakes-2019(94) } ;

   -- Message Digest Algorithms (mda-)
   -- used in SignedData, SignerInfo, DigestedData,
   -- and the AuthenticatedData digestAlgorithm
   -- fields in CMS
   --
   --  This expands MessageAuthAlgs from [RFC5652] and
   --  MessageDigestAlgs in [RFC5753]
   --
   -- MessageDigestAlgs DIGEST-ALGORITHM ::= {
   --  mda-shake128   |
   --  mda-shake256,
   --  ...
   -- }

   --
   -- One-Way Hash Functions
   -- SHAKE128
   mda-shake128 DIGEST-ALGORITHM ::= {
     IDENTIFIER id-shake128  -- with output length 32 bytes.
   }
   id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                       us(840) organization(1) gov(101)
                                       csor(3) nistAlgorithm(4)
                                       hashAlgs(2) 11 }

   -- SHAKE256
   mda-shake256 DIGEST-ALGORITHM ::= {
     IDENTIFIER id-shake256  -- with output length 64 bytes.
   }
   id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                       us(840) organization(1) gov(101)
                                       csor(3) nistAlgorithm(4)
                                       hashAlgs(2) 12 }

   --
   -- Public key algorithm identifiers located in the
   -- OriginatorPublicKey's algorithm attribute in CMS.
   -- And Signature identifiers used in SignerInfo
   -- signatureAlgorithm field of SignedData content
   -- type and countersignature attribute in CMS.
   --
   -- From RFC5280, for reference.
```

```
   -- rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }
      -- When the rsaEncryption algorithm identifier is used
      -- for a public key, the AlgorithmIdentifier parameters
      -- field MUST contain NULL.
   --
   id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6) 30 }

   id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6) 31 }

      -- When the id-RSASSA-PSS-* algorithm identifiers are used
      -- for a public key or signature in CMS, the AlgorithmIdentifier
      -- parameters field MUST be absent. The message digest algorithm
      -- used in RSASSA-PSS MUST be SHAKE128 or SHAKE256 with a 32 or
      -- 64 byte outout length, respectively. The mask generation
      -- function MUST be SHAKE128 or SHAKE256 with an output length
      -- of (8*ceil((n-1)/8) - 264) or (8*ceil((n-1)/8) - 520) bits,
      -- respectively, where n is the RSA modulus in bits.
      -- The RSASSA-PSS saltLength MUST be 32 or 64 bytes, respectively.
      -- The trailerField MUST be 1, which represents the trailer
      -- field with hexadecimal value 0xBC. Regardless of
      -- id-RSASSA-PSS-* or rsaEncryption being used as the
      -- AlgorithmIdentifier of the OriginatorPublicKey, the RSA
      -- public key MUST be encoded using the RSAPublicKey type.

   -- From RFC4055, for reference.
   -- RSAPublicKey ::= SEQUENCE {
   --   modulus INTEGER, -- -- n
   --   publicExponent INTEGER } -- -- e

   id-ecdsa-with-shake128 OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6) 32 }

   id-ecdsa-with-shake256 OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6) 33 }

      -- When the id-ecdsa-with-shake* algorithm identifiers are
      -- used in CMS, the AlgorithmIdentifier parameters field
      -- MUST be absent and the signature algorithm should be
      -- deterministic ECDSA [RFC6979]. The message digest MUST
      -- be SHAKE128 or SHAKE256 with a 32 or 64 byte outout
      -- length, respectively. In both cases, the ECDSA public key,
      -- MUST be encoded using the id-ecPublicKey type.
```

```
   -- From RFC5480, for reference.
   -- id-ecPublicKey OBJECT IDENTIFIER ::= {
   --    iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) 1 }
        -- The id-ecPublicKey parameters must be absent or present
        -- and are defined as
   -- ECParameters ::= CHOICE {
   --     namedCurve         OBJECT IDENTIFIER
   --     -- -- implicitCurve   NULL
   --     -- -- specifiedCurve  SpecifiedECDomain
   --  }

   -- This expands SignatureAlgorithms from [RFC5912]
   --
   -- SignatureAlgs SIGNATURE-ALGORITHM ::= {
   --   sa-rsassapssWithSHAKE128 |
   --   sa-rsassapssWithSHAKE256 |
   --   sa-ecdsaWithSHAKE128 |
   --   sa-ecdsaWithSHAKE256,
   --   ...
   -- }

   -- This expands MessageAuthAlgs from [RFC5652] and [RFC6268]
   --
   -- Message Authentication (maca-) Algorithms
   -- used in AuthenticatedData macAlgorithm in CMS
   --
   MessageAuthAlgs MAC-ALGORITHM ::= {
       maca-KMACwithSHAKE128   |
       maca-KMACwithSHAKE256,
       ...
   }

   -- This expands SMimeCaps from [RFC5911]
   --
   SMimeCaps SMIME-CAPS ::= {
      -- sa-rsassapssWithSHAKE128.&smimeCaps |
      -- sa-rsassapssWithSHAKE256.&smimeCaps |
      -- sa-ecdsaWithSHAKE128.&smimeCaps |
      -- sa-ecdsaWithSHAKE256.&smimeCaps,
      maca-KMACwithSHAKE128.&smimeCaps   |
      maca-KMACwithSHAKE256.&smimeCaps,
      ...
    }

   --
   -- KMAC with SHAKE128
   maca-KMACwithSHAKE128 MAC-ALGORITHM ::= {
         IDENTIFIER id-KMACWithSHAKE128
```

```
        PARAMS TYPE KMACwithSHAKE128-params ARE optional
          -- If KMACwithSHAKE128-params parameters are absent
          -- the SHAKE128 output length used in KMAC is 256 bits
          -- and the customization string is an empty string.
        IS-KEYED-MAC TRUE
        SMIME-CAPS {IDENTIFIED BY id-KMACWithSHAKE128}
   }
   id-KMACWithSHAKE128 OBJECT IDENTIFIER ::=  { joint-iso-itu-t(2)
                               country(16) us(840) organization(1)
                               gov(101) csor(3) nistAlgorithm(4)
                               hashAlgs(2) 19 }
   KMACwithSHAKE128-params ::= SEQUENCE {
     kMACOutputLength     INTEGER DEFAULT 256, -- Output length in bits
     customizationString  OCTET STRING DEFAULT ''H
   }

   -- KMAC with SHAKE256
   maca-KMACwithSHAKE256 MAC-ALGORITHM ::= {
         IDENTIFIER id-KMACWithSHAKE256
         PARAMS TYPE KMACwithSHAKE256-params ARE optional
            -- If KMACwithSHAKE256-params parameters are absent
            -- the SHAKE256 output length used in KMAC is 512 bits
            -- and the customization string is an empty string.
         IS-KEYED-MAC TRUE
         SMIME-CAPS {IDENTIFIED BY id-KMACWithSHAKE256}
   }
   id-KMACWithSHAKE256 OBJECT IDENTIFIER ::=  { joint-iso-itu-t(2)
                               country(16) us(840) organization(1)
                               gov(101) csor(3) nistAlgorithm(4)
                               hashAlgs(2) 20 }
   KMACwithSHAKE256-params ::= SEQUENCE {
      kMACOutputLength     INTEGER DEFAULT 512, -- Output length in bits
      customizationString  OCTET STRING DEFAULT ''H
   }

   END
```

Authors' Addresses

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Quynh Dang
NIST
100 Bureau Drive
Gaithersburg, MD 20899

Email: quynh.Dang@nist.gov

LAMPS
Internet-Draft
Updates: 5280 (if approved)
Intended status: Standards Track
Expires: September 5, 2018

A. Melnikov, Ed.
Isode Ltd
W. Chuang, Ed.
Google, Inc.
March 4, 2018

Internationalized Email Addresses in X.509 certificates
draft-ietf-lamps-eai-addresses-18

Abstract

This document defines a new name form for inclusion in the otherName field of an X.509 Subject Alternative Name and Issuer Alternative Name extension that allows a certificate subject to be associated with an Internationalized Email Address.

This document updates RFC 5280.

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 https://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 September 5, 2018.

Table of Contents

1. Introduction

[RFC5280] defines the rfc822Name subjectAltName name type for representing [RFC5321] email addresses. The syntax of rfc822Name is restricted to a subset of US-ASCII characters and thus can't be used to represent Internationalized Email addresses [RFC6531]. This document defines a new otherName variant to represent Internationalized Email addresses. In addition this document requires all email address domains in X.509 certificates to conform to IDNA2008 [RFC5890].

2. Conventions Used in This Document

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].

The formal syntax uses the Augmented Backus-Naur Form (ABNF) [RFC5234] notation.

3. Name Definitions

The GeneralName structure is defined in [RFC5280], and supports many different name forms including otherName for extensibility. This section specifies the SmtpUTF8Mailbox name form of otherName, so that Internationalized Email addresses can appear in the subjectAltName of

a certificate, the issuerAltName of a certificate, or anywhere else that GeneralName is used.

```
id-on-SmtpUTF8Mailbox OBJECT IDENTIFIER ::= { id-on 9 }

SmtpUTF8Mailbox ::= UTF8String (SIZE (1..MAX))
-- SmtpUTF8Mailbox conforms to Mailbox as specified
-- in Section 3.3 of RFC 6531.
```

When the subjectAltName (or issuerAltName) extension contains an Internationalized Email address with a non-ASCII local-part, the address MUST be stored in the SmtpUTF8Mailbox name form of otherName. The format of SmtpUTF8Mailbox is defined as the ABNF rule SmtpUTF8Mailbox. SmtpUTF8Mailbox is a modified version of the Internationalized Mailbox which was defined in Section 3.3 of [RFC6531] which was itself derived from SMTP Mailbox from Section 4.1.2 of [RFC5321]. [RFC6531] defines the following ABNF rules for Mailbox whose parts are modified for internationalization: <Local-part>, <Dot-string>, <Quoted-string>, <QcontentSMTP>, <Domain>, and <Atom>. In particular, <Local-part> was updated to also support UTF8-non-ascii. UTF8-non-ascii was described by Section 3.1 of [RFC6532]. Also, domain was extended to support U-labels, as defined in [RFC5890].

This document further refines Internationalized [RFC6531] Mailbox ABNF rules and calls this SmtpUTF8Mailbox. In SmtpUTF8Mailbox, labels that include non-ASCII characters MUST be stored in U-label (rather than A-label) [RFC5890] form. This restriction removes the need to determine which label encoding A- or U-label is present in the Domain. As per Section 2.3.2.1 of [RFC5890], U-label are encoded as UTF-8 [RFC3629] in Normalization Form C and other properties specified there. In SmtpUTF8Mailbox, domain labels that solely use ASCII characters (meaning not A- nor U-labels) SHALL use NR-LDH restrictions as specified by Section 2.3.1 of [RFC5890] and SHALL be restricted to lower case letters. NR-LDH stands for "Non-Reserved Letters Digits Hyphen" and is the set of LDH labels that do not have "--" characters in the third and forth character position, which excludes "tagged domain names" such as A-labels. Consistent with the treatment of rfc822Name in [RFC5280], SmtpUTF8Mailbox is an envelope <Mailbox> and has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

Due to name constraint compatibility reasons described in Section 6, SmtpUTF8Mailbox subjectAltName MUST NOT be used unless the local-part of the email address contains non-ASCII characters. When the local-part is ASCII, rfc822Name subjectAltName MUST be used instead of SmtpUTF8Mailbox. This is compatible with legacy software that

supports only rfc822Name (and not SmtpUTF8Mailbox). The appropriate usage of rfc822Name and SmtpUTF8Mailbox is summarized in Table 1 below.

SmtpUTF8Mailbox is encoded as UTF8String. The UTF8String encoding MUST NOT contain a Byte-Order- Mark (BOM) [RFC3629] to aid consistency across implementations particularly for comparison.

local-part char	domain char	domain label	subjectAltName
ASCII-only	ASCII-only	NR-LDH label	rfc822Name
non-ASCII	ASCII-only	NR-LDH label	SmtpUTF8Mailbox
ASCII-only	non-ASCII	A-label	rfc822Name
non-ASCII	non-ASCII	U-label	SmtpUTF8Mailbox

non-ASCII may additionally include ASCII characters.

Table 1: Email address formatting

4. IDNA2008

To facilitate comparison between email addresses, all email address domains in X.509 certificates MUST conform to IDNA2008 [RFC5890] (and avoid any "mappings" mentioned in that document). Use of non-conforming email address domains introduces the possibility of conversion errors between alternate forms. This applies to SmtpUTF8Mailbox and rfc822Name in subjectAltName, issuerAltName and anywhere else that these are used.

5. Matching of Internationalized Email Addresses in X.509 certificates

In equivalence comparison with SmtpUTF8Mailbox, there may be some setup work on one or both inputs depending of whether the input is already in comparison form. Comparing SmtpUTF8Mailboxs consists of a domain part step and a local-part step. The comparison form for local-parts is always UTF-8. The comparison form for domain parts depends on context. While some contexts such as certificate path validation in [RFC5280] specify transforming domain to A-label (Section 7.5 and 7.2 in [RFC5280] as updated by [ID-lamps-rfc5280-i18n-update]), this document recommends transforming to UTF-8 U-label instead. This reduces the likelihood of errors by reducing conversions as more implementations natively support U-label domains.

Comparison of two SmtpUTF8Mailbox is straightforward with no setup work needed. They are considered equivalent if there is an exact

octet-for-octet match. Comparison with email addresses such as Internationalized email address or rfc822Name requires additional setup steps for domain part and local-part. The initial preparation for the email addresses is to remove any phrases or comments, as well as "<" and ">" present. This document calls for comparison of domain labels that include non-ASCII characters be transformed to U-label if not already in that form. The first step is to detect use of the A-label by using Section 5.1 of [RFC5891]. Next if necessary, transform any A-labels to U-labels Unicode as specified in Section 5.2 of [RFC5891]. Finally if necessary convert the Unicode to UTF-8 as specified in Section 3 of [RFC3629]. For ASCII NR-LDH labels, upper case letters are converted to lower case letters. In setup for SmtpUTF8Mailbox, the email address local-part MUST conform to the requirements of [RFC6530] and [RFC6531], including being a string in UTF-8 form. In particular, the local-part MUST NOT be transformed in any way, such as by doing case folding or normalization of any kind. The <Local-part> part of an Internationalized email address is already in UTF-8. For rfc822Name the local-part, which is IA5String (ASCII), trivially maps to UTF-8 without change. Once setup is complete, they are again compared octet-for-octet.

To summarize non-normatively, the comparison steps including setup are:

1. If the domain contains A-labels, transform them to U-labels.

2. If the domain contains ASCII NR-LDH labels, lowercase them.

3. Compare strings octet-for-octet for equivalence.

This specification expressly does not define any wildcard characters and SmtpUTF8Mailbox comparison implementations MUST NOT interpret any character as wildcards. Instead, to specify multiple email addresses through SmtpUTF8Mailbox, the certificate MUST use multiple subjectAltNames or issuerAltNames to explicitly carry any additional email addresses.

6. Name constraints in path validation

This section updates Section 4.2.1.10 of [RFC5280] to extend rfc822Name name constraints to SmtpUTF8Mailbox subjectAltNames. A SmtpUTF8Mailbox aware path validators will apply name constraint comparison to the subject distinguished name and both forms of subject alternative name rfc822Name and SmtpUTF8Mailbox.

Both rfc822Name and SmtpUTF8Mailbox subject alternative names represent the same underlying email address namespace. Since legacy

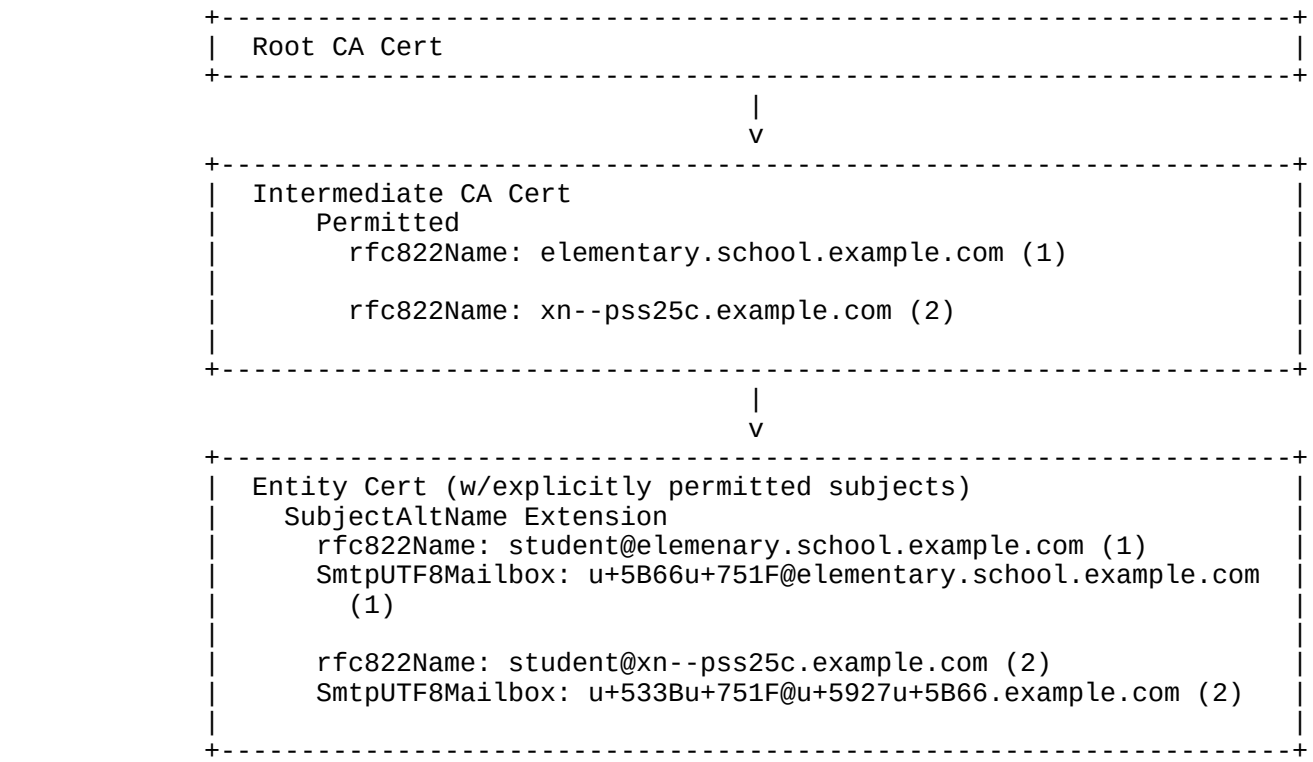
CAs constrained to issue certificates for a specific set of domains would lack corresponding UTF-8 constraints, [ID-lamps-rfc5280-i18n-update] updates modifies and extends rfc822Name name constraints defined in [RFC5280] to cover SmtpUTF8Mailbox subject alternative names. This ensures that the introduction of SmtpUTF8Mailbox does not violate existing name constraints. Since it is not valid to include non-ASCII UTF-8 characters in the local-part of rfc822Name name constraints, and since name constraints that include a local-part are rarely, if at all, used in practice, name constraints updated in [ID-lamps-rfc5280-i18n-update] admit the forms that represent all addresses at a host or all mailboxes in a domain, and deprecates rfc822Name name constraints that represent a particular mailbox. That is, rfc822Name constraints with a local-part SHOULD NOT be used.

Constraint comparison with SmtpUTF8Mailbox subjectAltName starts with the setup steps defined by Section 5. Setup converts the inputs of the comparison which is one of a subject distinguished name or a rfc822Name or SmtpUTF8Mailbox subjectAltName, and one of a rfc822Name name constraint, to constraint comparison form. For rfc822Name name constraint, this will convert any domain A-labels to U-labels. For both the name constraint and the subject, this will lower case any domain NR-LDH labels. Strip the local-part and "@" separator from each rfc822Name and SmtpUTF8Mailbox, leaving just the domain-part. After setup, this follows the comparison steps defined in 4.2.1.10 of [RFC5280] as follows. If the resulting name constraint domain starts with a "." character, then for the name constraint to match, a suffix of the resulting subject alternative name domain MUST match the name constraint (including the leading ".") octet for octet. If the resulting name constraint domain does not start with a "." character, then for the name constraint to match, the entire resulting subject alternative name domain MUST match the name constraint octet for octet.

Certificate Authorities that wish to issue CA certificates with email address name constraint MUST use rfc822Name subject alternative names only. These MUST be IDNA2008 conformant names with no mappings, and with non-ASCII domains encoded in A-labels only.

The name constraint requirement with SmtpUTF8Mailbox subject alternative name is illustrated in the non-normative diagram Figure 1. The first example (1) illustrates a permitted rfc822Name ASCII only hostname name constraint, and the corresponding valid rfc822Name subjectAltName and SmtpUTF8Mailbox subjectAltName email addresses. The second example (2) illustrates a permitted rfc822Name hostname name constraint with A-label, and the corresponding valid rfc822Name subjectAltName and SmtpUTF8Mailbox subjectAltName email addresses. Note that an email address with ASCII only local-part is

encoded as rfc822Name despite also having unicode present in the domain.



Name constraints with SmtpUTF8Name and rfc822Name

Figure 1

7. Security Considerations

Use of SmtpUTF8Mailbox for certificate subjectAltName (and issuerAltName) will incur many of the same security considerations as in Section 8 in [RFC5280], but introduces a new issue by permitting non-ASCII characters in the email address local-part. This issue, as mentioned in Section 4.4 of [RFC5890] and in Section 4 of [RFC6532], is that use of Unicode introduces the risk of visually similar and identical characters which can be exploited to deceive the recipient. The former document references some means to mitigate against these attacks. See [WEBER] for more background on security issues with Unicode.

8. IANA Considerations

In Section 3 and the ASN.1 module identifier defined in Appendix A. IANA is kindly requested to make the following assignments for:

The LAMPS-EaiAddresses-2016 ASN.1 module in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).

The SmtpUTF8Mailbox otherName in the "PKIX Other Name Forms" registry (1.3.6.1.5.5.7.8). {{ Note to IANA: id-on-smtputf8Name was assigned based on an earlier version of this document. Please change that entry to id-on-SmtpUTF8Mailbox. }}

9. References

9.1. Normative References

[ID-lamps-rfc5280-i18n-update]
Housley, R., "Internationalization Updates to RFC 5280", June 2017, <https://datatracker.ietf.org/doc/draft-housley-rfc5280-i18n-update/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[RFC6531] Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.

9.2. Informative References

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[WEBER] Weber, C., "Attacking Software Globalization", March 2010, <https://www.lookout.net/files/Chris_Weber_Character%20Transformations%20v1.7_IUC33.pdf>.

Appendix A. ASN.1 Module

The following ASN.1 module normatively specifies the SmtpUTF8Mailbox structure. This specification uses the ASN.1 definitions from [RFC5912] with the 2002 ASN.1 notation used in that document. [RFC5912] updates normative documents using older ASN.1 notation.

```
  LAMPS-EaiAddresses-2016
    { iso(1) identified-organization(3) dod(6)
      internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-lamps-eai-addresses-2016(TBD) }

  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN

  IMPORTS
    OTHER-NAME
    FROM PKIX1Implicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59) }

    id-pkix
    FROM PKIX1Explicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51) } ;

  --
  -- otherName carries additional name types for subjectAltName,
  -- issuerAltName, and other uses of GeneralNames.
  --

    id-on OBJECT IDENTIFIER ::= { id-pkix 8 }

    SmtpUtf8OtherNames OTHER-NAME ::= { on-SmtpUTF8Mailbox, ... }

    on-SmtpUTF8Mailbox OTHER-NAME ::= {
        SmtpUTF8Mailbox IDENTIFIED BY id-on-SmtpUTF8Mailbox
    }

    id-on-SmtpUTF8Mailbox OBJECT IDENTIFIER ::= { id-on 9 }

    SmtpUTF8Mailbox ::= UTF8String (SIZE (1..MAX))
     -- SmtpUTF8Mailbox conforms to Mailbox as specified
     -- in Section 3.3 of RFC 6531.

  END
```

Appendix B. Example of SmtpUTF8Mailbox

This non-normative example demonstrates using SmtpUTF8Mailbox as an otherName in GeneralName to encode the email address "u+8001u+5E2B@example.com".

```
The hexadecimal DER encoding of the email address is:
A022060A 2B060105 05070012 0809A014 0C12E880 81E5B8AB 40657861
6D706C65 2E636F6D

The text decoding is:
  0  34: [0] {
  2  10:   OBJECT IDENTIFIER '1 3 6 1 5 5 7 0 18 8 9'
 14  20:   [0] {
 16  18:     UTF8String '..@example.com'
       :     }
       :   }
```

Figure 2

The example was encoded on the OSS Nokalva ASN.1 Playground and the above text decoding is an output of Peter Gutmann's "dumpasn1" program.

Appendix C. Acknowledgements

Thank you to Magnus Nystrom for motivating this document. Thanks to Russ Housley, Nicolas Lidzborski, Laetitia Baudoin, Ryan Sleevi, Sean Leonard, Sean Turner, John Levine, and Patrik Falstrom for their feedback. Also special thanks to John Klensin for his valuable input on internationalization, Unicode and ABNF formatting, to Jim Schaad for his help with the ASN.1 example and his helpful feedback, and especially to Viktor Dukhovni for helping us with name constraints and his many detailed document reviews.

Authors' Addresses

Alexey Melnikov (editor)
Isode Ltd
14 Castle Mews
Hampton, Middlesex TW12 2NP
UK

Email: Alexey.Melnikov@isode.com

Weihaw Chuang (editor)
Google, Inc.
1600 Amphitheater Parkway
Mountain View, CA 94043
US

Email: weihaw@google.com

LAMPS WG
Internet-Draft
Updates: 3279 (if approved)
Intended status: Standards Track
Expires: January 22, 2020

P. Kampanakis
Cisco Systems
Q. Dang
NIST

July 21, 2019

Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA using SHAKEs draft-ietf-lamps-pkix-shake-15

Abstract

Digital signatures are used to sign messages, X.509 certificates and CRLs. This document updates the "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile" (RFC3279) and describes the conventions for using the SHAKE function family in Internet X.509 certificates and revocation lists as one-way hash functions with the RSA Probabilistic signature and ECDSA signature algorithms. The conventions for the associated subject public keys are also described.

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 https://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 22, 2020.

Copyright Notice

Table of Contents

1. Change Log

[EDNOTE: Remove this section before publication.]

- draft-ietf-lamps-pkix-shake-15:
  - Minor editorial nits.
- draft-ietf-lamps-pkix-shake-14:
  - Fixing error with incorrect preimage resistance bits for SHA128 and SHA256.
- draft-ietf-lamps-pkix-shake-13:
  - Addressing one applicable comment from Dan M. about sec levels while in secdir review of draft-ietf-lamps-cms-shakes.
  - Addressing comment from Scott B.'s opsdir review about references in the abstract.
- draft-ietf-lamps-pkix-shake-12:

    * Nits identified by Roman, Eric V. Ben K., Barry L. in ballot position review.

- draft-ietf-lamps-pkix-shake-11:

    * Nits identified by Roman in AD Review.

- draft-ietf-lamps-pkix-shake-10:

    * Updated IANA considerations section to request for OID assignments.

- draft-ietf-lamps-pkix-shake-09:

    * Fixed minor text nits.

    * Added text name allocation for SHAKEs in IANA considerations.

    * Updates in Sec Considerations section.

- draft-ietf-lamps-pkix-shake-08:

    * Small nits from Russ while in WGLC.

- draft-ietf-lamps-pkix-shake-07:

    * Incorporated Eric's suggestion from WGLC.

- draft-ietf-lamps-pkix-shake-06:

    * Added informative references.

    * Updated ASN.1 so it compiles.

    * Updated IANA considerations.

- draft-ietf-lamps-pkix-shake-05:

    * Added RFC8174 reference and text.

    * Explicitly explained why RSASSA-PSS-params are omitted in section 5.1.1.

    * Simplified Public Keys section by removing redundant info from RFCs.

- draft-ietf-lamps-pkix-shake-04:

  * Removed paragraph suggesting KMAC to be used in generating k in Deterministic ECDSA. That should be RFC6979-bis.

  * Removed paragraph from Security Considerations that talks about randomness of k because we are using deterministic ECDSA.

  * Various ASN.1 fixes.

  * Text fixes.

- draft-ietf-lamps-pkix-shake-03:

  * Updates based on suggestions and clarifications by Jim.

  * Added ASN.1.

- draft-ietf-lamps-pkix-shake-02:

  * Significant reorganization of the sections to simplify the introduction, the new OIDs and their use in PKIX.

  * Added new OIDs for RSASSA-PSS that hardcode hash, salt and MGF, according the WG consensus.

  * Updated Public Key section to use the new RSASSA-PSS OIDs and clarify the algorithm identifier usage.

  * Removed the no longer used SHAKE OIDs from section 3.1.

  * Consolidated subsection for message digest algorithms.

  * Text fixes.

- draft-ietf-lamps-pkix-shake-01:

  * Changed titles and section names.

  * Removed DSA after WG discussions.

  * Updated shake OID names and parameters, added MGF1 section.

  * Updated RSASSA-PSS section.

  * Added Public key algorithm OIDs.

  * Populated Introduction and IANA sections.

- draft-ietf-lamps-pkix-shake-00:

* Initial version

2. Introduction

[RFC3279] defines cryptographic algorithm identifiers for the Internet X.509 Certificate and Certificate Revocation Lists (CRL) profile [RFC5280]. This document updates RFC3279 and defines identifiers for several cryptographic algorithms that use variable length output SHAKE functions introduced in [SHA3] which can be used with .

In the SHA-3 family, two extendable-output functions (SHAKEs), SHAKE128 and SHAKE256, are defined. Four other hash function instances, SHA3-224, SHA3-256, SHA3-384, and SHA3-512, are also defined but are out of scope for this document. A SHAKE is a variable length hash function defined as SHAKE(M, d) where the output is a d-bits-long digest of message M. The corresponding collision and second-preimage-resistance strengths for SHAKE128 are min(d/2,128) and min(d,128) bits, respectively (Appendix A.1 [SHA3]). And the corresponding collision and second-preimage-resistance strengths for SHAKE256 are min(d/2,256) and min(d,256) bits, respectively.

A SHAKE can be used as the message digest function (to hash the message to be signed) in RSASSA-PSS [RFC8017] and ECDSA [X9.62] and as the hash in the mask generation function (MGF) in RSASSA-PSS.

3. Terminology

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Identifiers

This section defines four new object identifiers (OIDs), for RSASSA-PSS and ECDSA with each of SHAKE128 and SHAKE256. The same algorithm identifiers can be used for identifying a public key in RSASSA-PSS.

The new identifiers for RSASSA-PSS signatures using SHAKEs are below.

```
id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6)
          TBD1 }

id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6)
          TBD2 }
```

The new algorithm identifiers of ECDSA signatures using SHAKEs are below.

```
id-ecdsa-with-shake128 OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6)
          TBD3 }

id-ecdsa-with-shake256 OBJECT IDENTIFIER  ::=  { iso(1)
          identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) algorithms(6)
          TBD4 }
```

The parameters for the four identifiers above MUST be absent. That is, the identifier SHALL be a SEQUENCE of one component, the OID.

Section 5.1.1 and Section 5.1.2 specify the required output length for each use of SHAKE128 or SHAKE256 in RSASSA-PSS and ECDSA. In summary, when hashing messages to be signed, output lengths of SHAKE128 and SHAKE256 are 256 and 512 bits respectively. When the SHAKEs are used as mask generation functions RSASSA-PSS, their output length is (8*ceil((n-1)/8) - 264) or (8*ceil((n-1)/8) - 520) bits, respectively, where n is the RSA modulus size in bits.

5. Use in PKIX

5.1. Signatures

Signatures are used in a number of different ASN.1 structures. As shown in the ASN.1 representation from [RFC5280] below, in an X.509 certificate, a signature is encoded with an algorithm identifier in the signatureAlgorithm attribute and a signatureValue attribute that contains the actual signature.

```
Certificate  ::=  SEQUENCE  {
   tbsCertificate       TBSCertificate,
   signatureAlgorithm   AlgorithmIdentifier,
   signatureValue       BIT STRING  }
```

The identifiers defined in Section 4 can be used as the AlgorithmIdentifier in the signatureAlgorithm field in the sequence Certificate and the signature field in the sequence TBSCertificate in X.509 [RFC5280]. The parameters of these signature algorithms are absent as explained in Section 4.

Conforming CA implementations MUST specify the algorithms explicitly by using the OIDs specified in Section 4 when encoding RSASSA-PSS or ECDSA with SHAKE signatures in certificates and CRLs. Conforming client implementations that process certificates and CRLs using RSASSA-PSS or ECDSA with SHAKE MUST recognize the corresponding OIDs. Encoding rules for RSASSA-PSS and ECDSA signature values are specified in [RFC4055] and [RFC5480], respectively.

When using RSASSA-PSS or ECDSA with SHAKEs, the RSA modulus and ECDSA curve order SHOULD be chosen in line with the SHAKE output length. Refer to Section 7 for more details.

5.1.1. RSASSA-PSS Signatures

The RSASSA-PSS algorithm is defined in [RFC8017]. When id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 specified in Section 4 is used, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256. [RFC4055] defines RSASSA-PSS-params that are used to define the algorithms and inputs to the algorithm. This specification does not use parameters because the hash, mask generation algorithm, trailer and salt are embedded in the OID definition.

The hash algorithm to hash a message being signed and the hash algorithm used as the mask generation function in RSASSA-PSS MUST be the same: both SHAKE128 or both SHAKE256. The output length of the hash algorithm which hashes the message SHALL be 32 (for SHAKE128) or 64 bytes (for SHAKE256).

The mask generation function takes an octet string of variable length and a desired output length as input, and outputs an octet string of the desired length. In RSASSA-PSS with SHAKEs, the SHAKEs MUST be used natively as the MGF function, instead of the MGF1 algorithm that uses the hash function in multiple iterations as specified in Section B.2.1 of [RFC8017]. In other words, the MGF is defined as the SHAKE128 or SHAKE256 output of the mgfSeed for id-RSASSA-PSS-

SHAKE128 and id-RSASSA-PSS-SHAKE256, respectively. The mgfSeed is the seed from which mask is generated, an octet string [RFC8017]. As explained in Step 9 of section 9.1.1 of [RFC8017], the output length of the MGF is emLen - hLen - 1 bytes. emLen is the maximum message length ceil((n-1)/8), where n is the RSA modulus in bits. hLen is 32 and 64-bytes for id-RSASSA-PSS-SHAKE128 and id-RSASSA-PSS-SHAKE256, respectively. Thus when SHAKE is used as the MGF, the SHAKE output length maskLen is (8*emLen - 264) or (8*emLen - 520) bits, respectively. For example, when RSA modulus n is 2048, the output length of SHAKE128 or SHAKE256 as the MGF will be 1784 or 1528-bits when id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 is used, respectively.

The RSASSA-PSS saltLength MUST be 32 bytes for id-RSASSA-PSS-SHAKE128 or 64 bytes for id-RSASSA-PSS-SHAKE256. Finally, the trailerField MUST be 1, which represents the trailer field with hexadecimal value 0xBC [RFC8017].

5.1.2. ECDSA Signatures

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in [X9.62]. When the id-ecdsa-with-shake128 or id-ecdsa-with-shake256 (specified in Section 4) algorithm identifier appears, the respective SHAKE function (SHAKE128 or SHAKE256) is used as the hash. The encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, the OID id-ecdsa-with-shake128 or id-ecdsa-with-shake256.

For simplicity and compliance with the ECDSA standard specification, the output length of the hash function must be explicitly determined. The output length, d, for SHAKE128 or SHAKE256 used in ECDSA MUST be 256 or 512 bits, respectively.

Conforming CA implementations that generate ECDSA with SHAKE signatures in certificates or CRLs SHOULD generate such signatures with a deterministically generated, non-random k in accordance with all the requirements specified in [RFC6979]. They MAY also generate such signatures in accordance with all other recommendations in [X9.62] or [SEC1] if they have a stated policy that requires conformance to those standards. Those standards have not specified SHAKE128 and SHAKE256 as hash algorithm options. However, SHAKE128 and SHAKE256 with output length being 32 and 64 octets, respectively, can be used instead of 256 and 512-bit output hash algorithms such as SHA256 and SHA512.

5.2. Public Keys

Certificates conforming to [RFC5280] can convey a public key for any public key algorithm. The certificate indicates the public key algorithm through an algorithm identifier. This algorithm identifier is an OID and optionally associated parameters. The conventions and encoding for RSASSA-PSS and ECDSA public keys algorithm identifiers are as specified in Section 2.3.1 and 2.3.5 of [RFC3279], Section 3.1 of [RFC4055] and Section 2.1 of [RFC5480].

Traditionally, the rsaEncryption object identifier is used to identify RSA public keys. The rsaEncryption object identifier continues to identify the subject public key when the RSA private key owner does not wish to limit the use of the public key exclusively to RSASSA-PSS with SHAKEs. When the RSA private key owner wishes to limit the use of the public key exclusively to RSASSA-PSS with SHAKEs, the AlgorithmIdentifiers for RSASSA-PSS defined in Section 4 SHOULD be used as the algorithm field in the SubjectPublicKeyInfo sequence [RFC5280]. Conforming client implementations that process RSASSA-PSS with SHAKE public keys when processing certificates and CRLs MUST recognize the corresponding OIDs.

Conforming CA implementations MUST specify the X.509 public key algorithm explicitly by using the OIDs specified in Section 4 when encoding ECDSA with SHAKE public keys in certificates and CRLs. Conforming client implementations that process ECDSA with SHAKE public keys when processing certificates and CRLs MUST recognize the corresponding OIDs.

The identifier parameters, as explained in Section 4, MUST be absent.

6. IANA Considerations

One object identifier for the ASN.1 module in Appendix A is requested for the SMI Security for PKIX Module Identifiers (1.3.6.1.5.5.7.0) registry:

Decimal	Description	References
TBD	id-mod-pkix1-shakes-2019	[EDNOTE: THIS RFC]

IANA is requested to update the SMI Security for PKIX Algorithms [SMI-PKIX] (1.3.6.1.5.5.7.6) registry with four additional entries:

Decimal	Description	References
TBD1	id-RSASSA-PSS-SHAKE128	[EDNOTE: THIS RFC]
TBD2	id-RSASSA-PSS-SHAKE256	[EDNOTE: THIS RFC]
TBD3	id-ecdsa-with-shake128	[EDNOTE: THIS RFC]
TBD4	id-ecdsa-with-shake256	[EDNOTE: THIS RFC]

IANA is also requested to update the Hash Function Textual Names Registry [Hash-Texts] with two additional entries for SHAKE128 and SHAKE256:

Hash Function Name	OID	Reference
shake128	2.16.840.1.101.3.4.2.11	[EDNOTE: THIS RFC]
shake256	2.16.840.1.101.3.4.2.12	[EDNOTE: THIS RFC]

7. Security Considerations

This document updates [RFC3279]. The security considerations section of that document applies to this specification as well.

NIST has defined appropriate use of the hash functions in terms of the algorithm strengths and expected time frames for secure use in Special Publications (SPs) [SP800-78-4] and [SP800-107]. These documents can be used as guides to choose appropriate key sizes for various security scenarios.

SHAKE128 with output length of 256-bits offers 128-bits of collision and preimage resistance. Thus, SHAKE128 OIDs in this specification are RECOMMENDED with 2048 (112-bit security) or 3072-bit (128-bit security) RSA modulus or curves with group order of 256-bits (128-bit security). SHAKE256 with 512-bits output length offers 256-bits of collision and preimage resistance. Thus, the SHAKE256 OIDs in this specification are RECOMMENDED with 4096-bit RSA modulus or higher or curves with group order of at least 521-bits (256-bit security). Note that we recommended 4096-bit RSA because we would need 15360-bit modulus for 256-bits of security which is impractical for today's technology.

8. Acknowledgements

We would like to thank Sean Turner, Jim Schaad and Eric Rescorla for their valuable contributions to this document.

The authors would like to thank Russ Housley for his guidance and very valuable contributions with the ASN.1 module.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[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, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/info/rfc4055>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SHA3] National Institute of Standards and Technology (NIST), "SHA-3 Standard - Permutation-Based Hash and Extendable-Output Functions FIPS PUB 202", August 2015, <https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions>.

9.2. Informative References

[Hash-Texts]
IANA, "Hash Function Textual Names", July 2017, <https://www.iana.org/assignments/hash-function-text-names/hash-function-text-names.xhtml>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", May 2009, <http://www.secg.org/sec1-v2.pdf>.

[SMI-PKIX]
IANA, "SMI Security for PKIX Algorithms", March 2019, <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.6>.

[SP800-107]
National Institute of Standards and Technology (NIST), "SP800-107: Recommendation for Applications Using Approved Hash Algorithms", May 2014, <https://csrc.nist.gov/csrc/media/publications/sp/800-107/rev-1/final/documents/draft_revised_sp800-107.pdf>.

[SP800-78-4]
National Institute of Standards and Technology (NIST), "SP800-78-4: Cryptographic Algorithms and Key Sizes for Personal Identity Verification", May 2014, <https://csrc.nist.gov/csrc/media/publications/sp/800-78/4/final/documents/sp800_78-4_revised_draft.pdf>.

[X9.62] American National Standard for Financial Services (ANSI), "X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA)", November 2005.

Appendix A. ASN.1 module

This appendix includes the ASN.1 module for SHAKEs in X.509. This module does not come from any existing RFC.

```
 PKIXAlgsForSHAKE-2019 { iso(1) identified-organization(3) dod(6)
   internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
   id-mod-pkix1-shakes-2019(TBD) }

 DEFINITIONS EXPLICIT TAGS ::=

 BEGIN

 -- EXPORTS ALL;

 IMPORTS

 -- FROM [RFC5912]

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

 -- FROM [RFC5912]

 RSAPublicKey, rsaEncryption, pk-rsa, pk-ec,
 CURVE, id-ecPublicKey, ECPoint, ECParameters, ECDSA-Sig-Value
 FROM PKIXAlgs-2009 { iso(1) identified-organization(3) dod(6)
      internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-algorithms2008-02(56) }
;

 --
 -- Message Digest Algorithms (mda-)
 --
 DigestAlgorithms DIGEST-ALGORITHM ::= {
   -- This expands DigestAlgorithms from [RFC5912]
   mda-shake128   |
   mda-shake256,
   ...
 }
```

```
    --
    -- One-Way Hash Functions
    --

    -- SHAKE128
    mda-shake128 DIGEST-ALGORITHM ::= {
      IDENTIFIER id-shake128  -- with output length 32 bytes.
    }
    id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                        us(840) organization(1) gov(101)
                                        csor(3) nistAlgorithm(4)
                                        hashAlgs(2) 11 }

    -- SHAKE256
    mda-shake256 DIGEST-ALGORITHM ::= {
      IDENTIFIER id-shake256  -- with output length 64 bytes.
    }
    id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                        us(840) organization(1) gov(101)
                                        csor(3) nistAlgorithm(4)
                                        hashAlgs(2) 12 }

    --
    -- Public Key (pk-) Algorithms
    --
    PublicKeys PUBLIC-KEY ::= {
      -- This expands PublicKeys from [RFC5912]
      pk-rsaSSA-PSS-SHAKE128 |
      pk-rsaSSA-PSS-SHAKE256,
      ...
    }

    -- The hashAlgorithm is mda-shake128
    -- The maskGenAlgorithm is id-shake128
    -- Mask Gen Algorithm is SHAKE128 with output length
    -- (8*ceil((n-1)/8) - 264) bits, where n is the RSA
    -- modulus in bits.
    -- The saltLength is 32. The trailerField is 1.
    pk-rsaSSA-PSS-SHAKE128 PUBLIC-KEY ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE128
      KEY RSAPublicKey
      PARAMS ARE absent
      -- Private key format not in this module --
      CERT-KEY-USAGE { nonRepudiation, digitalSignature,
                       keyCertSign, cRLSign }
    }

    -- The hashAlgorithm is mda-shake256
```

```
    -- The maskGenAlgorithm is id-shake256
    -- Mask Gen Algorithm is SHAKE256 with output length
    -- (8*ceil((n-1)/8) - 520)-bits, where n is the RSA
    -- modulus in bits.
    -- The saltLength is 64. The trailerField is 1.
    pk-rsaSSA-PSS-SHAKE256 PUBLIC-KEY ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE256
      KEY RSAPublicKey
      PARAMS ARE absent
      -- Private key format not in this module --
      CERT-KEY-USAGE { nonRepudiation, digitalSignature,
                       keyCertSign, cRLSign }
    }

    --
    -- Signature Algorithms (sa-)
    --
    SignatureAlgs SIGNATURE-ALGORITHM ::= {
      -- This expands SignatureAlgorithms from [RFC5912]
      sa-rsassapssWithSHAKE128 |
      sa-rsassapssWithSHAKE256 |
      sa-ecdsaWithSHAKE128 |
      sa-ecdsaWithSHAKE256,
      ...
    }

    --
    -- SMIME Capabilities (sa-)
    --
    SMimeCaps SMIME-CAPS ::= {
      -- The expands SMimeCaps from [RFC5912]
      sa-rsassapssWithSHAKE128.&smimeCaps |
      sa-rsassapssWithSHAKE256.&smimeCaps |
      sa-ecdsaWithSHAKE128.&smimeCaps |
      sa-ecdsaWithSHAKE256.&smimeCaps,
      ...
    }

    -- RSASSA-PSS with SHAKE128
    sa-rsassapssWithSHAKE128 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE128
      PARAMS ARE absent
          -- The hashAlgorithm is mda-shake128
          -- The maskGenAlgorithm is id-shake128
          -- Mask Gen Algorithm is SHAKE128 with output length
          -- (8*ceil((n-1)/8) - 264) bits, where n is the RSA
          -- modulus in bits.
          -- The saltLength is 32. The trailerField is 1
```

```
      HASHES { mda-shake128 }
      PUBLIC-KEYS { pk-rsa | pk-rsaSSA-PSS-SHAKE128 }
      SMIME-CAPS { IDENTIFIED BY id-RSASSA-PSS-SHAKE128 }
    }
    id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6)
            TBD1 }

    -- RSASSA-PSS with SHAKE256
    sa-rsassapssWithSHAKE256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE256
      PARAMS ARE absent
          -- The hashAlgorithm is mda-shake256
          -- The maskGenAlgorithm is id-shake256
          -- Mask Gen Algorithm is SHAKE256 with output length
          -- (8*ceil((n-1)/8) - 520)-bits, where n is the
          -- RSA modulus in bits.
          -- The saltLength is 64. The trailerField is 1.
     HASHES { mda-shake256 }
     PUBLIC-KEYS { pk-rsa | pk-rsaSSA-PSS-SHAKE256 }
     SMIME-CAPS { IDENTIFIED BY id-RSASSA-PSS-SHAKE256 }
    }
    id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6)
            TBD2 }

    -- ECDSA with SHAKE128
    sa-ecdsaWithSHAKE128 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-ecdsa-with-shake128
      VALUE ECDSA-Sig-Value
      PARAMS ARE absent
      HASHES { mda-shake128 }
      PUBLIC-KEYS { pk-ec }
      SMIME-CAPS { IDENTIFIED BY id-ecdsa-with-shake128 }
    }
    id-ecdsa-with-shake128 OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6)
            TBD3 }

    -- ECDSA with SHAKE256
    sa-ecdsaWithSHAKE256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-ecdsa-with-shake256
      VALUE ECDSA-Sig-Value
      PARAMS ARE absent
      HASHES { mda-shake256 }
```

```
      PUBLIC-KEYS { pk-ec }
      SMIME-CAPS { IDENTIFIED BY id-ecdsa-with-shake256 }
    }
    id-ecdsa-with-shake256 OBJECT IDENTIFIER  ::=  { iso(1)
            identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) algorithms(6)
            TBD4 }

    END
```

Authors' Addresses

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Quynh Dang
NIST
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
USA

Email: quynh.dang@nist.gov

INTERNET-DRAFT
Internet Engineering Task Force
Intended Status: Proposed Standard
Updates: 5280 (once approved)
Expires: 12 April 2018

R. Housley
Vigil Security

12 October 2017

Internationalization Updates to RFC 5280
draft-ietf-lamps-rfc5280-i18n-update-04

Abstract

These updates to RFC 5280 provide alignment with the 2008 specification for Internationalized Domain Names (IDNs) and add support for Internationalized Email Addresses in X.509 Certificates.

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/1id-abstracts.html

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

Copyright and License Notice

1. Introduction

This document updates RFC 5280 [RFC5280]. The Introduction in Section 1, the Name Constraints certificate extension discussion in Section 4.2.1.10, and the Processing Rules for Internationalized Names in Section 7 are updated to provide alignment with the 2008 specification for Internationalized Domain Names (IDNs) and add support for Internationalized Email Addresses in X.509 Certificates.

An IDN in Unicode (native character) form contains at least one U-label [RFC5890]. With one exception, IDNs are carried in certificates in ACE-encoded form. That is, all U-labels within an IDN are converted to A-labels. Conversion of an U-label to an A-label is described in [RFC5891].

The GeneralName structure supports many different names forms, including otherName for extensibility. [ID.lamps-eai-addresses] specifies the SmtpUTF8Mailbox for Internationalized Email addresses, which include IDNs with U-labels.

Note that Internationalized Domain Names in Applications specifications published in 2003 (IDNA2003) [RFC3490] and 2008 (IDNA2008) [RFC5890] both refer to the Punycode Algorithm for conversion [RFC3492].

1.1. Terminology

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Updates

This section provides updates to several paragraphs of RFC 5280 [RFC5280]. For clarity, if the entire section is not replaced, then the original text and the replacement text are shown.

2.1. Update in Section 1, Introduction

This update provides references for IDNA2008.

OLD

* Enhanced support for internationalized names is specified in Section 7, with rules for encoding and comparing Internationalized Domain Names, Internationalized Resource Identifiers (IRIs), and distinguished names. These rules are aligned with comparison rules established in current RFCs, including [RFC3490], [RFC3987], and [RFC4518].

NEW

* Enhanced support for internationalized names is specified in Section 7, with rules for encoding and comparing Internationalized Domain Names, Internationalized Resource Identifiers (IRIs), and distinguished names. These rules are aligned with comparison rules established in current RFCs, including [RFC3987], [RFC4518], [RFC5890], and [RFC5891].

2.2. Update in Section 4.2.1.10, Name Constraints

This update removes the ability to include constraints for a particular mailbox. This capability was not used, and removing it allows name constraints to apply to email addresses in rfc822Name and SmtpUTF8Mailbox [ID.lamps-eai-addresses] within otherName.

OLD

A name constraint for Internet mail addresses MAY specify a particular mailbox, all addresses at a particular host, or all mailboxes in a domain. To indicate a particular mailbox, the

constraint is the complete mail address. For example, "root@example.com" indicates the root mailbox on the host "example.com". To indicate all Internet mail addresses on a particular host, the constraint is specified as the host name. For example, the constraint "example.com" is satisfied by any mail address at the host "example.com". To specify any address within a domain, the constraint is specified with a leading period (as with URIs). For example, ".example.com" indicates all the Internet mail addresses in the domain "example.com", but not Internet mail addresses on the host "example.com".

NEW

A name constraint for Internet mail addresses MAY specify all addresses at a particular host or all mailboxes in a domain. To indicate all Internet mail addresses on a particular host, the constraint is specified as the host name. For example, the constraint "example.com" is satisfied by any mail address at the host "example.com". To specify any address within a domain, the constraint is specified with a leading period (as with URIs). For example, ".example.com" indicates all the Internet mail addresses in the domain "example.com", but not Internet mail addresses on the host "example.com".

2.3. Update in Section 7.2, IDNs in GeneralName

This update aligns with IDNA2008. Since all of Section 7.2 is replaced, the OLD text is not provided.

NEW

Internationalized Domain Names (IDNs) may be included in certificates and CRLs in the subjectAltName and issuerAltName extensions, name constraints extension, authority information access extension, subject information access extension, CRL distribution points extension, and issuing distribution point extension. Each of these extensions uses the GeneralName type; one choice in GeneralName is the dNSName field, which is defined as type IA5String.

IA5String is limited to the set of ASCII characters. To accommodate internationalized domain names U-labels are converted to A-labels. The A-label is the encoding of the U-label according to the Punycode algorithm [RFC3492] with the ACE prefix "xn--" added at the beginning of the string.

When comparing DNS names for equality, conforming implementations MUST perform a case-insensitive exact match on the entire DNS name. When evaluating name constraints, conforming implementations MUST

perform a case-insensitive exact match on a label-by-label basis. As noted in Section 4.2.1.10, any DNS name that may be constructed by adding labels to the left-hand side of the domain name given as the constraint is considered to fall within the indicated subtree.

Implementations SHOULD convert IDNs to Unicode before display. Specifically, conforming implementations convert A-labels to U-labels for display.

Implementation consideration: There are increased memory requirements for IDNs. An IDN ACE label will begin with the four additional characters "xn--", and an IDN can require as many as five ASCII characters to specify a single international character.

2.3. Update in Section 7.3, IDNs in Distinguished Names

This update aligns with IDNA2008.

OLD

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.

NEW

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 convert all U-labels to A-labels.

2.4. Update in Section 7.5, Internationalized Electronic Mail Addresses

This update aligns with IDNA2008 and [ID.lamps-eai-addresses]. Since all of Section 7.5 is replaced, the OLD text is not provided.

NEW

Electronic Mail addresses may be included in certificates and CRLs in the subjectAltName and issuerAltName extensions, name constraints extension, authority information access extension, subject information access extension, issuing distribution point extension, or CRL distribution points extension. Each of these extensions uses the GeneralName construct. If the email address includes an IDN but the local-part of the email address can be represented in ASCII, then the email address is placed in the rfc822Name choice of GeneralName, which is defined as type IA5String. If the local-part of the internationalized email address cannot be represented in ASCII, then the internationalized email address is placed in the otherName choice of GeneralName using the conventions in [ID.lamps-eai-addresses].

7.5.1. Local-part Contains Only ASCII Characters

Where the host-part contains an IDN, conforming implementations MUST convert all U-labels to A-labels.

Two email addresses are considered to match if:

1) the local-part of each name is an exact match, AND

2) the host-part of each name matches using a case-insensitive ASCII comparison.

Implementations SHOULD convert the host-part of internationalized email addresses specified in these extensions to Unicode before display. Specifically, conforming implementations convert A-labels to U-labels for display.

7.5.2. Local-part Contains Non-ASCII Characters

When the local-part contains non-ASCII character, conforming implementations MUST place the internationalized email address in the SmtpUTF8Mailbox within the otherName choice of GeneralName as specified in Section 3 of [ID.lamps-eai-addresses]. Note that the UTF8 encoding of the internationalized email address MUST NOT contain a Byte-Order-Mark (BOM) [RFC3629] to aid comparison.

The comparison of two internationalized email addresses is specified in Section 5 of [ID.lamps-eai-addresses].

Implementations SHOULD convert the host-part of internationalized email addresses specified in these extensions to Unicode before display. Specifically, conforming implementations convert A-labels to U-labels for display.

3. Security Considerations

Conforming CAs SHOULD ensure that IDNs are valid. This can be done by validating all code points according to IDNA2008 [RFC5892]. Failure to use valid A-labels and valid U-labels may yield a domain name that cannot be correctly represented in the Domain Name System (DNS). In addition, the CA/Browser Forum offers some guidance regarding internal server names in certificates [CABF].

4. IANA Considerations

No IANA registries are changed by this update.

5. Normative References

[ID.lamps-eai-addresses] Melnikov, A. (Ed.) and W. Chuang (Ed.), "Internationalized Email Addresses in X.509 certificates", September 2017, <http://www.ietf.org/id/draft-ietf-lamps-eai-addresses>, work-in-progress.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <http://www.rfc-editor.org/info/rfc3492>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <http://www.rfc-editor.org/info/rfc3987>.

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation", RFC 4518, DOI 10.17487/RFC4518, June 2006, <http://www.rfc-editor.org/info/rfc4518>.

[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, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <http://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <http://www.rfc-editor.org/info/rfc5892>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017. <http://www.rfc-editor.org/info/rfc8174>.

6. Informative References

[CABF] CA/Browser Forum, "Internal Server Names and IP Address Requirements for SSL", Version 1.0, June 2012, <https://cabforum.org/internal-names/>

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, DOI 10.17487/RFC3490, March 2003, <http://www.rfc-editor.org/info/rfc3490>.

Acknowledgements

Thanks to Alexey Melnikov for the encouragement to write this update. Thanks to John Klensin and Patrik Falstrom for confirming many of the details in this update. Thanks to Ben Campbell, Wei Chuang, Spencer Dawkins, Phillip Hallam-Baker, Warren Kumari, Alexey Melnikov, Adam Roach, Tim Ruehsen, and Sean Turner for their careful review and comments.

Authors' Address

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: housley@vigilsec.com

LAMPS
Internet-Draft
Obsoletes: 5750 (if approved)
Intended status: Standards Track
Expires: March 8, 2019

J. Schaad
August Cellars
B. Ramsdell
Brute Squad Labs, Inc.
S. Turner
sn3rd

September 4, 2018

Secure/Multipurpose Internet Mail Extensions (S/ MIME) Version 4.0 Certificate Handling
draft-ietf-lamps-rfc5750-bis-08

Abstract

This document specifies conventions for X.509 certificate usage by Secure/Multipurpose Internet Mail Extensions (S/MIME) v4.0 agents. S/MIME provides a method to send and receive secure MIME messages, and certificates are an integral part of S/MIME agent processing. S/MIME agents validate certificates as described in RFC 5280, the Internet X.509 Public Key Infrastructure Certificate and CRL Profile. S/MIME agents must meet the certificate processing requirements in this document as well as those in RFC 5280. This document obsoletes RFC 5750.

Contributing to this document

The source for this draft is being maintained in GitHub. Suggested changes should be submitted as pull requests at <https://github.com/lamps-wg/smime>. Instructions are on that page as well. Editorial changes can be managed in GitHub, but any substantial issues need to be discussed on the LAMPS mailing list.

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 https://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 March 8, 2019.

Table of Contents

1. Introduction

S/MIME (Secure/Multipurpose Internet Mail Extensions) v4.0, described in [I-D.ietf-lamps-rfc5751-bis], provides a method to send and receive secure MIME messages. Before using a public key to provide security services, the S/MIME agent MUST verify that the public key is valid. S/MIME agents MUST use PKIX certificates to validate public keys as described in the Internet X.509 Public Key Infrastructure (PKIX) Certificate and CRL Profile [RFC5280]. S/MIME agents MUST meet the certificate processing requirements documented in this document in addition to those stated in [RFC5280].

This specification is compatible with the Cryptographic Message Syntax (CMS) RFC 5652 [RFC5652] in that it uses the data types defined by CMS. It also inherits all the varieties of architectures for certificate-based key management supported by CMS.

This document obsoletes [RFC5750]. The most significant changes revolve around changes in recommendations around the cryptographic algorithms used by the specification. More details can be found in Section 1.6.

1.1. Definitions

For the purposes of this document, the following definitions apply.

ASN.1: Abstract Syntax Notation One, as defined in ITU-T X.680 [X.680].

Attribute certificate (AC): An X.509 AC is a separate structure from a subject's public key X.509 certificate. A subject may have

multiple X.509 ACs associated with each of its public key X.509 certificates. Each X.509 AC binds one or more attributes with one of the subject's public key X.509 certificates. The X.509 AC syntax is defined in [RFC5755].

Certificate: A type that binds an entity's name to a public key with a digital signature. This type is defined in the Internet X.509 Public Key Infrastructure (PKIX) Certificate and CRL Profile [RFC5280]. This type also contains the distinguished name of the certificate issuer (the signer), an issuer-specific serial number, the issuer's signature algorithm identifier, a validity period, and extensions also defined in that document.

Certificate Revocation List (CRL): A type that contains information about certificates whose validity an issuer has revoked. The information consists of an issuer name, the time of issue, the next scheduled time of issue, a list of certificate serial numbers and their associated revocation times, and extensions as defined in [RFC5280]. The CRL is signed by the issuer. The type intended by this specification is the one defined in [RFC5280].

Receiving agent: Software that interprets and processes S/MIME CMS objects, MIME body parts that contain CMS objects, or both.

Sending agent: Software that creates S/MIME CMS objects, MIME body parts that contain CMS objects, or both.

S/MIME agent: User software that is a receiving agent, a sending agent, or both.

1.2. Conventions Used 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

We define the additional requirement levels:

SHOULD+ This term means the same as SHOULD. However, the authors expect that a requirement marked as SHOULD+ will be promoted at some future time to be a MUST.

SHOULD- This term means the same as SHOULD. However, the authors expect that a requirement marked as SHOULD- will be demoted to a MAY in a future version of this document.

MUST- This term means the same as MUST. However, the authors expect that this requirement will no longer be a MUST in a future document. Although its status will be determined at a later time, it is reasonable to expect that if a future revision of a document alters the status of a MUST-requirement, it will remain at least a SHOULD or a SHOULD-.

The term RSA in this document almost always refers to the PKCS#1 v1.5 RSA signature algorithm even when not qualified as such. There are a couple of places where it refers to the general RSA cryptographic operation; these can be determined from the context where it is used.

1.3. Compatibility with Prior Practice S/MIME

S/MIME version 4.0 agents ought to attempt to have the greatest interoperability possible with agents for prior versions of S/MIME.

S/MIME version 2 is described in RFC 2311 through RFC 2315 inclusive [SMIMEv2], S/MIME version 3 is described in RFC 2630 through RFC 2634 inclusive and RFC 5035 [SMIMEv3], and S/MIME version 3.1 is described in RFC 3850, RFC 3851, RFC 3852, RFC 2634, and RFC 5035 [SMIMEv3.1]. RFC 2311 also has historical information about the development of S/MIME.

Appendix A contains information about algorithms that were used for prior versions of S/MIME but are no longer considered to meet modern security standards. Support of these algorithms may be needed to support historic S/MIME artifacts such as messages or files, but SHOULD NOT be used for new artifacts.

1.4. Changes from S/MIME v3 to S/MIME v3.1

This section reflects the changes that were made when S/MIME v3.1 was released. The RFC2119 langauage may have superceeded in later versions.

Version 1 and version 2 CRLs MUST be supported.

Multiple certification authority (CA) certificates with the same subject and public key, but with overlapping validity periods, MUST be supported.

Version 2 attribute certificates SHOULD be supported, and version 1 attributes certificates MUST NOT be used.

The use of the MD2 digest algorithm for certificate signatures is discouraged, and security language was added.

Clarified use of email address use in certificates. Certificates that do not contain an email address have no requirements for verifying the email address associated with the certificate.

Receiving agents SHOULD display certificate information when displaying the results of signature verification.

Receiving agents MUST NOT accept a signature made with a certificate that does not have at least one of the the digitalSignature or nonRepudiation bits set.

Clarifications for the interpretation of the key usage and extended key usage extensions.

1.5. Changes from S/MIME v3.1 to S/MIME v3.2

This section reflects the changes that were made when S/MIME v3.2 was released. The RFC2119 langauage may have superceeded in later versions.

Conventions Used in This Document: Moved to Section 1.2. Added definitions for SHOULD+, SHOULD-, and MUST-.

Section 1.1: Updated ASN.1 definition and reference.

Section 1.3: Added text about v3.1 RFCs.

Section 3: Aligned email address text with RFC 5280. Updated note to indicate emailAddress IA5String upper bound is 255 characters. Added text about matching email addresses.

Section 4.2: Added text to indicate how S/MIME agents locate the correct user certificate.

Section 4.3: RSA with SHA-256 (PKCS #1 v1.5) added as MUST; DSA with SHA-256 added as SHOULD+; RSA with SHA-1, DSA with SHA-1, and RSA with MD5 changed to SHOULD-; and RSASSA-PSS with SHA-256 added as SHOULD+. Updated key sizes and changed pointer to PKIX RFCs.

Section 4.4.1: Aligned with PKIX on use of basic constraints extension in CA certificates. Clarified which extension is used to constrain end entities from using their keys to perform issuing authority operations.

Section 5: Updated security considerations.

Section 7: Moved references from Appendix B to Section 6. Updated the references.

Appendix A: Moved Appendix A to Appendix B. Added Appendix A to move S/MIME v2 Certificate Handling to Historic Status.

1.6. Changes since S/MIME 3.2

This section reflects the changes that were made when S/MIME v4.0 was released. The RFC2119 langauage may have superceeded in later versions.

Section 3: Require support for internationalized email addresses.

Section 4.3: Mandated support for ECDSA with P-256 and Ed25519. Moved algorithms with SHA-1 and MD5 to historical status. Moved DSA support to historical status. Increased lower bounds on RSA key sizes.

Appendix A: Add a new appendix for algorithms that are now considered to be historical.

2. CMS Options

The CMS message format allows for a wide variety of options in content and algorithm support. This section puts forth a number of support requirements and recommendations in order to achieve a base level of interoperability among all S/MIME implementations. Most of the CMS format for S/MIME messages is defined in [I-D.ietf-lamps-rfc5751-bis].

2.1. Certificate Revocation Lists

Receiving agents MUST support the Certificate Revocation List (CRL) format defined in [RFC5280]. If sending agents include CRLs in outgoing messages, the CRL format defined in [RFC5280] MUST be used. Receiving agents MUST support both v1 and v2 CRLs.

All agents MUST be capable of performing revocation checks using CRLs as specified in [RFC5280]. All agents MUST perform revocation status checking in accordance with [RFC5280]. Receiving agents MUST recognize CRLs in received S/MIME messages.

Agents SHOULD store CRLs received in messages for use in processing later messages.

2.2. Certificate Choices

Receiving agents MUST support v1 X.509 and v3 X.509 certificates as profiled in [RFC5280]. End-entity certificates MAY include an Internet mail address, as described in Section 3.

Receiving agents SHOULD support X.509 version 2 attribute certificates. See [RFC5755] for details about the profile for attribute certificates.

2.2.1. Historical Note about CMS Certificates

The CMS message format supports a choice of certificate formats for public key content types: PKIX, PKCS #6 extended certificates [PKCS6], and PKIX attribute certificates.

The PKCS #6 format is not in widespread use. In addition, PKIX certificate extensions address much of the same functionality and flexibility as was intended in the PKCS #6. Thus, sending and receiving agents MUST NOT use PKCS #6 extended certificates. Receiving agents MUST be able to parse and process a message containing PKCS #6 extended certificates although ignoring those certificates is expected behavior.

X.509 version 1 attribute certificates are also not widely implemented, and have been superseded with version 2 attribute certificates. Sending agents MUST NOT send version 1 attribute certificates.

2.3. CertificateSet

Receiving agents MUST be able to handle an arbitrary number of certificates of arbitrary relationship to the message sender and to each other in arbitrary order. In many cases, the certificates included in a signed message may represent a chain of certification from the sender to a particular root. There may be, however, situations where the certificates in a signed message may be unrelated and included for convenience.

Sending agents SHOULD include any certificates for the user’s public key(s) and associated issuer certificates. This increases the likelihood that the intended recipient can establish trust in the originator’s public key(s). This is especially important when sending a message to recipients that may not have access to the sender’s public key through any other means or when sending a signed message to a new recipient. The inclusion of certificates in outgoing messages can be omitted if S/MIME objects are sent within a group of correspondents that has established access to each other’s

certificates by some other means such as a shared directory or manual certificate distribution. Receiving S/MIME agents SHOULD be able to handle messages without certificates by using a database or directory lookup scheme to find them.

A sending agent SHOULD include at least one chain of certificates up to, but not including, a certification authority (CA) that it believes that the recipient may trust as authoritative. A receiving agent MUST be able to handle an arbitrarily large number of certificates and chains.

Agents MAY send CA certificates, that is, cross-certificates, self-issued certificates, and self-signed certificates. Note that receiving agents SHOULD NOT simply trust any self-signed certificates as valid CAs, but SHOULD use some other mechanism to determine if this is a CA that should be trusted. Also note that when certificates contain Digital Signature Algorithm (DSA) public keys the parameters may be located in the root certificate. This would require that the recipient possess both the end-entity certificate and the root certificate to perform a signature verification, and is a valid example of a case where transmitting the root certificate may be required.

Receiving agents MUST support chaining based on the distinguished name fields. Other methods of building certificate chains MAY be supported.

Receiving agents SHOULD support the decoding of X.509 attribute certificates included in CMS objects. All other issues regarding the generation and use of X.509 attribute certificates are outside of the scope of this specification. One specification that addresses attribute certificate use is defined in [RFC3114].

3. Using Distinguished Names for Internet Mail

End-entity certificates MAY contain an Internet mail address. Email addresses restricted to 7-bit ASCII characters use the pkcs-9-at-emailAddress OID (see below) and are encoded as described in Section 4.2.1.6 of [RFC5280]. Internationalized Email address names use the OID defined in [I-D.ietf-lamps-eai-addresses] and are encoded as described there. The email address SHOULD be in the subjectAltName extension, and SHOULD NOT be in the subject distinguished name.

Receiving agents MUST recognize and accept certificates that contain no email address. Agents are allowed to provide an alternative mechanism for associating an email address with a certificate that does not contain an email address, such as through the use of the

agent's address book, if available. Receiving agents MUST recognize both ASCII and internationalized email addresses in the subjectAltName field. Receiving agents MUST recognize email addresses in the Distinguished Name field in the PKCS #9 [RFC2985] emailAddress attribute:

```
pkcs-9-at-emailAddress OBJECT IDENTIFIER ::=
 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 1 }
```

Note that this attribute MUST be encoded as IA5String and has an upper bound of 255 characters. Comparing of email addresses is fraught with peril. [I-D.ietf-lamps-eai-addresses] defines the procedure for doing comparison of Internationalized email addresses. For ASCII email addresses the domain component (right-hand side of the '@') MUST be compared using a case-insensitive function. The local name component (left-hand side of the '@') SHOULD be compared using a case-insensitive function. Some localities may perform other transformations on the local name component before doing the comparison, however an S/MIME client cannot know what specific localities do.

Sending agents SHOULD make the address in the From or Sender header in a mail message match an Internet mail address in the signer's certificate. Receiving agents MUST check that the address in the From or Sender header of a mail message matches an Internet mail address in the signer's certificate, if mail addresses are present in the certificate. A receiving agent SHOULD provide some explicit alternate processing of the message if this comparison fails; this might be done by displaying or logging a message that shows the recipient the mail addresses in the certificate or other certificate details.

A receiving agent SHOULD display a subject name or other certificate details when displaying an indication of successful or unsuccessful signature verification.

All subject and issuer names MUST be populated (i.e., not an empty SEQUENCE) in S/MIME-compliant X.509 certificates, except that the subject distinguished name (DN) in a user's (i.e., end-entity) certificate MAY be an empty SEQUENCE in which case the subjectAltName extension will include the subject's identifier and MUST be marked as critical.

4. Certificate Processing

S/MIME agents need to provide some certificate retrieval mechanism in order to gain access to certificates for recipients of digital envelopes. There are many ways to implement certificate retrieval

mechanisms. [X.500] directory service is an excellent example of a certificate retrieval-only mechanism that is compatible with classic X.500 Distinguished Names. The IETF has published [RFC8162] which describes an experimental protocol to retrieve certificates from the Domain Name System (DNS). Until such mechanisms are widely used, their utility may be limited by the small number of the correspondent's certificates that can be retrieved. At a minimum, for initial S/MIME deployment, a user agent could automatically generate a message to an intended recipient requesting the recipient's certificate in a signed return message.

Receiving and sending agents SHOULD also provide a mechanism to allow a user to "store and protect" certificates for correspondents in such a way so as to guarantee their later retrieval. In many environments, it may be desirable to link the certificate retrieval/ storage mechanisms together in some sort of certificate database. In its simplest form, a certificate database would be local to a particular user and would function in a similar way as an "address book" that stores a user's frequent correspondents. In this way, the certificate retrieval mechanism would be limited to the certificates that a user has stored (presumably from incoming messages). A comprehensive certificate retrieval/storage solution might combine two or more mechanisms to allow the greatest flexibility and utility to the user. For instance, a secure Internet mail agent might resort to checking a centralized certificate retrieval mechanism for a certificate if it cannot be found in a user's local certificate storage/retrieval database.

Receiving and sending agents SHOULD provide a mechanism for the import and export of certificates, using a CMS certs-only message. This allows for import and export of full certificate chains as opposed to just a single certificate. This is described in [RFC5751].

Agents MUST handle multiple valid certification authority (CA) certificates containing the same subject name and the same public keys but with overlapping validity intervals.

4.1. Certificate Revocation Lists

In general, it is always better to get the latest CRL information from a CA than to get information stored in an incoming messages. A receiving agent SHOULD have access to some CRL retrieval mechanism in order to gain access to certificate revocation information when validating certification paths. A receiving or sending agent SHOULD also provide a mechanism to allow a user to store incoming certificate revocation information for correspondents in such a way so as to guarantee its later retrieval.

Receiving and sending agents SHOULD retrieve and utilize CRL information every time a certificate is verified as part of a certification path validation even if the certificate was already verified in the past. However, in many instances (such as off-line verification) access to the latest CRL information may be difficult or impossible. The use of CRL information, therefore, may be dictated by the value of the information that is protected. The value of the CRL information in a particular context is beyond the scope of this specification but may be governed by the policies associated with particular certification paths.

All agents MUST be capable of performing revocation checks using CRLs as specified in [RFC5280]. All agents MUST perform revocation status checking in accordance with [RFC5280]. Receiving agents MUST recognize CRLs in received S/MIME messages.

4.2. Certificate Path Validation

In creating a user agent for secure messaging, certificate, CRL, and certification path validation should be highly automated while still acting in the best interests of the user. Certificate, CRL, and path validation MUST be performed as per [RFC5280] when validating a correspondent's public key. This is necessary before using a public key to provide security services such as verifying a signature, encrypting a content-encryption key (e.g., RSA), or forming a pairwise symmetric key (e.g., Diffie-Hellman) to be used to encrypt or decrypt a content-encryption key.

Certificates and CRLs are made available to the path validation procedure in two ways: a) incoming messages, and b) certificate and CRL retrieval mechanisms. Certificates and CRLs in incoming messages are not required to be in any particular order nor are they required to be in any way related to the sender or recipient of the message (although in most cases they will be related to the sender). Incoming certificates and CRLs SHOULD be cached for use in path validation and optionally stored for later use. This temporary certificate and CRL cache SHOULD be used to augment any other certificate and CRL retrieval mechanisms for path validation on incoming signed messages.

When verifying a signature and the certificates that are included in the message, if a signingCertificate attribute from RFC 2634 [ESS] or a signingCertificateV2 attribute from RFC 5035 [ESS] is found in an S/MIME message, it SHALL be used to identify the signer's certificate. Otherwise, the certificate is identified in an S/MIME message, either using the issuerAndSerialNumber, which identifies the signer's certificate by the issuer's distinguished name and the

certificate serial number, or the subjectKeyIdentifier, which identifies the signer's certificate by a key identifier.

When decrypting an encrypted message, if a SMIMEEncryptionKeyPreference attribute is found in an encapsulating SignedData, it SHALL be used to identify the originator's certificate found in OriginatorInfo. See [RFC5652] for the CMS fields that reference the originator's and recipient's certificates.

4.3. Certificate and CRL Signing Algorithms and Key Sizes

Certificates and Certificate Revocation Lists (CRLs) are signed by the certificate issuer. Receiving agents:

- MUST support ECDSA with curve P-256 with SHA-256.

- MUST support EdDSA with curve 25519 using PureEdDSA mode.

- MUST- support RSA PKCS#1 v1.5 with SHA-256.

- SHOULD support RSASSA-PSS with SHA-256.

Implementations SHOULD use deterministic generation for the parameter 'k' for ECDSA as outlined in [RFC6979]. EdDSA is defined to generate this parameter deterministically.

The following are the RSA and RSASSA-PSS key size requirements for S/MIME receiving agents during certificate and CRL signature verification:

```
        key size <= 2047 : SHOULD NOT (see Historic Considerations)
2048 <= key size <= 4096 : MUST (see Security Considerations)
4096 <  key size         : MAY  (see Security Considerations)
```

The signature algorithm object identifiers for RSA PKCS#1 v1.5 and RSASSA-PSS with SHA-256 using 1024-bit through 3072-bit public keys are specified in [RFC4055] and the signature algorithm definition is found in [FIPS186-2] with Change Notice 1.

The signature algorithm object identifiers for RSA PKCS#1 v1.5 and RSASSA-PSS with SHA-256 using 4096-bit public keys are specified in [RFC4055] and the signature algorithm definition is found in [RFC3447].

For RSASSA-PSS with SHA-256 see [RFC4056].

For ECDSA see [RFC5758] and [RFC6090]. The first reference provides the signature algorithm's object identifier and the second provides

the signature algorithm's definition. Curves other than curve P-256 MAY be used as well.

For EdDSA see [I-D.ietf-curdle-pkix] and [RFC8032]. The first reference provides the signature algorithm's object identifier and the second provides the signature algorithm's definition. Other curves than curve 25519 MAY be used as well.

4.4. PKIX Certificate Extensions

PKIX describes an extensible framework in which the basic certificate information can be extended and describes how such extensions can be used to control the process of issuing and validating certificates. The LAMPS Working Group has ongoing efforts to identify and create extensions that have value in particular certification environments. Further, there are active efforts underway to issue PKIX certificates for business purposes. This document identifies the minimum required set of certificate extensions that have the greatest value in the S/MIME environment. The syntax and semantics of all the identified extensions are defined in [RFC5280].

Sending and receiving agents MUST correctly handle the basic constraints, key usage, authority key identifier, subject key identifier, and subject alternative names certificate extensions when they appear in end-entity and CA certificates. Some mechanism SHOULD exist to gracefully handle other certificate extensions when they appear in end-entity or CA certificates.

Certificates issued for the S/MIME environment SHOULD NOT contain any critical extensions (extensions that have the critical field set to TRUE) other than those listed here. These extensions SHOULD be marked as non-critical unless the proper handling of the extension is deemed critical to the correct interpretation of the associated certificate. Other extensions may be included, but those extensions SHOULD NOT be marked as critical.

Interpretation and syntax for all extensions MUST follow [RFC5280], unless otherwise specified here.

4.4.1. Basic Constraints

The basic constraints extension serves to delimit the role and position that an issuing authority or end-entity certificate plays in a certification path.

For example, certificates issued to CAs and subordinate CAs contain a basic constraints extension that identifies them as issuing authority certificates. End-entity certificates contain the key usage

extension that restrains end-entities from using the key when performing issuing authority operations (see Section 4.4.2).

As per [RFC5280], certificates MUST contain a basicConstraints extension in CA certificates, and SHOULD NOT contain that extension in end-entity certificates.

4.4.2. Key Usage Certificate Extension

The key usage extension serves to limit the technical purposes for which a public key listed in a valid certificate may be used. Issuing authority certificates may contain a key usage extension that restricts the key to signing certificates, certificate revocation lists, and other data.

For example, a certification authority may create subordinate issuer certificates that contain a key usage extension that specifies that the corresponding public key can be used to sign end user certificates and sign CRLs.

If a key usage extension is included in a PKIX certificate, then it MUST be marked as critical.

S/MIME receiving agents MUST NOT accept the signature of a message if it was verified using a certificate that contains the key usage extension without at least one of the digitalSignature or nonRepudiation bits set. Sometimes S/MIME is used as a secure message transport for applications beyond interpersonal messaging; in such cases, the S/MIME-enabled application can specify additional requirements concerning the digitalSignature or nonRepudiation bits within this extension.

If the key usage extension is not specified, receiving clients MUST presume that both the digitalSignature and nonRepudiation bits are set.

4.4.3. Subject Alternative Name

The subject alternative name extension is used in S/MIME as the preferred means to convey the email address(es) that correspond(s) to the entity for this certificate. If the local portion of the email address is ASCII, it MUST be encoded using the rfc822Name CHOICE of the GeneralName type as described in [RFC5280], Section 4.2.1.6. If the local portion of the email address is not ASCII, it MUST be encoded using the otherName CHOICE of the GeneralName type as described in [I-D.ietf-lamps-eai-addresses], Section 3. Since the SubjectAltName type is a SEQUENCE OF GeneralName, multiple email addresses MAY be present.

4.4.4. Extended Key Usage Extension

The extended key usage extension also serves to limit the technical purposes for which a public key listed in a valid certificate may be used. The set of technical purposes for the certificate therefore are the intersection of the uses indicated in the key usage and extended key usage extensions.

For example, if the certificate contains a key usage extension indicating digital signature and an extended key usage extension that includes the email protection OID, then the certificate may be used for signing but not encrypting S/MIME messages. If the certificate contains a key usage extension indicating digital signature but no extended key usage extension, then the certificate may also be used to sign but not encrypt S/MIME messages.

If the extended key usage extension is present in the certificate, then interpersonal message S/MIME receiving agents MUST check that it contains either the emailProtection or the anyExtendedKeyUsage OID as defined in [RFC5280]. S/MIME uses other than interpersonal messaging MAY require the explicit presence of the extended key usage extension or other OIDs to be present in the extension or both.

5. IANA Considertions

This document has no new IANA considerations.

6. Security Considerations

All of the security issues faced by any cryptographic application must be faced by a S/MIME agent. Among these issues are protecting the user’s private key, preventing various attacks, and helping the user avoid mistakes such as inadvertently encrypting a message for the wrong recipient. The entire list of security considerations is beyond the scope of this document, but some significant concerns are listed here.

When processing certificates, there are many situations where the processing might fail. Because the processing may be done by a user agent, a security gateway, or other program, there is no single way to handle such failures. Just because the methods to handle the failures have not been listed, however, the reader should not assume that they are not important. The opposite is true: if a certificate is not provably valid and associated with the message, the processing software should take immediate and noticeable steps to inform the end user about it.

Some of the many places where signature and certificate checking might fail include:

- no Internet mail addresses in a certificate match the sender of a message, if the certificate contains at least one mail address
- no certificate chain leads to a trusted CA
- no ability to check the CRL for a certificate
- an invalid CRL was received
- the CRL being checked is expired
- the certificate is expired
- the certificate has been revoked

There are certainly other instances where a certificate may be invalid, and it is the responsibility of the processing software to check them all thoroughly, and to decide what to do if the check fails.

It is possible for there to be multiple unexpired CRLs for a CA. If an agent is consulting CRLs for certificate validation, it SHOULD make sure that the most recently issued CRL for that CA is consulted, since an S/MIME message sender could deliberately include an older unexpired CRL in an S/MIME message. This older CRL might not include recently revoked certificates, which might lead an agent to accept a certificate that has been revoked in a subsequent CRL.

When determining the time for a certificate validity check, agents have to be careful to use a reliable time. In most cases the time used SHOULD be the current time, some exceptions to this would be:

- The time the message was received is stored in a secure manner and is used at a later time to validate the message.
- The time in a SigningTime attribute found in a counter signature attribute which has been successfully validated.

The SigningTime attribute could be deliberately set to direct the receiving agent to check a CRL that could have out-of-date revocation status for a certificate, or cause an improper result when checking the Validity field of a certificate. This could be done either by the sender of the message, or an attacker which has compromised the key of the sender.

In addition to the Security Considerations identified in [RFC5280], caution should be taken when processing certificates that have not first been validated to a trust anchor. Certificates could be manufactured by untrusted sources for the purpose of mounting denial of service or other attacks. For example, keys selected to require excessive cryptographic processing, or extensive lists of CRL Distribution Point (CDP) and/or Authority Information Access (AIA) addresses in the certificate, could be used to mount denial-of-service attacks. Similarly, attacker-specified CDP and/or AIA addresses could be included in fake certificates to allow the originator to detect receipt of the message even if signature verification fails.

RSA keys of less than 2048 bits are now considered by many experts to be cryptographically insecure (due to advances in computing power), and SHOULD no longer be used to sign certificates or CRLs. Such keys were previously considered secure, so processing previously received signed and encrypted mail may require processing certificates or CRLs signed with weak keys. Implementations that wish to support previous versions of S/MIME or process old messages need to consider the security risks that result from accepting certificates and CRLs with smaller key sizes (e.g., spoofed certificates) versus the costs of denial of service. If an implementation supports verification of certificates or CRLs generated with RSA and DSA keys of less than 2048 bits, it MUST warn the user. Implementers should consider providing a stronger warning for weak signatures on certificates and CRLs associated with newly received messages than the one provided for certificates and CRLs associated with previously stored messages. Server implementations (e.g., secure mail list servers) where user warnings are not appropriate SHOULD reject messages with weak cryptography.

If an implementation is concerned about compliance with National Institute of Standards and Technology (NIST) key size recommendations, then see [SP800-57].

7. References

7.1. Normative References

[FIPS186-2]
National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS) [With Change Notice 1]", Federal Information Processing Standards Publication 186-2, January 2000.

[FIPS186-3]
National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-3, June 2009.

[I-D.ietf-lamps-eai-addresses]
Melnikov, A. and W. Chuang, "Internationalized Email Addresses in X.509 certificates", draft-ietf-lamps-eai-addresses-18 (work in progress), March 2018.

[I-D.ietf-lamps-rfc5751-bis]
Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", draft-ietf-lamps-rfc5751-bis-11 (work in progress), July 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, DOI 10.17487/RFC2634, June 1999, <https://www.rfc-editor.org/info/rfc2634>.

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, DOI 10.17487/RFC2985, November 2000, <https://www.rfc-editor.org/info/rfc2985>.

[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, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <https://www.rfc-editor.org/info/rfc3447>.

[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, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/info/rfc4055>.

[RFC4056] Schaad, J., "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", RFC 4056, DOI 10.17487/RFC4056, June 2005, <https://www.rfc-editor.org/info/rfc4056>.

[RFC5035] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, DOI 10.17487/RFC5035, August 2007, <https://www.rfc-editor.org/info/rfc5035>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5750] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, DOI 10.17487/RFC5750, January 2010, <https://www.rfc-editor.org/info/rfc5750>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/info/rfc5751>.

[RFC5755] Farrell, S., Housley, R., and S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC 5755, DOI 10.17487/RFC5755, January 2010, <https://www.rfc-editor.org/info/rfc5755>.

[RFC5758] Dang, Q., Santesson, S., Moriarty, K., Brown, D., and T. Polk, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA", RFC 5758, DOI 10.17487/RFC5758, January 2010, <https://www.rfc-editor.org/info/rfc5758>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SMIMEv3.2] "S/MIME version 3.2".

This group of documents represents S/MIME version 3.2. This set of documents are [RFC2634], [RFC5750], [[This Document]], [RFC5652], and [RFC5035].

[SMIMEv4.0] "S/MIME version 4.0".

This group of documents represents S/MIME version 4.0. This set of documents are [RFC2634], [I-D.ietf-lamps-rfc5751-bis], [[This Document]], [RFC5652], and [RFC5035].

[X.680] "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002.".

7.2. Informational References

[ESS] "Enhanced Security Services for S/ MIME".

This is the set of documents dealing with enhanced security services and refers to [RFC2634] and [RFC5035].

[I-D.ietf-curdle-pkix] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519 and X448 for use in the Internet X.509 Public Key Infrastructure", draft-ietf-curdle-pkix-10 (work in progress), May 2018.

[PKCS6] RSA Laboratories, "PKCS #6: Extended-Certificate Syntax Standard", November 1993.

[RFC2311] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, DOI 10.17487/RFC2311, March 1998, <https://www.rfc-editor.org/info/rfc2311>.

[RFC2312] Dusse, S., Hoffman, P., Ramsdell, B., and J. Weinstein, "S/MIME Version 2 Certificate Handling", RFC 2312, DOI 10.17487/RFC2312, March 1998, <https://www.rfc-editor.org/info/rfc2312>.

[RFC2313] Kaliski, B., "PKCS #1: RSA Encryption Version 1.5", RFC 2313, DOI 10.17487/RFC2313, March 1998, <https://www.rfc-editor.org/info/rfc2313>.

[RFC2314] Kaliski, B., "PKCS #10: Certification Request Syntax Version 1.5", RFC 2314, DOI 10.17487/RFC2314, March 1998, <https://www.rfc-editor.org/info/rfc2314>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <https://www.rfc-editor.org/info/rfc2315>.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, DOI 10.17487/RFC2630, June 1999, <https://www.rfc-editor.org/info/rfc2630>.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, DOI 10.17487/RFC2631, June 1999, <https://www.rfc-editor.org/info/rfc2631>.

[RFC2632] Ramsdell, B., Ed., "S/MIME Version 3 Certificate Handling", RFC 2632, DOI 10.17487/RFC2632, June 1999, <https://www.rfc-editor.org/info/rfc2632>.

[RFC2633] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, DOI 10.17487/RFC2633, June 1999, <https://www.rfc-editor.org/info/rfc2633>.

[RFC3114] Nicolls, W., "Implementing Company Classification Policy with the S/MIME Security Label", RFC 3114, DOI 10.17487/RFC3114, May 2002, <https://www.rfc-editor.org/info/rfc3114>.

[RFC3850] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, DOI 10.17487/RFC3850, July 2004, <https://www.rfc-editor.org/info/rfc3850>.

[RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <https://www.rfc-editor.org/info/rfc3851>.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, DOI 10.17487/RFC3852, July 2004, <https://www.rfc-editor.org/info/rfc3852>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <https://www.rfc-editor.org/info/rfc6194>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8162] Hoffman, P. and J. Schlyter, "Using Secure DNS to Associate Certificates with Domain Names for S/MIME", RFC 8162, DOI 10.17487/RFC8162, May 2017, <https://www.rfc-editor.org/info/rfc8162>.

[SMIMEv2] "S/MIME version v2".

This group of documents represents S/MIME version 2. This set of documents are [RFC2311], [RFC2312], [RFC2313], [RFC2314], and [RFC2315].

[SMIMEv3] "S/MIME version 3".

This group of documents represents S/MIME version 3. This set of documents are [RFC2630], [RFC2631], [RFC2632], [RFC2633], [RFC2634], and [RFC5035].

[SMIMEv3.1] "S/MIME version 3.1".

This group of documents represents S/MIME version 3.1. This set of documents are [RFC2634], [RFC3850], [RFC3851], [RFC3852], and [RFC5035].

[SP800-57] National Institute of Standards and Technology (NIST), "Special Publication 800-57: Recommendation for Key Management", August 2005.

[X.500] "ITU-T Recommendation X.500 (1997) | ISO/IEC 9594- 1:1997, Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services.".

Appendix A. Historic Considerations

A.1. Signature Algorithms and Key Sizes

There are a number of problems with validating certificates on sufficiently historic messages. For this reason it is strongly suggested that UAs treat these certificates differently from those on current messages. These problems include:

- CAs are not required to keep certificates on a CRL beyond one update after a certificate has expired. This means that unless CRLs are cached as part of the message it is not always possible to check if a certificate has been revoked. The same problems exist with OCSP responses as they may be based on a CRL rather than on the certificate database.

- RSA and DSA keys of less than 2048 bits are now considered by many experts to be cryptographically insecure (due to advances in computing power). Such keys were previously considered secure, so processing of historic certificates will often result in the use of weak keys. Implementations that wish to support previous versions of S/MIME or process old messages need to consider the security risks that result from smaller key sizes (e.g., spoofed messages) versus the costs of denial of service.

  [SMIMEv3.1] set the lower limit on suggested key sizes for creating and validation at 1024 bits. Prior to that the lower bound on key sizes was 512 bits.

- Hash functions used to validate signatures on historic messages may no longer be considered to be secure (see below). While there are not currently any known practical pre-image or second pre-image attacks against MD5 or SHA-1, the fact they are no longer considered to be collision resistant implies that the security level of any signature that is created with that these hash algorithms should also be considered as suspect.

The following algorithms have been called out for some level of support by previous S/MIME specifications:

- RSA with MD5 was dropped in [SMIMEv4.0]. MD5 is no longer considered to be secure as it is no longer collision-resistant. Details can be found in [RFC6151].

- RSA and DSA with SHA-1 were dropped in [SMIMEv4.0]. SHA-1 is no longer considered to be secure as it is no longer collision-resistant. The IETF statement on SHA-1 can be found in [RFC6194] but it is out-of-date relative to the most recent advances.

- DSA with SHA-256 support was dropped in [SMIMEv4.0]. DSA was dropped as part of a general movement from finite fields to elliptic curves. Issues have come up dealing with non-deterministic generation of the parameter 'k' (see [RFC6979]).

For 512-bit RSA with SHA-1 see [RFC3279] and [FIPS186-2] without Change Notice 1, for 512-bit RSA with SHA-256 see [RFC4055] and [FIPS186-2] without Change Notice 1.

For 512-bit DSA with SHA-1 see [RFC3279] and [FIPS186-2] without Change Notice 1, for 512-bit DSA with SHA-256 see [RFC5758] and [FIPS186-2] without Change Notice 1, for 1024-bit DSA with SHA-1 see [RFC3279] and [FIPS186-2] with Change Notice 1, for 1024-bit through 3072 DSA with SHA-256 see [RFC5758] and [FIPS186-3]. In either case, the first reference provides the signature algorithm's object identifier and the second provides the signature algorithm's definition.

Appendix B. Moving S/MIME v2 Certificate Handling to Historic Status

The S/MIME v3 [SMIMEv3], v3.1 [SMIMEv3.1], v3.2 [SMIMEv3.2], and v4.0 (this document) are backward compatible with the S/MIME v2 Certificate Handling Specification [SMIMEv2], with the exception of the algorithms (dropped RC2/40 requirement and added DSA and RSASSA-PSS requirements). Therefore, RFC 2312 [SMIMEv2] was moved to Historic status.

Appendix C. Acknowledgments

Many thanks go out to the other authors of the S/MIME v2 RFC: Steve Dusse, Paul Hoffman, and Jeff Weinstein. Without v2, there wouldn't be a v3, v3.1, v3.2 or v4.0.

A number of the members of the S/MIME Working Group have also worked very hard and contributed to this document. Any list of people is doomed to omission, and for that I apologize. In alphabetical order, the following people stand out in my mind because they made direct contributions to this document.

Bill Flanigan, Trevor Freeman, Elliott Ginsburg, Alfred Hoenes, Paul Hoffman, Russ Housley, David P. Kemp, Michael Myers, John Pawling, and Denis Pinkas.

The version 4 update to the S/MIME documents was done under the auspices of the LAMPS Working Group.

Authors' Addresses

Jim Schaad
August Cellars

Email: ietf@augustcellars.com

Blake Ramsdell
Brute Squad Labs, Inc.

Email: blaker@gmail.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

LAMPS
Internet-Draft
Obsoletes: 5751 (if approved)
Intended status: Standards Track
Expires: March 8, 2019

J. Schaad
August Cellars
B. Ramsdell
Brute Squad Labs, Inc.
S. Turner
sn3rd

September 4, 2018

Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification draft-ietf-lamps-rfc5751-bis-12

Abstract

This document defines Secure/Multipurpose Internet Mail Extensions (S/MIME) version 4.0. S/MIME provides a consistent way to send and receive secure MIME data. Digital signatures provide authentication, message integrity, and non-repudiation with proof of origin. Encryption provides data confidentiality. Compression can be used to reduce data size. This document obsoletes RFC 5751.

Contributing to this document

The source for this draft is being maintained in GitHub. Suggested changes should be submitted as pull requests at <https://github.com/lamps-wg/smime>. Instructions are on that page as well. Editorial changes can be managed in GitHub, but any substantial issues need to be discussed on the LAMPS mailing list.

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 https://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 March 8, 2019.

Table of Contents

1. Introduction

S/MIME (Secure/Multipurpose Internet Mail Extensions) provides a consistent way to send and receive secure MIME data. Based on the popular Internet MIME standard, S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures), and data confidentiality (using encryption). As a supplementary service, S/MIME provides message compression.

S/MIME can be used by traditional mail user agents (MUAs) to add cryptographic security services to mail that is sent, and to interpret cryptographic security services in mail that is received. However, S/MIME is not restricted to mail; it can be used with any transport mechanism that transports MIME data, such as HTTP or SIP. As such, S/MIME takes advantage of the object-based features of MIME and allows secure messages to be exchanged in mixed-transport systems.

Further, S/MIME can be used in automated message transfer agents that use cryptographic security services that do not require any human intervention, such as the signing of software-generated documents and the encryption of FAX messages sent over the Internet.

This document defines the version 4.0 of the S/MIME Message specification. As such this document obsoletes version 3.2 of the S/MIME Message specification [RFC5751].

1.1. Specification Overview

This document describes a protocol for adding cryptographic signature and encryption services to MIME data. The MIME standard [MIME-SPEC] provides a general structure for the content of Internet messages and allows extensions for new content-type-based applications.

This specification defines how to create a MIME body part that has been cryptographically enhanced according to the Cryptographic Message Syntax (CMS) [CMS], which is derived from PKCS #7 [RFC2315]. This specification also defines the application/pkcs7-mime media type that can be used to transport those body parts.

This document also discusses how to use the multipart/signed media type defined in [RFC1847] to transport S/MIME signed messages. multipart/signed is used in conjunction with the application/pkcs7-signature media type, which is used to transport a detached S/MIME signature.

In order to create S/MIME messages, an S/MIME agent MUST follow the specifications in this document, as well as the specifications listed in the Cryptographic Message Syntax document [CMS], [RFC3370], [RFC4056], [RFC3560], and [RFC5754].

Throughout this specification, there are requirements and recommendations made for how receiving agents handle incoming messages. There are separate requirements and recommendations for how sending agents create outgoing messages. In general, the best strategy is to "be liberal in what you receive and conservative in what you send". Most of the requirements are placed on the handling of incoming messages, while the recommendations are mostly on the creation of outgoing messages.

The separation for requirements on receiving agents and sending agents also derives from the likelihood that there will be S/MIME systems that involve software other than traditional Internet mail clients. S/MIME can be used with any system that transports MIME data. An automated process that sends an encrypted message might not be able to receive an encrypted message at all, for example. Thus, the requirements and recommendations for the two types of agents are listed separately when appropriate.

1.2. Definitions

For the purposes of this specification, the following definitions apply.

ASN.1: Abstract Syntax Notation One, as defined in ITU-T Recommendations X.680, X.681, X.682 and X.683 [ASN.1].

BER: Basic Encoding Rules for ASN.1, as defined in ITU-T Recommendation X.690 [X.690].

Certificate: A type that binds an entity’s name to a public key with a digital signature.

DER: Distinguished Encoding Rules for ASN.1, as defined in ITU-T Recommendation X.690 [X.690].

7-bit data: Text data with lines less than 998 characters long, where none of the characters have the 8th bit set, and there are no NULL characters. <CR> and <LF> occur only as part of a <CR><LF> end-of-line delimiter.

8-bit data: Text data with lines less than 998 characters, and where none of the characters are NULL characters. <CR> and <LF> occur only as part of a <CR><LF> end-of-line delimiter.

Binary data: Arbitrary data.

Transfer encoding: A reversible transformation made on data so 8-bit or binary data can be sent via a channel that only transmits 7-bit data.

Receiving agent: Software that interprets and processes S/MIME CMS objects, MIME body parts that contain CMS content types, or both.

Sending agent: Software that creates S/MIME CMS content types, MIME body parts that contain CMS content types, or both.

S/MIME agent: User software that is a receiving agent, a sending agent, or both.

Data Integrity Service: A security service that protects against unauthorized changes to data by ensuring that changes to the data are detectable. [RFC4949]

Data Confidentiality: The property that data is not disclosed to system entities unless they have been authorized to know the data. [RFC4949]

1.3. Conventions Used 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

We define the additional requirement levels:

SHOULD+ This term means the same as SHOULD. However, the authors expect that a requirement marked as SHOULD+ will be promoted at some future time to be a MUST.

SHOULD- This term means the same as SHOULD. However, the authors expect that a requirement marked as SHOULD- will be demoted to a MAY in a future version of this document.

MUST- This term means the same as MUST. However, the authors expect that this requirement will no longer be a MUST in a future document. Although its status will be determined at a later time, it is reasonable to expect that if a future revision of a document alters the status of a MUST-requirement, it will remain at least a SHOULD or a SHOULD-.

The term RSA in this document almost always refers to the PKCS#1 v1.5 RSA [RFC2313] signature or encryption algorithms even when not qualified as such. There are a couple of places where it refers to the general RSA cryptographic operation, these can be determined from the context where it is used.

1.4. Compatibility with Prior Practice of S/MIME

S/MIME version 4.0 agents ought to attempt to have the greatest interoperability possible with agents for prior versions of S/MIME. S/MIME version 2 is described in RFC 2311 through RFC 2315 inclusive [SMIMEv2], S/MIME version 3 is described in RFC 2630 through RFC 2634 inclusive and RFC 5035 [SMIMEv3], S/MIME version 3.1 is described in RFC 3850, RFC 3851, RFC 3852, RFC 2634, and RFC 5035 [SMIMEv3.1], and S/MIME version 3.2 is described in [SMIMEv3.2]. [RFC2311] also has historical information about the development of S/MIME.

1.5. Changes from S/MIME v3 to S/MIME v3.1

This section describes the changes made between S/MIME v3 and S/MIME v3.1.

The RSA public key algorithm was changed to a MUST implement. Key wrap algorithm and the Diffie-Hellman (DH) algorithm [RFC2631] changed to a SHOULD implement.

The AES symmetric encryption algorithm has been included as a SHOULD implement.

The RSA public key algorithm was changed to a MUST implement signature algorithm.

Ambiguous language about the use of "empty" SignedData messages to transmit certificates was clarified to reflect that transmission of Certificate Revocation Lists is also allowed.

The use of binary encoding for some MIME entities is now explicitly discussed.

Header protection through the use of the message/rfc822 media type has been added.

Use of the CompressedData CMS type is allowed, along with required media type and file extension additions.

1.6. Changes from S/MIME v3.1 to S/MIME v3.2

This section describes the changes made between S/MIME v3.1 and S/MIME v3.2.

Editorial changes, e.g., replaced "MIME type" with "media type", content-type with Content-Type.

Moved "Conventions Used in This Document" to Section 1.3. Added definitions for SHOULD+, SHOULD-, and MUST-.

Section 1.1 and Appendix A: Added references to RFCs for RSASSA-PSS, RSAES-OAEP, and SHA2 CMS algorithms. Added CMS Multiple Signers Clarification to CMS reference.

Section 1.2: Updated references to ASN.1 to X.680 and BER and DER to X.690.

Section 1.4: Added references to S/MIME MSG 3.1 RFCs.

Section 2.1 (digest algorithm): SHA-256 added as MUST, SHA-1 and MD5 made SHOULD-.

Section 2.2 (signature algorithms): RSA with SHA-256 added as MUST, and DSA with SHA-256 added as SHOULD+, RSA with SHA-1, DSA with SHA-1, and RSA with MD5 changed to SHOULD-, and RSASSA-PSS with SHA-256 added as SHOULD+. Also added note about what S/MIME v3.1 clients support.

Section 2.3 (key encryption): DH changed to SHOULD-, and RSAES-OAEP added as SHOULD+. Elaborated requirements for key wrap algorithm.

Section 2.5.1: Added requirement that receiving agents MUST support both GeneralizedTime and UTCTime.

Section 2.5.2: Replaced reference "sha1WithRSAEncryption" with "sha256WithRSAEncryption", "DES-3EDE-CBC" with "AES-128 CBC", and deleted the RC5 example.

Section 2.5.2.1: Deleted entire section (discussed deprecated RC2).

Section 2.7, 2.7.1, Appendix A: references to RC2/40 removed.

Section 2.7 (content encryption): AES-128 CBC added as MUST, AES-192 and AES-256 CBC SHOULD+, tripleDES now SHOULD-.

Section 2.7.1: Updated pointers from 2.7.2.1 through 2.7.2.4 to 2.7.1.1 to 2.7.1.2.

Section 3.1.1: Removed text about MIME character sets.

Section 3.2.2 and 3.6: Replaced "encrypted" with "enveloped". Update OID example to use AES-128 CBC oid.

Section 3.4.3.2: Replace "micalg" parameter for "SHA-1" with "sha-1".

Section 4: Updated reference to CERT v3.2.

Section 4.1: Updated RSA and DSA key size discussion. Moved last four sentences to security considerations. Updated reference to randomness requirements for security.

Section 5: Added IANA registration templates to update media type registry to point to this document as opposed to RFC 2311.

Section 6: Updated security considerations.

Section 7: Moved references from Appendix B to this section. Updated references. Added informational references to SMIMEv2, SMIMEv3, and SMIMEv3.1.

Appendix C: Added Appendix C to move S/MIME v2 to Historic status.

1.7. Changes for S/MIME v4.0

This section describes the changes made between S/MIME v3.2 and S/MIME v4.0.

- Add the use of AuthEnvelopedData, including defining and registering an smime-type value (Section 2.4.4 and Section 3.4).

- Update the content encryption algorithms (Section 2.7 and Section 2.7.1.2): Add AES-256 GCM, add ChaCha200-Poly1305, remove mention of AES-192 CBC, mark tripleDES as historic.

- Update the set of signature algorithms (Section 2.2): Add Edwards-curve DSA (EdDSA) and ECDSA, mark DSA as historic

- Update the set of digest algorithms (Section 2.1): Add SHA-512, mark SHA-1 as historic.

- Update the size of keys to be used for RSA encryption and RSA signing (Section 4).

- Create Appendix B which deals with considerations for dealing with historic email messages.

2. CMS Options

CMS allows for a wide variety of options in content, attributes, and algorithm support. This section puts forth a number of support requirements and recommendations in order to achieve a base level of interoperability among all S/MIME implementations. [RFC3370] and [RFC5754] provides additional details regarding the use of the cryptographic algorithms. [ESS] provides additional details regarding the use of additional attributes.

2.1. DigestAlgorithmIdentifier

The algorithms here are used for digesting the body of the message and are not the same as the digest algorithms used as part the signature algorithms. The result of this is placed in the message-digest attribute of the signed attributes. It is RECOMMENDED that the algorithm used for digesting the body of the message be of similar or greater strength than the signature algorithm.

Sending and Receiving agents:

- MUST support SHA-256.

- MUST support SHA-512.

[RFC5754] provides the details for using these algorithms with S/MIME.

2.2. SignatureAlgorithmIdentifier

There are different sets of requirements placed on receiving and sending agents. By having the different requirements, the maximum amount of interoperability is achieved as it allows for specialized protection of private key material but maximum signature validation.

Receiving agents:

- MUST support ECDSA with curve P-256 and SHA-256.
- MUST support EdDSA with curve 25519 using Pure EdDSA mode [I-D.ietf-curdle-cms-eddsa-signatures].
- MUST- support RSA PKCS#1 v1.5 with SHA-256.
- SHOULD support RSASSA-PSS with SHA-256.

Sending agents:

- MUST support at least one of the following algorithms: ECDSA with curve P-256 and SHA-256, or EdDSA with curve 25519 using PureEdDSA mode.
- MUST- support RSA PKCS#1 v1.5 with SHA-256.
- SHOULD support RSASSA-PSS with SHA-256.

See Section 4.1 for information on key size and algorithm references.

2.3. KeyEncryptionAlgorithmIdentifier

Receiving and sending agents:

- MUST support ECDH ephemeral-static mode for P-256, as specified in [RFC5753].
- MUST support ECDH ephemeral-static mode for X25519 using HKDF-256 for the KDF, as specified in [I-D.ietf-curdle-cms-ecdh-new-curves].
- MUST- support RSA Encryption, as specified in [RFC3370].
- SHOULD+ support RSAES-OAEP, as specified in [RFC3560].

When ECDH ephemeral-static is used, a key wrap algorithm is also specified in the KeyEncryptionAlgorithmIdentifier [RFC5652]. The underlying encryption functions for the key wrap and content

encryption algorithm ([RFC3370] and [RFC3565]) and the key sizes for the two algorithms MUST be the same (e.g., AES-128 key wrap algorithm with AES-128 content encryption algorithm). As both 128 and 256 bit AES modes are mandatory-to-implement as content encryption algorithms (Section 2.7), both the AES-128 and AES-256 key wrap algorithms MUST be supported when ECDH ephemeral-static is used. Recipients MAY enforce this, but MUST use the weaker of the two as part of any cryptographic strength computation it might do.

Appendix B provides information on algorithms support in older versions of S/MIME.

2.4. General Syntax

There are several CMS content types. Of these, only the Data, SignedData, EnvelopedData, AuthEnvelopedData, and CompressedData content types are currently used for S/MIME.

2.4.1. Data Content Type

Sending agents MUST use the id-data content type identifier to identify the "inner" MIME message content. For example, when applying a digital signature to MIME data, the CMS SignedData encapContentInfo eContentType MUST include the id-data object identifier and the media type MUST be stored in the SignedData encapContentInfo eContent OCTET STRING (unless the sending agent is using multipart/signed, in which case the eContent is absent, per Section 3.5.3 of this document). As another example, when applying encryption to MIME data, the CMS EnvelopedData encryptedContentInfo contentType MUST include the id-data object identifier and the encrypted MIME content MUST be stored in the EnvelopedData encryptedContentInfo encryptedContent OCTET STRING.

2.4.2. SignedData Content Type

Sending agents MUST use the SignedData content type to apply a digital signature to a message or, in a degenerate case where there is no signature information, to convey certificates. Applying a signature to a message provides authentication, message integrity, and non-repudiation of origin.

2.4.3. EnvelopedData Content Type

This content type is used to apply data confidentiality to a message. In order to distribute the symmetric key, a sender needs to have access to a public key for each intended message recipient to use this service.

2.4.4. AuthEnvelopedData Content Type

This content type is used to apply data confidentiality and message integrity to a message. This content type does not provide authentication or non-repudiation. In order to distribute the symmetric key, a sender needs to have access to a public key for each intended message recipient to use this service.

2.4.5. CompressedData Content Type

This content type is used to apply data compression to a message. This content type does not provide authentication, message integrity, non-repudiation, or data confidentiality, and is only used to reduce the message's size.

See Section 3.7 for further guidance on the use of this type in conjunction with other CMS types.

2.5. Attributes and the SignerInfo Type

The SignerInfo type allows the inclusion of unsigned and signed attributes along with a signature. These attributes can be required for processing of message (i.e. Message Digest), information the signer supplied (i.e. SMIME Capabilities) that should be processed, or attributes which are not relevant in the current situation (i.e. mlExpansionList [RFC2634] for mail viewers).

Receiving agents MUST be able to handle zero or one instance of each of the signed attributes listed here. Sending agents SHOULD generate one instance of each of the following signed attributes in each S/MIME message:

- Signing Time (Section 2.5.1 in this document)
- SMIME Capabilities (Section 2.5.2 in this document)
- Encryption Key Preference (Section 2.5.3 in this document)
- Message Digest (Section 11.2 in [RFC5652])
- Content Type (Section 11.1 in [RFC5652])

Further, receiving agents SHOULD be able to handle zero or one instance of the signingCertificate and signingCertificatev2 signed attributes, as defined in Section 5 of RFC 2634 [ESS] and Section 3 of RFC 5035 [ESS].

Sending agents SHOULD generate one instance of the signingCertificate or signingCertificatev2 signed attribute in each SignerInfo structure.

Additional attributes and values for these attributes might be defined in the future. Receiving agents SHOULD handle attributes or values that they do not recognize in a graceful manner.

Interactive sending agents that include signed attributes that are not listed here SHOULD display those attributes to the user, so that the user is aware of all of the data being signed.

2.5.1. Signing Time Attribute

The signing-time attribute is used to convey the time that a message was signed. The time of signing will most likely be created by a signer and therefore is only as trustworthy as that signer.

Sending agents MUST encode signing time through the year 2049 as UTCTime; signing times in 2050 or later MUST be encoded as GeneralizedTime. When the UTCTime CHOICE is used, S/MIME agents MUST interpret the year field (YY) as follows:

> If YY is greater than or equal to 50, the year is interpreted as 19YY; if YY is less than 50, the year is interpreted as 20YY.

Receiving agents MUST be able to process signing-time attributes that are encoded in either UTCTime or GeneralizedTime.

2.5.2. SMIME Capabilities Attribute

The SMIMECapabilities attribute includes signature algorithms (such as "sha256WithRSAEncryption"), symmetric algorithms (such as "AES-128 CBC"), authenticated symmetric algorithms (such as "AES-128 GCM") and key encipherment algorithms (such as "rsaEncryption"). The presence of an algorithm based SMIME Capability attribute in this sequence implies that the sender can deal with the algorithm as well as understand the ASN.1 structures associated with that algorithm. There are also several identifiers that indicate support for other optional features such as binary encoding and compression. The SMIMECapabilities were designed to be flexible and extensible so that, in the future, a means of identifying other capabilities and preferences such as certificates can be added in a way that will not cause current clients to break.

If present, the SMIMECapabilities attribute MUST be a SignedAttribute. CMS defines SignedAttributes as a SET OF Attribute. The SignedAttributes in a signerInfo MUST include a single instance

of the SMIMECapabilities attribute. CMS defines the ASN.1 syntax for Attribute to include attrValues SET OF AttributeValue. A SMIMECapabilities attribute MUST only include a single instance of AttributeValue. If a signature is detected as violating these requirements, the signature SHOULD be treated as failing.

The semantics of the SMIMECapabilities attribute specify a partial list as to what the client announcing the SMIMECapabilities can support. A client does not have to list every capability it supports, and need not list all its capabilities so that the capabilities list doesn't get too long. In an SMIMECapabilities attribute, the object identifiers (OIDs) are listed in order of their preference, but SHOULD be separated logically along the lines of their categories (signature algorithms, symmetric algorithms, key encipherment algorithms, etc.).

The structure of the SMIMECapabilities attribute is to facilitate simple table lookups and binary comparisons in order to determine matches. For instance, the encoding for the SMIMECapability for sha256WithRSAEncryption includes rather than omits the NULL parameter. Because of the requirement for identical encoding, individuals documenting algorithms to be used in the SMIMECapabilities attribute SHOULD explicitly document the correct byte sequence for the common cases.

For any capability, the associated parameters for the OID MUST specify all of the parameters necessary to differentiate between two instances of the same algorithm.

The OIDs that correspond to algorithms SHOULD use the same OID as the actual algorithm, except in the case where the algorithm usage is ambiguous from the OID. For instance, in an earlier specification, rsaEncryption was ambiguous because it could refer to either a signature algorithm or a key encipherment algorithm. In the event that an OID is ambiguous, it needs to be arbitrated by the maintainer of the registered SMIMECapabilities list as to which type of algorithm will use the OID, and a new OID MUST be allocated under the smimeCapabilities OID to satisfy the other use of the OID.

The registered SMIMECapabilities list specifies the parameters for OIDs that need them, most notably key lengths in the case of variable-length symmetric ciphers. In the event that there are no differentiating parameters for a particular OID, the parameters MUST be omitted, and MUST NOT be encoded as NULL. Additional values for the SMIMECapabilities attribute might be defined in the future. Receiving agents MUST handle a SMIMECapabilities object that has values that it does not recognize in a graceful manner.

Section 2.7.1 explains a strategy for caching capabilities.

2.5.3. Encryption Key Preference Attribute

The encryption key preference attribute allows the signer to unambiguously describe which of the signer's certificates has the signer's preferred encryption key. This attribute is designed to enhance behavior for interoperating with those clients that use separate keys for encryption and signing. This attribute is used to convey to anyone viewing the attribute which of the listed certificates is appropriate for encrypting a session key for future encrypted messages.

If present, the SMIMEEncryptionKeyPreference attribute MUST be a SignedAttribute. CMS defines SignedAttributes as a SET OF Attribute. The SignedAttributes in a signerInfo MUST include a single instance of the SMIMEEncryptionKeyPreference attribute. CMS defines the ASN.1 syntax for Attribute to include attrValues SET OF AttributeValue. A SMIMEEncryptionKeyPreference attribute MUST only include a single instance of AttributeValue. If a signature is detected to violate these requirements, the signature SHOULD be treated as failing.

The sending agent SHOULD include the referenced certificate in the set of certificates included in the signed message if this attribute is used. The certificate MAY be omitted if it has been previously made available to the receiving agent. Sending agents SHOULD use this attribute if the commonly used or preferred encryption certificate is not the same as the certificate used to sign the message.

Receiving agents SHOULD store the preference data if the signature on the message is valid and the signing time is greater than the currently stored value. (As with the SMIMECapabilities, the clock skew SHOULD be checked and the data not used if the skew is too great.) Receiving agents SHOULD respect the sender's encryption key preference attribute if possible. This, however, represents only a preference and the receiving agent can use any certificate in replying to the sender that is valid.

Section 2.7.1 explains a strategy for caching preference data.

2.5.3.1. Selection of Recipient Key Management Certificate

In order to determine the key management certificate to be used when sending a future CMS EnvelopedData message for a particular recipient, the following steps SHOULD be followed:

- If an SMIMEEncryptionKeyPreference attribute is found in a SignedData object received from the desired recipient, this identifies the X.509 certificate that SHOULD be used as the X.509 key management certificate for the recipient.

- If an SMIMEEncryptionKeyPreference attribute is not found in a SignedData object received from the desired recipient, the set of X.509 certificates SHOULD be searched for a X.509 certificate with the same subject name as the signer of a X.509 certificate that can be used for key management.

- Or use some other method of determining the user's key management key. If a X.509 key management certificate is not found, then encryption cannot be done with the signer of the message. If multiple X.509 key management certificates are found, the S/MIME agent can make an arbitrary choice between them.

2.6. SignerIdentifier SignerInfo Type

S/MIME v4.0 implementations MUST support both issuerAndSerialNumber and subjectKeyIdentifier. Messages that use the subjectKeyIdentifier choice cannot be read by S/MIME v2 clients.

It is important to understand that some certificates use a value for subjectKeyIdentifier that is not suitable for uniquely identifying a certificate. Implementations MUST be prepared for multiple certificates for potentially different entities to have the same value for subjectKeyIdentifier, and MUST be prepared to try each matching certificate during signature verification before indicating an error condition.

2.7. ContentEncryptionAlgorithmIdentifier

Sending and receiving agents:

- MUST support encryption and decryption with AES-128 GCM and AES-256 GCM [RFC5084].

- MUST- support encryption and decryption with AES-128 CBC [RFC3565].

- SHOULD+ support encryption and decryption with ChaCha20-Poly1305 [RFC7905].

2.7.1. Deciding Which Encryption Method to Use

When a sending agent creates an encrypted message, it has to decide which type of encryption to use. The decision process involves using information garnered from the capabilities lists included in messages received from the recipient, as well as out-of-band information such as private agreements, user preferences, legal restrictions, and so on.

Section 2.5.2 defines a method by which a sending agent can optionally announce, among other things, its decrypting capabilities in its order of preference. The following method for processing and remembering the encryption capabilities attribute in incoming signed messages SHOULD be used.

- If the receiving agent has not yet created a list of capabilities for the sender's public key, then, after verifying the signature on the incoming message and checking the timestamp, the receiving agent SHOULD create a new list containing at least the signing time and the symmetric capabilities.

- If such a list already exists, the receiving agent SHOULD verify that the signing time in the incoming message is greater than the signing time stored in the list and that the signature is valid. If so, the receiving agent SHOULD update both the signing time and capabilities in the list. Values of the signing time that lie far in the future (that is, a greater discrepancy than any reasonable clock skew), or a capabilities list in messages whose signature could not be verified, MUST NOT be accepted.

The list of capabilities SHOULD be stored for future use in creating messages.

Before sending a message, the sending agent MUST decide whether it is willing to use weak encryption for the particular data in the message. If the sending agent decides that weak encryption is unacceptable for this data, then the sending agent MUST NOT use a weak algorithm. The decision to use or not use weak encryption overrides any other decision in this section about which encryption algorithm to use.

Section 2.7.1.1 and Section 2.7.1.2 describe the decisions a sending agent SHOULD use in deciding which type of encryption will be applied to a message. These rules are ordered, so the sending agent SHOULD make its decision in the order given.

2.7.1.1. Rule 1: Known Capabilities

If the sending agent has received a set of capabilities from the recipient for the message the agent is about to encrypt, then the sending agent SHOULD use that information by selecting the first capability in the list (that is, the capability most preferred by the intended recipient) that the sending agent knows how to encrypt. The sending agent SHOULD use one of the capabilities in the list if the agent reasonably expects the recipient to be able to decrypt the message.

2.7.1.2. Rule 2: Unknown Capabilities, Unknown Version of S/MIME

If the following two conditions are met:

- the sending agent has no knowledge of the encryption capabilities of the recipient, and

- the sending agent has no knowledge of the version of S/MIME of the recipient,

then the sending agent SHOULD use AES-256 GCM because it is a stronger algorithm and is required by S/MIME v4.0. If the sending agent chooses not to use AES-256 GCM in this step, given the presumption is that a client implementing AES-GCM would do both AES-256 and AES-128, it SHOULD use AES-128 CBC.

2.7.2. Choosing Weak Encryption

Algorithms such as RC2 are considered to be weak encryption algorithms. Algorithms such as TripleDES are not state of the art and are considered to be weaker algorithms than AES. A sending agent that is controlled by a human SHOULD allow a human sender to determine the risks of sending data using a weaker encryption algorithm before sending the data, and possibly allow the human to use a stronger encryption algorithm such as AES GCM or AES CBC even if there is a possibility that the recipient will not be able to process that algorithm.

2.7.3. Multiple Recipients

If a sending agent is composing an encrypted message to a group of recipients where the encryption capabilities of some of the recipients do not overlap, the sending agent is forced to send more than one message. Please note that if the sending agent chooses to send a message encrypted with a strong algorithm, and then send the same message encrypted with a weak algorithm, someone watching the

communications channel could learn the contents of the strongly encrypted message simply by decrypting the weakly encrypted message.

3. Creating S/MIME Messages

This section describes the S/MIME message formats and how they are created. S/MIME messages are a combination of MIME bodies and CMS content types. Several media types as well as several CMS content types are used. The data to be secured is always a canonical MIME entity. The MIME entity and other data, such as certificates and algorithm identifiers, are given to CMS processing facilities that produce a CMS object. Finally, the CMS object is wrapped in MIME. The Enhanced Security Services for S/MIME [ESS] document provides descriptions of how nested, secured S/MIME messages are formatted. ESS provides a description of how a triple-wrapped S/MIME message is formatted using multipart/signed and application/pkcs7-mime for the signatures.

S/MIME provides one format for enveloped-only data, several formats for signed-only data, and several formats for signed and enveloped data. Several formats are required to accommodate several environments, in particular for signed messages. The criteria for choosing among these formats are also described.

The reader of this section is expected to understand MIME as described in [MIME-SPEC] and [RFC1847].

3.1. Preparing the MIME Entity for Signing, Enveloping, or Compressing

S/MIME is used to secure MIME entities. A MIME message is composed of a MIME header and a MIME body. The body can consist of a single part or of multiple parts. Any of these parts is designated as a MIME message part. A MIME entity can be a sub-part, sub-parts of a MIME message, or the whole MIME message with all of its sub-parts. A MIME entity that is the whole message includes only the MIME message headers and MIME body, and does not include the RFC-822 header. Note that S/MIME can also be used to secure MIME entities used in applications other than Internet mail. If protection of the RFC-822 header is required, the use of the message/rfc822 media type is explained later in this section.

The MIME entity that is secured and described in this section can be thought of as the "inside" MIME entity. That is, it is the "innermost" object in what is possibly a larger MIME message. Processing "outside" MIME entities into CMS content types is described in Section 3.2, Section 3.5, and elsewhere.

The procedure for preparing a MIME entity is given in [MIME-SPEC]. The same procedure is used here with some additional restrictions when signing. The description of the procedures from [MIME-SPEC] is repeated here, but it is suggested that the reader refer to that document for the exact procedure. This section also describes additional requirements.

A single procedure is used for creating MIME entities that are to have any combination of signing, enveloping, and compressing applied. Some additional steps are recommended to defend against known corruptions that can occur during mail transport that are of particular importance for clear-signing using the multipart/signed format. It is recommended that these additional steps be performed on enveloped messages, or signed and enveloped messages, so that the message can be forwarded to any environment without modification.

These steps are descriptive rather than prescriptive. The implementer is free to use any procedure as long as the result is the same.

Step 1. The MIME entity is prepared according to the local conventions.

Step 2. The leaf parts of the MIME entity are converted to canonical form.

Step 3. Appropriate transfer encoding is applied to the leaves of the MIME entity.

When an S/MIME message is received, the security services on the message are processed, and the result is the MIME entity. That MIME entity is typically passed to a MIME-capable user agent where it is further decoded and presented to the user or receiving application.

In order to protect outer, non-content-related message header fields (for instance, the "Subject", "To", "From", and "Cc" fields), the sending client MAY wrap a full MIME message in a message/rfc822 wrapper in order to apply S/MIME security services to these header fields. It is up to the receiving client to decide how to present this "inner" header along with the unprotected "outer" header. Given the security difference between headers, it is RECOMMENDED that client provide a distinction between header fields depending on where they are located.

When an S/MIME message is received, if the top-level protected MIME entity has a Content-Type of message/rfc822, it can be assumed that the intent was to provide header protection. This entity SHOULD be

presented as the top-level message, taking into account header merging issues as previously discussed.

3.1.1. Canonicalization

Each MIME entity MUST be converted to a canonical form that is uniquely and unambiguously representable in the environment where the signature is created and the environment where the signature will be verified. MIME entities MUST be canonicalized for enveloping and compressing as well as signing.

The exact details of canonicalization depend on the actual media type and subtype of an entity, and are not described here. Instead, the standard for the particular media type SHOULD be consulted. For example, canonicalization of type text/plain is different from canonicalization of audio/basic. Other than text types, most types have only one representation regardless of computing platform or environment that can be considered their canonical representation. In general, canonicalization will be performed by the non-security part of the sending agent rather than the S/MIME implementation.

The most common and important canonicalization is for text, which is often represented differently in different environments. MIME entities of major type "text" MUST have both their line endings and character set canonicalized. The line ending MUST be the pair of characters <CR><LF>, and the charset SHOULD be a registered charset [CHARSETS]. The details of the canonicalization are specified in [MIME-SPEC].

Note that some charsets such as ISO-2022 have multiple representations for the same characters. When preparing such text for signing, the canonical representation specified for the charset MUST be used.

3.1.2. Transfer Encoding

When generating any of the secured MIME entities below, except the signing using the multipart/signed format, no transfer encoding is required at all. S/MIME implementations MUST be able to deal with binary MIME objects. If no Content-Transfer-Encoding header field is present, the transfer encoding is presumed to be 7BIT.

As a rule, S/MIME implementations SHOULD use transfer encoding described in Section 3.1.3 for all MIME entities they secure. The reason for securing only 7-bit MIME entities, even for enveloped data that is not exposed to the transport, is that it allows the MIME entity to be handled in any environment without changing it. For example, a trusted gateway might remove the envelope, but not the

signature, of a message, and then forward the signed message on to the end recipient so that they can verify the signatures directly. If the transport internal to the site is not 8-bit clean, such as on a wide-area network with a single mail gateway, verifying the signature will not be possible unless the original MIME entity was only 7-bit data.

In the case where S/MIME implementations can determine that all intended recipients are capable of handling inner (all but the outermost) binary MIME objects, implementations SHOULD use binary encoding as opposed to a 7-bit-safe transfer encoding for the inner entities. The use of a 7-bit-safe encoding (such as base64) unnecessarily expands the message size. Implementations MAY determine that recipient implementations are capable of handling inner binary MIME entities either by interpreting the id-cap-preferBinaryInside SMIMECapabilities attribute, by prior agreement, or by other means.

If one or more intended recipients are unable to handle inner binary MIME objects, or if this capability is unknown for any of the intended recipients, S/MIME implementations SHOULD use transfer encoding described in Section 3.1.3 for all MIME entities they secure.

3.1.3. Transfer Encoding for Signing Using multipart/signed

If a multipart/signed entity is ever to be transmitted over the standard Internet SMTP infrastructure or other transport that is constrained to 7-bit text, it MUST have transfer encoding applied so that it is represented as 7-bit text. MIME entities that are 7-bit data already need no transfer encoding. Entities such as 8-bit text and binary data can be encoded with quoted-printable or base-64 transfer encoding.

The primary reason for the 7-bit requirement is that the Internet mail transport infrastructure cannot guarantee transport of 8-bit or binary data. Even though many segments of the transport infrastructure now handle 8-bit and even binary data, it is sometimes not possible to know whether the transport path is 8-bit clean. If a mail message with 8-bit data were to encounter a message transfer agent that cannot transmit 8-bit or binary data, the agent has three options, none of which are acceptable for a clear-signed message:

- The agent could change the transfer encoding; this would invalidate the signature.

- The agent could transmit the data anyway, which would most likely result in the 8th bit being corrupted; this too would invalidate the signature.

- The agent could return the message to the sender.

[RFC1847] prohibits an agent from changing the transfer encoding of the first part of a multipart/signed message. If a compliant agent that cannot transmit 8-bit or binary data encountered a multipart/signed message with 8-bit or binary data in the first part, it would have to return the message to the sender as undeliverable.

3.1.4. Sample Canonical MIME Entity

This example shows a multipart/mixed message with full transfer encoding. This message contains a text part and an attachment. The sample message text includes characters that are not ASCII and thus need to be transfer encoded. Though not shown here, the end of each line is <CR><LF>. The line ending of the MIME headers, the text, and the transfer encoded parts, all MUST be <CR><LF>.

Note that this example is not of an S/MIME message.

```
Content-Type: multipart/mixed; boundary=bar

--bar
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

=A1Hola Michael!

How do you like the new S/MIME specification?

It’s generally a good idea to encode lines that begin with
From=20because some mail transport agents will insert a greater-
than (>) sign, thus invalidating the signature.

Also, in some cases it might be desirable to encode any =20
trailing whitespace that occurs on lines in order to ensure =20
that the message signature is not invalidated when passing =20
a gateway that modifies such whitespace (like BITNET). =20

--bar
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//
jJV5bNvkZIGPIcEmI5iFd9boEgvpirHtIREEqLQRkYNoBActFBZmh9GC3C041WGq
uMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9Brn
HOxEa44b+EI=

--bar--
```

3.2. The application/pkcs7-mime Media Type

The application/pkcs7-mime media type is used to carry CMS content types including EnvelopedData, SignedData, and CompressedData. The details of constructing these entities are described in subsequent sections. This section describes the general characteristics of the application/pkcs7-mime media type.

The carried CMS object always contains a MIME entity that is prepared as described in Section 3.1 if the eContentType is id-data. Other contents MAY be carried when the eContentType contains different values. See [ESS] for an example of this with signed receipts.

Since CMS content types are binary data, in most cases base-64 transfer encoding is appropriate, in particular, when used with SMTP transport. The transfer encoding used depends on the transport through which the object is to be sent, and is not a characteristic of the media type.

Note that this discussion refers to the transfer encoding of the CMS object or "outside" MIME entity. It is completely distinct from, and unrelated to, the transfer encoding of the MIME entity secured by the CMS object, the "inside" object, which is described in Section 3.1.

Because there are several types of application/pkcs7-mime objects, a sending agent SHOULD do as much as possible to help a receiving agent know about the contents of the object without forcing the receiving agent to decode the ASN.1 for the object. The Content-Type header field of all application/pkcs7-mime objects SHOULD include the optional "smime-type" parameter, as described in the following sections.

3.2.1. The name and filename Parameters

For the application/pkcs7-mime, sending agents SHOULD emit the optional "name" parameter to the Content-Type field for compatibility with older systems. Sending agents SHOULD also emit the optional Content-Disposition field [RFC2183] with the "filename" parameter. If a sending agent emits the above parameters, the value of the parameters SHOULD be a file name with the appropriate extension:

Media Type	File Extension
application/pkcs7-mime (SignedData, EnvelopedData, AuthEnvelopedData)	.p7m
application/pkcs7-mime (degenerate SignedData certificate management message)	.p7c
application/pkcs7-mime (CompressedData)	.p7z
application/pkcs7-signature (SignedData)	.p7s

In addition, the file name SHOULD be limited to eight characters followed by a three-letter extension. The eight-character filename base can be any distinct name; the use of the filename base "smime" SHOULD be used to indicate that the MIME entity is associated with S/MIME.

Including a file name serves two purposes. It facilitates easier use of S/MIME objects as files on disk. It also can convey type information across gateways. When a MIME entity of type application/pkcs7-mime (for example) arrives at a gateway that has no special knowledge of S/MIME, it will default the entity’s media type to application/octet-stream and treat it as a generic attachment, thus losing the type information. However, the suggested filename for an attachment is often carried across a gateway. This often allows the receiving systems to determine the appropriate application to hand the attachment off to, in this case, a stand-alone S/MIME processing application. Note that this mechanism is provided as a

convenience for implementations in certain environments. A proper S/MIME implementation MUST use the media types and MUST NOT rely on the file extensions.

3.2.2. The smime-type Parameter

The application/pkcs7-mime content type defines the optional "smime-type" parameter. The intent of this parameter is to convey details about the security applied (signed or enveloped) along with information about the contained content. This specification defines the following smime-types.

Name	CMS Type	Inner Content
enveloped-data	EnvelopedData	id-data
signed-data	SignedData	id-data
certs-only	SignedData	id-data
compressed-data	CompressedData	id-data
authEnveloped-data	AuthEnvelopedData	id-data

In order for consistency to be obtained with future specifications, the following guidelines SHOULD be followed when assigning a new smime-type parameter.

1. If both signing and encryption can be applied to the content, then three values for smime-type SHOULD be assigned "signed-*", "authEnv-*", and "enveloped-*". If one operation can be assigned, then this can be omitted. Thus, since "certs-only" can only be signed, "signed-" is omitted.

2. A common string for a content OID SHOULD be assigned. We use "data" for the id-data content OID when MIME is the inner content.

3. If no common string is assigned, then the common string of "OID.<oid>" is recommended (for example, "OID.2.16.840.1.101.3.4.1.2" would be AES-128 CBC).

It is explicitly intended that this field be a suitable hint for mail client applications to indicate whether a message is "signed", "authEnveloped" or "enveloped" without having to tunnel into the CMS payload.

A registry for additional smime-type parameter values has been defined in [RFC7114].

3.3. Creating an Enveloped-Only Message

This section describes the format for enveloping a MIME entity without signing it. It is important to note that sending enveloped but not signed messages does not provide for data integrity. The Enveloped-Only structure does not support authenticated symmetric algorithmm. Use the Authenticated Enveloped structure for these algorithms. Thus, it is possible to replace ciphertext in such a way that the processed message will still be valid, but the meaning can be altered.

Step 1. The MIME entity to be enveloped is prepared according to Section 3.1.

Step 2. The MIME entity and other required data is processed into a CMS object of type EnvelopedData. In addition to encrypting a copy of the content-encryption key for each recipient, a copy of the content-encryption key SHOULD be encrypted for the originator and included in the EnvelopedData (see [RFC5652], Section 6).

Step 3. The EnvelopedData object is wrapped in a CMS ContentInfo object.

Step 4. The ContentInfo object is inserted into an application/pkcs7-mime MIME entity.

The smime-type parameter for enveloped-only messages is "enveloped-data". The file extension for this type of message is ".p7m".

A sample message would be:

```
Content-Type: application/pkcs7-mime; name=smime.p7m;
        smime-type=enveloped-data
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

MIIBHgYJKoZIhvcNAQcDoIIBDzCCAQsCAQAxgcAwgb0CAQAwJjASMRAwDgYDVQQDEw
dDYXJsUlNBAhBGNGvHgABWvBHTbi7NXXHQMA0GCSqGSIb3DQEBAQUABIGAC3EN5nGI
iJi2lsGPcP2iJ97a4e8kbKQz36zg6Z2i0yx6zYC4mZ7mX7FBs3IWg+f6KgCLx3M1eC
bWx8+MDFbbpXadCDgO8/nUkUNYeNxJtuzubGgzoyEd8Ch4H/dd9gdzTd+taTEgS0ip
dSJuNnkVY4/M652jKKHRLFf02hosdR8wQwYJKoZIhvcNAQcBMBQGCCqGSIb3DQMHBA
gtaMXpRwZRNYAgDsiSf8Z9P43LrY4OxUk660cu1lXeCSFOSOpOJ7FuVyU=
```

3.4. Creating an Authenticated Enveloped-Only Message

This section describes the format for enveloping a MIME entity without signing it. Authenticated enveloped messages provide confidentiality and data integrity. It is important to note that sending authenticated enveloped messages does not provide for proof of origination when using S/MIME. It is possible for a third party to replace ciphertext in such a way that the processed message will still be valid, but the meaning can be altered. However this is substantially more difficult than it is for an enveloped-only message as the algorithm does provide a level of authentication. Any recipient for whom the message is encrypted can replace it without detection.

Step 1. The MIME entity to be enveloped is prepared according to Section 3.1.

Step 2. The MIME entity and other required data is processed into a CMS object of type AuthEnvelopedData. In addition to encrypting a copy of the content-encryption key for each recipient, a copy of the content-encryption key SHOULD be encrypted for the originator and included in the AuthEnvelopedData (see [RFC5083]).

Step 3. The AuthEnvelopedData object is wrapped in a CMS ContentInfo object.

Step 4. The ContentInfo object is inserted into an application/pkcs7-mime MIME entity.

The smime-type parameter for authenticated enveloped-only messages is "authEnveloped-data". The file extension for this type of message is ".p7m".

A sample message would be:

```
Content-Type: application/pkcs7-mime; smime-type=authEnveloped-data;
      name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

MIIDWQYLKoZIhvcNAQkQAREgggNIMIIDRAIBADGBvjCBuwIBADAmMBIxEDAO
BgNVBAMTB0NhcmxSU0ECEEY0a8eAAFa8EdNuLs1dcdAwCwYJKoZIhvcNAQEB
BIGAgyZJo0ERTxA4xdTri5P5tVMyh0RARepTUCORZvlUbcUlaI8IpJZH3/J1
Fv6MxTRS4O/K+ZcTlQmYeWLQvwdltQdOIP3mhpqXzTnOYhTK1IDtF2zx75Lg
vE+ilpcLIzXfJB4RCBPtBWaHAof4Wb+VMQvLkk9OolX4mRSH1LPktgAwggJq
BgkqhkiG9w0BBwEwGwYJYIZIAWUDBAEGMA4EDGPizioC9OHSsnNx4oCCAj7Y
Cb8rOy8+55106newEJohC/aDgWbJhrMKzSOwa7JraXOV3HXD3NvKbl665dRx
vmDwSCNaLCRU5q8/AxQx2SvnAbM+JKcEfC/VFdd4SiHNiUECAApLku2rMi5B
WrhW/FXmx9d+cjum2BRwB3wj0q1wajdB0/kVRbQwg697dnlYyUog4vpJERjr
7KAkawZx1RMHaM18wgZjUNpCBXFS3chQi9mTBp2i2Hf5iZ8OOtTx+rCQUmI6
Jhy03vdcPCCARBjn3v0d3upZYDZddMA41CB9fKnnWFjadV1KpYwv80tqsEfx
Vo0lJQ5VtJ8MHJiBpLVKadRIZ4iH2ULC0JtN5mXE1SrFKh7cqbJ4+7nqSRL3
oBTud3rX41DGshOjpqcYHT4sqYlgZkc6dp0g1+hF1p3cGmjHdpysV2NVSUev
ghHbvSqhIsXFzRSWKiZOigmlkv3R5LnjpYyP4brM62Jl7y0qborvV4dNMz7m
D+5YxSlH0KAe8z6TT3LHuQdN7QCkFoiUSCaNhpAFaakkGIpqcqLhpOK4lXxt
kptCG93eUwNCcTxtx6bXufPR5TUHohvZvfeqMp42kL37FJC/A8ZHoOxXy8+X
X5QYxCQNuofWlvnIWv0Nr8w65x6lgVjPYmd/cHwzQKBTBMXN6pBud/PZL5zF
tw3QHlQkBR+UflMWZKeN9L0KdQ27mQlCo5gQS85aifxoiiA2v9+0hxZw91rP
IW4D+GS7oMMoKj8ZNyCJJsyf5smRZ+WxeBoolb3+TiGcBBCsRnfe6noLZiFO
6Zeu2ZwE
```

3.5. Creating a Signed-Only Message

There are two formats for signed messages defined for S/MIME:

- application/pkcs7-mime with SignedData.

- multipart/signed.

In general, the multipart/signed form is preferred for sending, and receiving agents MUST be able to handle both.

3.5.1. Choosing a Format for Signed-Only Messages

There are no hard-and-fast rules as to when a particular signed-only format is chosen. It depends on the capabilities of all the receivers and the relative importance of receivers with S/MIME facilities being able to verify the signature versus the importance of receivers without S/MIME software being able to view the message.

Messages signed using the multipart/signed format can always be viewed by the receiver whether or not they have S/MIME software. They can also be viewed whether they are using a MIME-native user

agent or they have messages translated by a gateway. In this context, "be viewed" means the ability to process the message essentially as if it were not a signed message, including any other MIME structure the message might have.

Messages signed using the SignedData format cannot be viewed by a recipient unless they have S/MIME facilities. However, the SignedData format protects the message content from being changed by benign intermediate agents. Such agents might do line wrapping or content-transfer encoding changes that would break the signature.

3.5.2. Signing Using application/pkcs7-mime with SignedData

This signing format uses the application/pkcs7-mime media type. The steps to create this format are:

Step 1. The MIME entity is prepared according to Section 3.1.

Step 2. The MIME entity and other required data are processed into a CMS object of type SignedData.

Step 3. The SignedData object is wrapped in a CMS ContentInfo object.

Step 4. The ContentInfo object is inserted into an application/pkcs7-mime MIME entity.

The smime-type parameter for messages using application/pkcs7-mime with SignedData is "signed-data". The file extension for this type of message is ".p7m".

A sample message would be:

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

MIIDmQYJKoZIhvcNAQcCoIIDijCCA4YCAQExCTAHBgUrDgMCGjAtBgkqhkiG9w0BBw
GgIAQeDQpUaGlzIGlzIHNvbWUgc2FtcGxlIGNvbnRlbnQuoIIC4DCCAtwwggKboAMC
AQICAgDIMAkGByqGSM44BAMwEjEQMA4GA1UEAxMHQ2FybERTUzAeFw05OTA4MTcwMT
EwNDlaFw0zOTEyMzEyMzU5NTlaMBMxETAPBgNVBAMTCEFsaWNlRFNTMIIBtjCCASsG
ByqGSM44BAEwggEeAoGBAIGNze2D6gqeOT7CSCij5EeT3Q7XqA7sU8WrhAhP/5Thc0
h+DNbzREjR/p+vpKGJL+HZMMg23j+bv7dM3F9piuR10DcMkQiVm96nXvn89J8v3UOo
i1TxP7AHCEdNXYjDw7Wz41UIddU5dhDEeL3/nbCElzfy5FEbteQJllzzflvbAhUA4k
emGkVmuBPG2o+4NyErYov3k80CgYAmONAUiTKqOfs+bdlLWWpMdiM5BAI1XPLLGjDD
HlBd3ZtZ4s2qBT1YwHuiNrhuB699ikIlp/R1z0oIXks+kPht6pzJIYo7dhTpzi5dow
fNI4W4LzABfG1JiRGJNkS9+MiVSlNWteL5c+waYTYfEX/Cve3RUP+YdMLRgUpgObo2
OQOBhAACgYBc47ladRSWC6l63eM/qeysXty9txMRNKYWiSgRI9k0hmd1dRMSPUNbb+
VRv/qJ8qIbPiR9PQeNW2PIu0WloErjhdbOBoA/6CN+GvIkq1MauCcNHu8Iv2YUgFxi
rGX6FYvxuzTU0pY39mFHssQyhPB+QUD9RqdjTjPypeL08oPluKOBgTB/MAwGA1UdEw
EB/wQCMAAwDgYDVR0PAQH/BAQDAgbAMB8GA1UdIwQYMBaAFHBEPoIub4feStN14z0g
vEMrk/EfMB0GA1UdDgQWBBS+bKGz48H37UNwpM4TAeL945f+zTAfBgNVHREEGDAWgR
RBbGljZURTU0BleGFtcGxlLmNvbTAJBgcqhkjOOAQDAzAAMC0CFFUMpBkfQiuJcSIz
jYNqtT1na79FAhUAn2FTUlQLXLLd2ud2HeIQUltDXr0xYzBhAgEBMBgwEjEQMA4GA1
UEAxMHQ2FybERTUwICAMgwBwYFKw4DAhowCQYHKoZIzjgEAwQuMCwCFD1cSW6LIUFz
eXle3YI5SKSBer/sAhQmCq7s/CTFHOEjgASeUjbMpx5g6A==
```

3.5.3. Signing Using the multipart/signed Format

This format is a clear-signing format. Recipients without any S/MIME or CMS processing facilities are able to view the message. It makes use of the multipart/signed media type described in [RFC1847]. The multipart/signed media type has two parts. The first part contains the MIME entity that is signed; the second part contains the "detached signature" CMS SignedData object in which the encapContentInfo eContent field is absent.

3.5.3.1. The application/pkcs7-signature Media Type

This media type always contains a CMS ContentInfo containing a single CMS object of type SignedData. The SignedData encapContentInfo eContent field MUST be absent. The signerInfos field contains the signatures for the MIME entity.

The file extension for signed-only messages using application/pkcs7-signature is ".p7s".

3.5.3.2. Creating a multipart/signed Message

Step 1. The MIME entity to be signed is prepared according to Section 3.1, taking special care for clear-signing.

Step 2. The MIME entity is presented to CMS processing in order to obtain an object of type SignedData in which the encapContentInfo eContent field is absent.

Step 3. The MIME entity is inserted into the first part of a multipart/signed message with no processing other than that described in Section 3.1.

Step 4. Transfer encoding is applied to the "detached signature" CMS SignedData object, and it is inserted into a MIME entity of type application/pkcs7-signature.

Step 5. The MIME entity of the application/pkcs7-signature is inserted into the second part of the multipart/signed entity.

The multipart/signed Content-Type has two required parameters: the protocol parameter and the micalg parameter.

The protocol parameter MUST be "application/pkcs7-signature". Note that quotation marks are required around the protocol parameter because MIME requires that the "/" character in the parameter value MUST be quoted.

The micalg parameter allows for one-pass processing when the signature is being verified. The value of the micalg parameter is dependent on the message digest algorithm(s) used in the calculation of the Message Integrity Check. If multiple message digest algorithms are used, they MUST be separated by commas per [RFC1847]. The values to be placed in the micalg parameter SHOULD be from the following:

Algorithm	Value Used
MD5*	md5
SHA-1*	sha-1
SHA-224	sha-224
SHA-256	sha-256
SHA-384	sha-384
SHA-512	sha-512
Any other	(defined separately in algorithm profile or "unknown" if not defined)

*Note: MD5 and SHA-1 are historical and no longer considered secure. See Appendix B for details.

(Historical note: some early implementations of S/MIME emitted and expected "rsa-md5", "rsa-sha1", and "sha1" for the micalg parameter.) Receiving agents SHOULD be able to recover gracefully from a micalg parameter value that they do not recognize. Future names for this parameter will be consistent with the IANA "Hash Function Textual Names" registry.

3.5.3.3. Sample multipart/signed Message

```
Content-Type: multipart/signed;
    micalg=sha-256;
    boundary="----=_NextBoundry____Fri,_06_Sep_2002_00:25:21";
    protocol="application/pkcs7-signature"

This is a multi-part message in MIME format.

------=_NextBoundry____Fri,_06_Sep_2002_00:25:21

This is some sample content.
------=_NextBoundry____Fri,_06_Sep_2002_00:25:21
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

MIIBJgYJKoZIhvcNAQcCoIIBFzCCARMCAQExADALBgkqhkiG9w0BBwExgf4w
gfsCAQIwJjASMRAwDgYDVQQDEwdDYXJsUlNBAhBGNGvHgABWvBHTbi7EELOw
MAsGCWCGSAFlAwQCAaAxMC8GCSqGSIb3DQEJBDEiBCCxwpZGNZzTSsugsn+f
lEidzQK4mf/ozKqfmbxhcIkKqjALBgkqhkiG9w0BAQsEgYB0XJV7fjPa5Nuh
oth5msDfP8A5urYUMjhNpWgXG8ae3XpppqVrPi2nVO41onHnkByjkeD/wc31
A9WH8MzFQgSTsrJ65JvffTTXkOpRPxsSHn3wJFwP/atWHkh8YK/jR9bULhUl
Mv5jQEDiwVX5DRasxu6Ld8zv9u5/TsdBNiufGw==

------=_NextBoundry____Fri,_06_Sep_2002_00:25:21--
```

The content that is digested (the first part of the multipart/signed) consists of the bytes:

```
54 68 69 73 20 69 73 20 73 6f 6d 65 20 73 61 6d 70 6c 65 20 63 6f 6e
74 65 6e 74 2e 0d 0a
```

3.6. Creating a Compressed-Only Message

This section describes the format for compressing a MIME entity. Please note that versions of S/MIME prior to version 3.1 did not specify any use of CompressedData, and will not recognize it. The

use of a capability to indicate the ability to receive CompressedData is described in [RFC3274] and is the preferred method for compatibility.

Step 1. The MIME entity to be compressed is prepared according to Section 3.1.

Step 2. The MIME entity and other required data are processed into a CMS object of type CompressedData.

Step 3. The CompressedData object is wrapped in a CMS ContentInfo object.

Step 4. The ContentInfo object is inserted into an application/pkcs7-mime MIME entity.

The smime-type parameter for compressed-only messages is "compressed-data". The file extension for this type of message is ".p7z".

A sample message would be:

```
Content-Type: application/pkcs7-mime; smime-type=compressed-data;
   name=smime.p7z
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7z

eNoLycgsVgCi4vzcVIXixNyCnFSF5Py8ktS8Ej0AlCkKVA==
```

3.7. Multiple Operations

The signed-only, enveloped-only, and compressed-only MIME formats can be nested. This works because these formats are all MIME entities that encapsulate other MIME entities.

An S/MIME implementation MUST be able to receive and process arbitrarily nested S/MIME within reasonable resource limits of the recipient computer.

It is possible to apply any of the signing, encrypting, and compressing operations in any order. It is up to the implementer and the user to choose. When signing first, the signatories are then securely obscured by the enveloping. When enveloping first the signatories are exposed, but it is possible to verify signatures without removing the enveloping. This can be useful in an environment where automatic signature verification is desired, as no private key material is required to verify a signature.

There are security ramifications to choosing whether to sign first or encrypt first. A recipient of a message that is encrypted and then signed can validate that the encrypted block was unaltered, but cannot determine any relationship between the signer and the unencrypted contents of the message. A recipient of a message that is signed then encrypted can assume that the signed message itself has not been altered, but that a careful attacker could have changed the unauthenticated portions of the encrypted message.

When using compression, keep the following guidelines in mind:

- Compression of binary encoded encrypted data is discouraged, since it will not yield significant compression. Base64 encrypted data could very well benefit, however.

- If a lossy compression algorithm is used with signing, you will need to compress first, then sign.

3.8. Creating a Certificate Management Message

The certificate management message or MIME entity is used to transport certificates and/or Certificate Revocation Lists, such as in response to a registration request.

Step 1. The certificates and/or Certificate Revocation Lists are made available to the CMS generating process that creates a CMS object of type SignedData. The SignedData encapContentInfo eContent field MUST be absent and signerInfos field MUST be empty.

Step 2. The SignedData object is wrapped in a CMS ContentInfo object.

Step 3. The ContentInfo object is enclosed in an application/pkcs7-mime MIME entity.

The smime-type parameter for a certificate management message is "certs-only". The file extension for this type of message is ".p7c".

3.9. Registration Requests

A sending agent that signs messages MUST have a certificate for the signature so that a receiving agent can verify the signature. There are many ways of getting certificates, such as through an exchange with a certification authority, through a hardware token or diskette, and so on.

S/MIME v2 [SMIMEv2] specified a method for "registering" public keys with certificate authorities using an application/pkcs10 body part. Since that time, the IETF PKIX Working Group has developed other methods for requesting certificates. However, S/MIME v4.0 does not require a particular certificate request mechanism.

3.10. Identifying an S/MIME Message

Because S/MIME takes into account interoperation in non-MIME environments, several different mechanisms are employed to carry the type information, and it becomes a bit difficult to identify S/MIME messages. The following table lists criteria for determining whether or not a message is an S/MIME message. A message is considered an S/MIME message if it matches any of the criteria listed below.

The file suffix in the table below comes from the "name" parameter in the Content-Type header field, or the "filename" parameter on the Content-Disposition header field. These parameters that give the file suffix are not listed below as part of the parameter section.

Media type	parameters	file suffix
application/pkcs7-mime	n/a	n/a
multipart/signed	protocol= "application/pkcs7-signature"	n/a
application/octet-stream	n/a	p7m, p7s, p7c, p7z

4. Certificate Processing

A receiving agent MUST provide some certificate retrieval mechanism in order to gain access to certificates for recipients of digital envelopes. This specification does not cover how S/MIME agents handle certificates, only what they do after a certificate has been validated or rejected. S/MIME certificate issues are covered in [RFC5750].

At a minimum, for initial S/MIME deployment, a user agent could automatically generate a message to an intended recipient requesting that recipient’s certificate in a signed return message. Receiving and sending agents SHOULD also provide a mechanism to allow a user to "store and protect" certificates for correspondents in such a way so as to guarantee their later retrieval.

4.1. Key Pair Generation

All generated key pairs MUST be generated from a good source of non-deterministic random input [RFC4086] and the private key MUST be protected in a secure fashion.

An S/MIME user agent MUST NOT generate asymmetric keys less than 2048 bits for use with an RSA signature algorithm.

For 2048-bit through 4096-bit RSA with SHA-256 see [RFC5754] and [FIPS186-4]. The first reference provides the signature algorithm's object identifier, and the second provides the signature algorithm's definition.

For RSASSA-PSS with SHA-256, see [RFC4056]. For RSAES-OAEP, see [RFC3560].

4.2. Signature Generation

The following are the requirements for an S/MIME agent generated RSA and RSASSA-PSS signatures:

```
        key size <= 2047 : SHOULD NOT (Note 1)
2048 <= key size <= 4096 : SHOULD     (see Security Considerations)
4096 <  key size         : MAY        (see Security Considerations)
```

Note 1: see Historical Mail Considerations in Section 6.
Note 2: see Security Considerations in Appendix B.

Key sizes for ECDSA and EdDSA are fixed by the curve.

4.3. Signature Verification

The following are the requirements for S/MIME receiving agents during signature verification of RSA and RSASSA-PSS signatures:

```
        key size <= 2047 : SHOULD NOT (Note 1)
2048 <= key size <= 4096 : MUST       (Note 2)
4096 <  key size         : MAY        (Note 2)
```

Note 1: see Historical Mail Considerations in Section 6.
Note 2: see Security Considerations in Appendix B.

Key sizes for ECDSA and EdDSA are fixed by the curve.

4.4. Encryption

The following are the requirements for an S/MIME agent when establishing keys for content encryption using the RSA, and RSA-OAEP algorithms:

```
        key size <= 2047 : SHOULD NOT (Note 1)
2048 <= key size <= 4096 : SHOULD     (Note 2)
4096 <  key size         : MAY        (Note 2)
```

Note 1: see Historical Mail Considerations in Section 6.
Note 2: see Security Considerations in Appendix B.

Key sizes for ECDH are fixed by the curve.

4.5. Decryption

The following are the requirements for an S/MIME agent when establishing keys for content decryption using the RSA and RSAES-OAEP algorithms:

```
        key size <= 2047 : MAY        (Note 1)
2048 <= key size <= 4096 : MUST       (Note 2)
4096 <  key size         : MAY        (Note 2)
```

Note 1: see Historical Mail Considerations in Section 6.
Note 2: see Security Considerations in Appendix B.

Key sizes for ECDH are fixed by the curve.

5. IANA Considerations

The following information updates the media type registration for application/pkcs7-mime and application/pkcs7-signature to refer to this document as opposed to RFC 2311.

Note that other documents can define additional MIME media types for S/MIME.

5.1. Media Type for application/pkcs7-mime

Type name: application

Subtype Name: pkcs7-mime

Required Parameters: NONE

Optional Parameters: smime-type/signed-data
smime-type/enveloped-data
smime-type/compressed-data
smime-type/certs-only
name

Encoding Considerations: See Section 3 of this document

Security Considerations: See Section 6 of this document

Interoperability Considerations: See Sections 1-6 of this document

Published Specification: RFC 2311, RFC 2633, and this document

Applications that use this media type: Security applications

Additional information: NONE

Person & email to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: NONE

Author: Sean Turner

Change Controller: S/MIME working group delegated from the IESG

5.2. Media Type for application/pkcs7-signature

Type name: application

Subtype Name: pkcs7-signature

Required Parameters: NONE

Optional Parameters: NONE

Encoding Considerations: See Section 3 of this document

Security Considerations: See Section 6 of this document

Interoperability Considerations: See Sections 1-6 of this document

Published Specification: RFC 2311, RFC 2633, and this document

Applications that use this media type: Security applications

Additional information: NONE

Person & email to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: NONE

Author: Sean Turner

Change Controller: S/MIME working group delegated from the IESG

5.3. Register authEnveloped-data smime-type

IANA is required to register the following value in the "Parameter Values for the smime-type Parameter" registry. The values to be registered are:

smime-type value: authEnveloped-data

Reference: [[This Document, Section 3.2.2]]

6. Security Considerations

Cryptographic algorithms will be broken or weakened over time. Implementers and users need to check that the cryptographic algorithms listed in this document continue to provide the expected level of security. The IETF from time to time may issue documents dealing with the current state of the art. For example:

- The Million Message Attack described in RFC 3218 [RFC3218].

- The Diffie-Hellman "small-subgroup" attacks described in RFC 2785 [RFC2785].

- The attacks against hash algorithms described in RFC 4270 [RFC4270].

This specification uses Public-Key Cryptography technologies. It is assumed that the private key is protected to ensure that it is not accessed or altered by unauthorized parties.

It is impossible for most people or software to estimate the value of a message's content. Further, it is impossible for most people or software to estimate the actual cost of recovering an encrypted message content that is encrypted with a key of a particular size. Further, it is quite difficult to determine the cost of a failed decryption if a recipient cannot process a message's content. Thus, choosing between different key sizes (or choosing whether to just use plaintext) is also impossible for most people or software. However, decisions based on these criteria are made all the time, and therefore this specification gives a framework for using those estimates in choosing algorithms.

The choice of 2048 bits as an RSA asymmetric key size in this specification is based on the desire to provide at least 100 bits of security. The key sizes that must be supported to conform to this specification seem appropriate for the Internet based on [RFC3766]. Of course, there are environments, such as financial and medical systems, that may select different key sizes. For this reason, an implementation MAY support key sizes beyond those recommended in this specification.

Receiving agents that validate signatures and sending agents that encrypt messages need to be cautious of cryptographic processing usage when validating signatures and encrypting messages using keys larger than those mandated in this specification. An attacker could send certificates with keys that would result in excessive cryptographic processing, for example, keys larger than those mandated in this specification, which could swamp the processing element. Agents that use such keys without first validating the certificate to a trust anchor are advised to have some sort of cryptographic resource management system to prevent such attacks.

Some cryptographic algorithms such as RC2 offer little actual security over sending plaintext. Other algorithms such as TripleDES, provide security but are no longer considered to be state of the art. S/MIME requires the use of current state of the art algorithms such

as AES and provides the ability to announce cryptographic capabilities to parties with whom you communicate. This allows the sender to create messages which can use the strongest common encryption algorithm. Using algorithms such as RC2 is never recommended unless the only alternative is no cryptography.

RSA and DSA keys of less than 2048 bits are now considered by many experts to be cryptographically insecure (due to advances in computing power), and should no longer be used to protect messages. Such keys were previously considered secure, so processing previously received signed and encrypted mail will often result in the use of weak keys. Implementations that wish to support previous versions of S/MIME or process old messages need to consider the security risks that result from smaller key sizes (e.g., spoofed messages) versus the costs of denial of service. If an implementation supports verification of digital signatures generated with RSA and DSA keys of less than 1024 bits, it MUST warn the user. Implementers should consider providing different warnings for newly received messages and previously stored messages. Server implementations (e.g., secure mail list servers) where user warnings are not appropriate SHOULD reject messages with weak signatures.

Implementers SHOULD be aware that multiple active key pairs can be associated with a single individual. For example, one key pair can be used to support confidentiality, while a different key pair can be used for digital signatures.

If a sending agent is sending the same message using different strengths of cryptography, an attacker watching the communications channel might be able to determine the contents of the strongly encrypted message by decrypting the weakly encrypted version. In other words, a sender SHOULD NOT send a copy of a message using weaker cryptography than they would use for the original of the message.

Modification of the ciphertext in EnvelopedData can go undetected if authentication is not also used, which is the case when sending EnvelopedData without wrapping it in SignedData or enclosing SignedData within it. This is one of the reasons for moving from EnvelopedData to AuthEnvelopedData, as the authenticated encryption algorithms provide the authentication without needing the SignedData layer.

If an implementation is concerned about compliance with National Institute of Standards and Technology (NIST) key size recommendations, then see [SP800-57].

If messaging environments make use of the fact that a message is signed to change the behavior of message processing (examples would be running rules or UI display hints), without first verifying that the message is actually signed and knowing the state of the signature, this can lead to incorrect handling of the message. Visual indicators on messages may need to have the signature validation code checked periodically if the indicator is supposed to give information on the current status of a message.

Many people assume that the use of an authenticated encryption algorithm is all that is needed for the sender of the message to be authenticated. In almost all cases this is not a correct statement. There are a number of preconditions that need to hold for an authenticated encryption algorithm to provide this service:

- The starting key must be bound to a single entity. The use of a group key only would allow for the statement that a message was sent by one of the entities that held the key but will not identify a specific entity.

- The message must have exactly one sender and one recipient. Having more than one recipient would allow for the second recipient to create a message that the first recipient would believe is from the sender by stripping the second recipient from the message.

- A direct path needs to exist from the starting key to the key used as the content encryption key (CEK). That path needs to guarantees that no third party could have seen the resulting CEK. This means that one needs to be using an algorithm that is called a "Direct Encryption" or a "Direct Key Agreement" algorithm in other contexts. This means that the starting key is used directly as the CEK key, or that the starting key is used to create a secret which then is transformed into the CEK via a KDF step.

S/MIME implementations almost universally use ephemeral-static rather than static-static key agreement and do not use a shared secret for encryption. This means that the first precondition is not met. There is a document [RFC6278] which defined how to use static-static key agreement with CMS, so the first precondition can be met. Currently, all S/MIME key agreement methods derive a KEK and wrap a CEK. This violates the third precondition above. New key agreement algorithms that directly created the CEK without creating an intervening KEK would need to be defined.

Even when all of the preconditions are met and origination of a message is established by the use of an authenticated encryption algorithm, users need to be aware that there is no way to prove this

to a third party. This is because either of the parties can successfully create the message (or just alter the content) based on the fact that the CEK is going to be known to both parties. Thus the origination is always built on a presumption that "I did not send this message to myself."

All of the authenticated encryption algorithms in this document use counter mode for the encryption portion of the algorithm. This means that the length of the plain text will always be known as the cipher text length and the plain text length are always the same. This information can enable passive observers to infer information based solely on the length of the message. Applications for which this is a concern need to provide some type of padding so that the length of the message does not provide this information.

When compression is used with encryption, it has the potential to add an additional layer of security. However, care needs to be taken when designing a protocol that relies on this not to create a compression oracle. Compression oracle attacks require an adaptive input to the process and attack the unknown content of a message based on the length of the compressed output. This means that no attack on the encryption key is necessarily required.

A recent paper on S/MIME and OpenPGP Email security [Efail] has pointed out a number of problems with the current S/MIME specifications and how people have implemented mail clients. Due to the nature of how CBC mode operates, the modes allow for malleability of plaintexts. This malleability allows for attackers to make changes in the cipher text and, if parts of the plain text are known, create arbitrary plaintexts blocks. These changes can be made without the weak integrity check in CBC mode being triggered. This type of attack can be prevented by the use of an AEAD algorithm with a more robust integrity check on the decryption process. It is therefore recommended that mail systems migrate to using AES-GCM as quickly as possible and that the decrypted content not be acted on prior to finishing the integrity check.

The other attack that is highlighted in [Efail] is due to an error in how mail clients deal with HTML and multipart/mixed messages. Clients MUST require that a text/html content type is a complete HTML document (per [RFC1866]). Clients SHOULD treat each of the different pieces of the multipart/mixed construct as being of different origins. Clients MUST treat each encrypted or signed piece of a MIME message as being of different origins both from unprotected content and from each other.

7. References

7.1. Normative References

[ASN.1] "Information Technology - Abstract Syntax Notation (ASN.1)".

ASN.1 syntax consists of the following references [X.680], [X.681], [X.682], and [X.683].

[CHARSETS] "Character sets assigned by IANA.", <http://www.iana.org/assignments/character-sets.>.

[CMS] "Cryptographic Message Syntax".

This is the set of documents dealing with the cryptographic message syntax and refers to [RFC5652] and [RFC5083].

[ESS] "Enhanced Security Services for S/MIME".

This is the set of documents dealing with enhanced security services and refers to [RFC2634] and [RFC5035].

[FIPS186-4] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-4, July 2013.

[I-D.ietf-curdle-cms-ecdh-new-curves] Housley, R., "Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)", draft-ietf-curdle-cms-ecdh-new-curves-10 (work in progress), August 2017.

[I-D.ietf-curdle-cms-eddsa-signatures] Housley, R., "Use of EdDSA Signatures in the Cryptographic Message Syntax (CMS)", draft-ietf-curdle-cms-eddsa-signatures-08 (work in progress), October 2017.

[I-D.ietf-lamps-rfc5750-bis] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/ MIME) Version 4.0 Certificate Handling", draft-ietf-lamps-rfc5750-bis-07 (work in progress), June 2018.

[MIME-SPEC]
"MIME Message Specifications".

This is the set of documents that define how to use MIME. This set of documents is [RFC2045], [RFC2046], [RFC2047], [RFC2049], [RFC6838], and [RFC4289].

[RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, DOI 10.17487/RFC1847, October 1995, <https://www.rfc-editor.org/info/rfc1847>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, DOI 10.17487/RFC2047, November 1996, <https://www.rfc-editor.org/info/rfc2047>.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, DOI 10.17487/RFC2049, November 1996, <https://www.rfc-editor.org/info/rfc2049>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2183] Troost, R., Dorner, S., and K. Moore, Ed., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, DOI 10.17487/RFC2183, August 1997, <https://www.rfc-editor.org/info/rfc2183>.

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, DOI 10.17487/RFC2634, June 1999, <https://www.rfc-editor.org/info/rfc2634>.

[RFC3274] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, DOI 10.17487/RFC3274, June 2002, <https://www.rfc-editor.org/info/rfc3274>.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, DOI 10.17487/RFC3370, August 2002, <https://www.rfc-editor.org/info/rfc3370>.

[RFC3560] Housley, R., "Use of the RSAES-OAEP Key Transport Algorithm in Cryptographic Message Syntax (CMS)", RFC 3560, DOI 10.17487/RFC3560, July 2003, <https://www.rfc-editor.org/info/rfc3560>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/info/rfc3565>.

[RFC4056] Schaad, J., "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", RFC 4056, DOI 10.17487/RFC4056, June 2005, <https://www.rfc-editor.org/info/rfc4056>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, DOI 10.17487/RFC4289, December 2005, <https://www.rfc-editor.org/info/rfc4289>.

[RFC5035] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, DOI 10.17487/RFC5035, August 2007, <https://www.rfc-editor.org/info/rfc5035>.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, DOI 10.17487/RFC5083, November 2007, <https://www.rfc-editor.org/info/rfc5083>.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC 5084, DOI 10.17487/RFC5084, November 2007, <https://www.rfc-editor.org/info/rfc5084>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, DOI 10.17487/RFC5753, January 2010, <https://www.rfc-editor.org/info/rfc5753>.

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, DOI 10.17487/RFC5754, January 2010, <https://www.rfc-editor.org/info/rfc5754>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SMIMEv4.0] "S/MIME version 4.0".

This group of documents represents S/MIME version 4.0. This set of documents are [RFC2634], [I-D.ietf-lamps-rfc5750-bis], [[This Document]], [RFC5652], and [RFC5035].

[X.680] "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T Recommendation X.680 (2002)", ITU-T X.680, ISO/IEC 8824-1:2008, November 2008.

[X.681] "Information Technology - Abstract Syntax Notation One (ASN.1): Information object specification", ITU-T X.681, ISO/IEC 8824-2:2008, November 2008.

[X.682] "Information Technology - Abstract Syntax Notation One (ASN.1): Constraint specification", ITU-T X.682, ISO/IEC 8824-3:2008, November 2008.

[X.683] "Information Technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications", ITU-T X.683, ISO/IEC 8824-4:2008, November 2008.

[X.690] "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).", ITU-T X.690, ISO/IEC 8825-1:2002, July 2002.

7.2. Informative References

[Efail] Poddebniak, D., Muller, J., Dresen, C., Ising, F., Schinzel, S., Friedberger, S., Somorovsky, J., and J. Schwenk, "Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels", Work in Progress , May 2018.

[FIPS186-2] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS) [With Change Notice 1]", Federal Information Processing Standards Publication 186-2, January 2000.

[RFC1866] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, DOI 10.17487/RFC1866, November 1995, <https://www.rfc-editor.org/info/rfc1866>.

[RFC2268] Rivest, R., "A Description of the RC2(r) Encryption Algorithm", RFC 2268, DOI 10.17487/RFC2268, March 1998, <https://www.rfc-editor.org/info/rfc2268>.

[RFC2311] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, DOI 10.17487/RFC2311, March 1998, <https://www.rfc-editor.org/info/rfc2311>.

[RFC2312] Dusse, S., Hoffman, P., Ramsdell, B., and J. Weinstein, "S/MIME Version 2 Certificate Handling", RFC 2312, DOI 10.17487/RFC2312, March 1998, <https://www.rfc-editor.org/info/rfc2312>.

[RFC2313] Kaliski, B., "PKCS #1: RSA Encryption Version 1.5", RFC 2313, DOI 10.17487/RFC2313, March 1998, <https://www.rfc-editor.org/info/rfc2313>.

[RFC2314] Kaliski, B., "PKCS #10: Certification Request Syntax Version 1.5", RFC 2314, DOI 10.17487/RFC2314, March 1998, <https://www.rfc-editor.org/info/rfc2314>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <https://www.rfc-editor.org/info/rfc2315>.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, DOI 10.17487/RFC2630, June 1999, <https://www.rfc-editor.org/info/rfc2630>.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, DOI 10.17487/RFC2631, June 1999, <https://www.rfc-editor.org/info/rfc2631>.

[RFC2632] Ramsdell, B., Ed., "S/MIME Version 3 Certificate Handling", RFC 2632, DOI 10.17487/RFC2632, June 1999, <https://www.rfc-editor.org/info/rfc2632>.

[RFC2633] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, DOI 10.17487/RFC2633, June 1999, <https://www.rfc-editor.org/info/rfc2633>.

[RFC2785] Zuccherato, R., "Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME", RFC 2785, DOI 10.17487/RFC2785, March 2000, <https://www.rfc-editor.org/info/rfc2785>.

[RFC3218] Rescorla, E., "Preventing the Million Message Attack on Cryptographic Message Syntax", RFC 3218, DOI 10.17487/RFC3218, January 2002, <https://www.rfc-editor.org/info/rfc3218>.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, DOI 10.17487/RFC3766, April 2004, <https://www.rfc-editor.org/info/rfc3766>.

[RFC3850] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, DOI 10.17487/RFC3850, July 2004, <https://www.rfc-editor.org/info/rfc3850>.

[RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <https://www.rfc-editor.org/info/rfc3851>.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, DOI 10.17487/RFC3852, July 2004, <https://www.rfc-editor.org/info/rfc3852>.

[RFC4134] Hoffman, P., Ed., "Examples of S/MIME Messages", RFC 4134, DOI 10.17487/RFC4134, July 2005, <https://www.rfc-editor.org/info/rfc4134>.

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, DOI 10.17487/RFC4270, November 2005, <https://www.rfc-editor.org/info/rfc4270>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5750] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, DOI 10.17487/RFC5750, January 2010, <https://www.rfc-editor.org/info/rfc5750>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/info/rfc5751>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <https://www.rfc-editor.org/info/rfc6194>.

[RFC6278] Herzog, J. and R. Khazan, "Use of Static-Static Elliptic Curve Diffie-Hellman Key Agreement in Cryptographic Message Syntax", RFC 6278, DOI 10.17487/RFC6278, June 2011, <https://www.rfc-editor.org/info/rfc6278>.

[RFC7114] Leiba, B., "Creation of a Registry for smime-type Parameter Values", RFC 7114, DOI 10.17487/RFC7114, January 2014, <https://www.rfc-editor.org/info/rfc7114>.

[RFC7905] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and S. Josefsson, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)", RFC 7905, DOI 10.17487/RFC7905, June 2016, <https://www.rfc-editor.org/info/rfc7905>.

[SMIMEv2] "S/MIME version v2".

This group of documents represents S/MIME version 2. This set of documents are [RFC2311], [RFC2312], [RFC2313], [RFC2314], and [RFC2315].

[SMIMEv3] "S/MIME version 3".

This group of documents represents S/MIME version 3. This set of documents are [RFC2630], [RFC2631], [RFC2632], [RFC2633], [RFC2634], and [RFC5035].

[SMIMEv3.1]
"S/MIME version 3.1".

This group of documents represents S/MIME version 3.1. This set of documents are [RFC2634], [RFC3850], [RFC3851], [RFC3852], and [RFC5035].

[SMIMEv3.2]
"S/MIME version 3.2".

This group of documents represents S/MIME version 3.2. This set of documents are [RFC2634], [RFC5750], [RFC5751], [RFC5652], and [RFC5035].

[SP800-56A]
National Institute of Standards and Technology (NIST), "Special Publication 800-56A Revision 2: Recommendation Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", May 2013.

[SP800-57]
National Institute of Standards and Technology (NIST), "Special Publication 800-57: Recommendation for Key Management", August 2005.

[TripleDES]
Tuchman, W., "Hellman Presents No Shortcut Solutions to DES"", IEEE Spectrum v. 16, n. 7, pp 40-41, July 1979.

Appendix A. ASN.1 Module

Note: The ASN.1 module contained herein is unchanged from RFC 3851 [SMIMEv3.1] with the exception of a change to the prefersBinaryInside ASN.1 comment. This module uses the 1988 version of ASN.1.

```
SecureMimeMessageV3dot1

  { iso(1) member-body(2) us(840) rsadsi(113549)
```

```
            pkcs(1) pkcs-9(9) smime(16) modules(0) msg-v3dot1(21) }

   DEFINITIONS IMPLICIT TAGS ::=

   BEGIN

   IMPORTS

   -- Cryptographic Message Syntax [CMS]
      SubjectKeyIdentifier, IssuerAndSerialNumber,
      RecipientKeyIdentifier
          FROM  CryptographicMessageSyntax
                { iso(1) member-body(2) us(840) rsadsi(113549)
                  pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2001(14) };

   --  id-aa is the arc with all new authenticated and unauthenticated
   --  attributes produced by the S/MIME Working Group

   id-aa OBJECT IDENTIFIER ::= {iso(1) member-body(2) usa(840)
           rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) attributes(2)}

   -- S/MIME Capabilities provides a method of broadcasting the
   -- symmetric capabilities understood.  Algorithms SHOULD be ordered
   -- by preference and grouped by type

   smimeCapabilities OBJECT IDENTIFIER ::= {iso(1) member-body(2)
           us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 15}

   SMIMECapability ::= SEQUENCE {
      capabilityID OBJECT IDENTIFIER,
      parameters ANY DEFINED BY capabilityID OPTIONAL }

   SMIMECapabilities ::= SEQUENCE OF SMIMECapability

   -- Encryption Key Preference provides a method of broadcasting the
   -- preferred encryption certificate.

   id-aa-encrypKeyPref OBJECT IDENTIFIER ::= {id-aa 11}

   SMIMEEncryptionKeyPreference ::= CHOICE {
      issuerAndSerialNumber   [0] IssuerAndSerialNumber,
      receipentKeyId          [1] RecipientKeyIdentifier,
      subjectAltKeyIdentifier [2] SubjectKeyIdentifier
   }

   -- receipentKeyId is spelt incorrectly, but kept for historical
   -- reasons.
```

```
   id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
           rsadsi(113549) pkcs(1) pkcs9(9) 16 }

   id-cap  OBJECT IDENTIFIER ::= { id-smime 11 }

   -- The preferBinaryInside OID indicates an ability to receive
   -- messages with binary encoding inside the CMS wrapper.
   -- The preferBinaryInside attribute's value field is ABSENT.

   id-cap-preferBinaryInside  OBJECT IDENTIFIER ::= { id-cap 1 }

   --  The following list OIDs to be used with S/MIME V3

   -- Signature Algorithms Not Found in [RFC3370], [RFC5754], [RFC4056],
   -- and [RFC3560]

   --
   -- md2WithRSAEncryption OBJECT IDENTIFIER ::=
   --    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)
   --     2}

   --
   -- Other Signed Attributes
   --
   -- signingTime OBJECT IDENTIFIER ::=
   --    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
   --     5}
   --    See [CMS] for a description of how to encode the attribute
   --    value.

   SMIMECapabilitiesParametersForRC2CBC ::= INTEGER
   --        (RC2 Key Length (number of bits))

   END
```

Appendix B. Historic Mail Considerations

Over the course of updating the S/MIME specifications, the set of recommended algorithms has been modified each time the document has been updated. This means that if a user has historic emails and their user agent has been updated to only support the current set of recommended algorithms some of those old emails will no longer be accessible. It is strongly suggested that user agents implement some of the following algorithms for dealing with historic emails.

This appendix contains a number of references to documents that have been obsoleted or replaced. This is intentional as frequently the updated documents do not have the same information in them.

B.1. DigestAlgorithmIdentifier

The following algorithms have been called our for some level of support by previous S/MIME specifications:

- SHA-1 was dropped in [SMIMEv4.0]. SHA-1 is no longer considered to be secure as it is no longer collision-resistant. The IETF statement on SHA-1 can be found in [RFC6194] but it is out-of-date relative to the most recent advances.

- MD5 was dropped in [SMIMEv4.0]. MD5 is no longer considered to be secure as it is no longer collision-resistant. Details can be found in [RFC6151].

B.2. Signature Algorithms

There are a number of problems with validating signatures on sufficiently historic messages. For this reason it is strongly suggested that UAs treat these signatures differently from those on current messages. These problems include:

- CAs are not required to keep certificates on a CRL beyond one update after a certificate has expired. This means that unless CRLs are cached as part of the message it is not always possible to check if a certificate has been revoked. The same problems exist with OCSP responses as they may be based on a CRL rather than on the certificate database.

- RSA and DSA keys of less than 2048 bits are now considered by many experts to be cryptographically insecure (due to advances in computing power). Such keys were previously considered secure, so processing of historic signed messages will often result in the use of weak keys. Implementations that wish to support previous versions of S/MIME or process old messages need to consider the security risks that result from smaller key sizes (e.g., spoofed messages) versus the costs of denial of service.

  [SMIMEv3.1] set the lower limit on suggested key sizes for creating and validation at 1024 bits. Prior to that the lower bound on key sizes was 512 bits.

- Hash functions used to validate signatures on historic messages may longer be considered to be secure. (See below.) While there are not currently any known practical pre-image or second pre-image attacks against MD5 or SHA-1, the fact they are no longer considered to be collision resistant implies that the security levels of the signatures are generally considered suspect. If a message is known to be historic, and it has been in the possession

of the client for some time, then it might still be considered to be secure.

- The previous two issues apply to the certificates used to validate the binding of the public key to the identity that signed the message as well.

The following algorithms have been called out for some level of support by previous S/MIME specifications:

- RSA with MD5 was dropped in [SMIMEv4.0]. MD5 is no longer considered to be secure as it is no longer collision-resistant. Details can be found in [RFC6151].

- RSA and DSA with SHA-1 were dropped in [SMIMEv4.0]. SHA-1 is no longer considered to be secure as it is no longer collision-resistant. The IETF statement on SHA-1 can be found in [RFC6194] but it is out-of-date relative to the most recent advances.

- DSA with SHA-256 was dropped in [SMIMEv4.0]. DSA has been replaced by elliptic curve versions.

As requirements for mandatory to implement has changed over time, some issues have been created that can cause interoperability problems:

- S/MIME v2 clients are only required to verify digital signatures using the rsaEncryption algorithm with SHA-1 or MD5, and might not implement id-dsa-with-sha1 or id-dsa at all.

- S/MIME v3 clients might only implement signing or signature verification using id-dsa-with-sha1, and might also use id-dsa as an AlgorithmIdentifier in this field.

- Note that S/MIME v3.1 clients support verifying id-dsa-with-sha1 and rsaEncryption and might not implement sha256withRSAEncryption.

NOTE: Receiving clients SHOULD recognize id-dsa as equivalent to id-dsa-with-sha1.

For 512-bit RSA with SHA-1 see [RFC3370] and [FIPS186-2] without Change Notice 1, for 512-bit RSA with SHA-256 see [RFC5754] and [FIPS186-2] without Change Notice 1, and for 1024-bit through 2048-bit RSA with SHA-256 see [RFC5754] and [FIPS186-2] with Change Notice 1. The first reference provides the signature algorithm’s object identifier, and the second provides the signature algorithm’s definition.

For 512-bit DSA with SHA-1 see [RFC3370] and [FIPS186-2] without Change Notice 1, for 512-bit DSA with SHA-256 see [RFC5754] and [FIPS186-2] without Change Notice 1, for 1024-bit DSA with SHA-1 see [RFC3370] and [FIPS186-2] with Change Notice 1, for 1024-bit and above DSA with SHA-256 see [RFC5754] and [FIPS186-4]. The first reference provides the signature algorithm's object identifier and the second provides the signature algorithm's definition.

B.3. ContentEncryptionAlgorithmIdentifier

The following algorithms have been called out for some level of support by previous S/MIME specifications:

- RC2/40 [RFC2268] was dropped in [SMIMEv3.2]. The algorithm is known to be insecure and, if supported, should only be used to decrypt existing email.

- DES EDE3 CBC [TripleDES], also known as "tripleDES" is dropped in [SMIMEv4.0]. This algorithms is removed from the supported list due to the fact that it has a 64-bit block size and the fact that it offers less that 128-bits of security. This algorithm should be supported only to decrypt existing email, it should not be used to encrypt new emails.

B.4. KeyEncryptionAlgorithmIdentifier

The following algorithms have been called out for some level of support by previous S/MIME specifications:

- DH ephemeral-static mode, as specified in [RFC3370] and [SP800-57], was dropped in [SMIMEv4.0].

- RSA key sizes have been increased over time. Decrypting old mail with smaller key sizes is reasonable, however new mail should use the updated key sizes.

For 1024-bit DH, see [RFC3370]. For 1024-bit and larger DH, see [SP800-56A]; regardless, use the KDF, which is from X9.42, specified in [RFC3370].

Appendix C. Moving S/MIME v2 Message Specification to Historic Status

The S/MIME v3 [SMIMEv3], v3.1 [SMIMEv3.1], and v3.2 [SMIMEv3.2] are backwards compatible with the S/MIME v2 Message Specification [SMIMEv2], with the exception of the algorithms (dropped RC2/40 requirement and added DSA and RSASSA-PSS requirements). Therefore, it is recommended that RFC 2311 [SMIMEv2] be moved to Historic status.

Appendix D. Acknowledgments

Many thanks go out to the other authors of the S/MIME version 2 Message Specification RFC: Steve Dusse, Paul Hoffman, Laurence Lundblade, and Lisa Repka. Without v2, there wouldn't be a v3, v3.1, v3.2 or v4.0.

Some of the examples in this document were copied from [RFC4134]. Thanks go the the people who wrote and verified the examples in that document.

A number of the members of the S/MIME Working Group have also worked very hard and contributed to this document. Any list of people is doomed to omission, and for that I apologize. In alphabetical order, the following people stand out in my mind because they made direct contributions to various versions of this document:

Tony Capel, Piers Chivers, Dave Crocker, Bill Flanigan, Peter Gutmann, Alfred Hoenes, Paul Hoffman, Russ Housley, William Ottaway, and John Pawling.

The version 4 update to the S/MIME documents was done under the auspices of the LAMPS Working Group.

Authors' Addresses

Jim Schaad
August Cellars

Email: ietf@augustcellars.com

Blake Ramsdell
Brute Squad Labs, Inc.

Email: blaker@gmail.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

LAMPS
Internet-Draft
Intended status: Standards Track
Expires: March 2, 2019

A. Truskovsky
D. Van Geest
ISARA Corporation
S. Fluhrer
P. Kampanakis
Cisco Systems
M. Ounsworth
S. Mister
Entrust Datacard, Ltd

August 29, 2018

Multiple Public-Key Algorithm X.509 Certificates
draft-truskovsky-lamps-pq-hybrid-x509-01

Abstract

This document describes a method of embedding alternative sets of cryptographic materials into X.509v3 digital certificates, X.509v2 Certificate Revocation Lists (CRLs), and PKCS #10 Certificate Signing Requests (CSRs). The embedded alternative cryptographic materials allow a Public Key Infrastructure (PKI) to use multiple cryptographic algorithms in a single object, and allow it to transition to the new cryptographic algorithms while maintaining backwards compatibility with systems using the existing algorithms. Three X.509 extensions and three PKCS #10 attributes are defined, and the signing and verification procedures for the alternative cryptographic material contained in the extensions and attributes are detailed.

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 https://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 March 2, 2019.

Table of Contents

1. Introduction

Modern Public Key Infrastructure (PKI) extensively relies on classical signature algorithms such as RSA or ECDSA to achieve secure authentication. The security of these algorithms is based on the time-tested difficulty of certain number-theoretic problems. However, it is well known that such schemes offer insufficient security against an adversary in possession of a universal quantum computer. Such an adversary can efficiently recover the private key from the public key and impersonate any entity in the system -- even a root Certification Authority (CA). Hence, it is necessary to upgrade these PKIs to utilize algorithms that are secure against such adversaries.

An obvious solution is for relying parties to require multiple certificates to establish trust in an entity. One could theoretically continue to use certificates as they currently are and introduce separate certificates that utilize the new algorithms. However, managing different cryptographic algorithms within a single PKI in this way requires multiple certificate chains. This would greatly increase the complexity of the already complex system. Furthermore, some systems rely on physical solutions for credential storage. These physical solutions may be limited in terms of capacity as well as in terms of how such systems are interacted with. Instead, it is far simpler to keep only a single identity and employ a single certificate chain for each user.

The goal of this document is to profile new X.509v3 certificate extensions, X.509v2 CRL extensions and PKCS #10 CSR attributes that facilitate the use of a simple and efficient approach for executing this upgrade. A key design requirement for this approach is to not affect the behavior of non-upgraded systems and ensure they can process any new attributes or extensions without breaking.

By placing an alternative public key and alternative signature into custom extensions, one effectively embeds multiple certificate chains within a single chain. By utilizing these multiple public-key algorithm certificates, legacy applications can continue using their current choices of cryptographic algorithms and upgraded applications can use new algorithms while remaining interoperable with the legacy systems.

It is useful to observe that even though the motivation for this document is to upgrade PKIs to use quantum-safe cryptography, the same methodology can be used to upgrade such systems to any new

algorithm. For this reason, this document does not specify that quantum-safe algorithms are the new technology the PKI is being upgraded to use.

The remainder of this document is organized as follows.

Section 2 profiles the three new PKCS #10 attributes and three new X.509 extensions. Sections 3, 4 and 5 profile methods for signing and verifying CSRs, certificates and CRLs respectively using the new extensions.

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. Terminology

The following terms are defined:

- alternative algorithm: The algorithm, whose usage is profiled in this document, which can be used to sign and verify a certificate instead of, or in addition to, the conventional algorithm.

- alternative [public, private] key: The keys, whose usage is profiled in this document, which can be used to create or verify a signature instead of, or in addition to, the conventional keys.

- alternative signature: The signature, whose usage is profiled in this document, which can be used to validate a certificate instead of, or in addition to, the conventional signature.

- conventional algorithm: The algorithm specified in the signatureAlgorithm field of an X.509v3 certificate.

- conventional [public, private] key: The key used to create or verify a conventional signature in an X.509v3 certificate.

- conventional signature: The value specified in the signature field of an X.509v3 certificate.

- multiple public-key algorithm certificate: A certificate which is equipped with the extensions introduced in this document. Thus, the certificate is signed and can be verified using two different public-key algorithms. One public-key algorithm (the "conventional" one) uses the keys, signatures and algorithms specified in the standard X.509v3 fields. The other

("alternative") public-key algorithm uses the keys, signatures and algorithms in the extensions defined in this document.

- upgraded [application, system]: An application or system which is capable of understanding and using the extensions introduced in this document.

2. Alternative Public-Key Algorithm Objects

2.1. OIDs

The following OIDs are used to identify the CSR attributes and X.509v3 extensions defined in the following sections.

```
id-subjectAltPublicKeyInfo  OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureAlgorithm    OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureValue        OBJECT IDENTIFIER  ::=  { TBD }
```

2.2. CSR Attributes

Three new CSR attributes are used to submit an alternative public key for certification. Each of these attributes mirror existing fields within a CSR and serve the same purpose as those fields, but with the alternative algorithms. An entity creating a CSR MUST include either all three of these attributes or none.

2.2.1. Subject Alt Public Key Info Attribute

The Subject Alt Public Key Info Attribute corresponds to the SubjectPublicKeyInfo type defined in Section 4.1 of [RFC2986]. This attribute carries information about the alternative public key being certified. The information also identifies the entity's alternative public-key algorithm (and any associated parameters).

This attribute is identified using the id-subjectAltPublicKeyInfo OID.

```
SubjectAltPublicKeyInfoAttr ATTRIBUTE  ::=  {
     WITH SYNTAX SubjectPublicKeyInfo
     ID id-subjectAltPublicKeyInfo }
```

2.2.2. Alt Signature Algorithm Attribute

The Alt Signature Algorithm attribute corresponds to the signatureAlgorithm field of the CertificationRequest type described in Section 4.2 of [RFC2986]. This attribute contains the identifier

for the alternative cryptographic algorithm used by the requesting entity to sign the CertificationRequestInfo.

This attribute is identified using the id-altSignatureAlgorithm OID.

```
AltSignatureAlgorithmAttr ATTRIBUTE  ::=  {
     WITH SYNTAX AlgorithmIdentifier
     ID id-altSignatureAlgorithm }
```

2.2.3. Alt Signature Value Attribute

The Alt Signature Value attribute corresponds to the signature field of the CertificationRequest type described in Section 4.2 of [RFC2986]. This attribute contains a digital signature computed upon the ASN.1 DER encoded PreCertificationRequestInfo as described in Section 3 of this document.

By generating this alternative signature, a certification request subject proves possession of the alternative private key.

This attribute is identified using the id-altSignatureValue OID.

```
AltSignatureValueAttr ATTRIBUTE  ::=  {
     WITH SYNTAX BIT STRING
     EQUALITY MATCHING RULE bitStringMatch
     ID id-altSignatureValue }
```

2.3. X.509v3 Extensions

Three new X.509v3 extensions are used to authenticate a certificate using alternative algorithms. Each of these extensions mirror existing fields within an X.509v3 certificate and serve the same purpose as those fields, but with the alternative algorithms.

2.3.1. Subject Alt Public Key Info Extension

The Subject Alt Public Key Info extension corresponds to the Subject Public Key Info field described in Section 4.1.2.7 of [RFC5280]. This extension carries the alternative public key, and identifies the algorithm with which the key is used.

This extension is identified using the id-subjectAltPublicKeyInfo OID.

```
SubjectAltPublicKeyInfoExt  ::=  SEQUENCE  {
     algorithm                AlgorithmIdentifier,
     subjectAltPublicKey      BIT STRING }
```

2.3.2. Alt Signature Algorithm Extension

The Alt Signature Algorithm extension corresponds to the signature field described in Section 4.1.2.3 of [RFC5280]. It also corresponds to the signatureAlgorithm field described in Section 4.1.1.2 of [RFC5280] since both those fields have the same values. This extension contains the identifier for the alternative digital signature algorithm used by the CA to sign the preTBSCertificate.

This extension is identified using the id-altSignatureAlgorithm OID.

```
AltSignatureAlgorithmExt  ::=  AlgorithmIdentifier
```

2.3.3. Alt Signature Value Extension

The Alt Signature Value extension corresponds to the signatureValue field described in Section 4.1.1.3 of [RFC5280]. This extension contains a digital signature computed upon the ASN.1 DER encoded preTBSCertificate as described in Section 4.

By generating this alternative signature, a CA certifies the validity of the preTBSCertificate data. In particular, the CA certifies the binding between the alternative public key material and the subject of the certificate.

This extension is identified using the id-altSignatureValue OID.

```
AltSignatureValueExt  ::=  BIT STRING
```

3. Multiple Public-Key Algorithm Certificate Signing Requests

A Certificate Signing Request (CSR) is a sequence of three required fields as defined in Section 4.2 of [RFC2986].

```
CertificationRequest  ::=  SEQUENCE  {
     certificationRequestInfo    CertificationRequestInfo,
     signatureAlgorithm          AlgorithmIdentifier,
     signature                   BIT STRING }
```

A CSR’s signature is calculated on the ASN.1 DER encoding of the CertificationRequestInfo object as defined in Section 4.2 of [RFC2986].

```
CertificationRequestInfo  ::=  SEQUENCE  {
     version       INTEGER  { v1(0) } (v1,...),
     subject       Name,
     subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
     attributes    [0] Attributes{{ CRIAttributes }} }
```

The alternative signature is calculated on the ASN.1 DER encoding of the identical PreCertificationRequestInfo object.

```
PreCertificationRequestInfo  ::=  SEQUENCE  {
     version       INTEGER  { v1(0) } (v1,...),
     subject       Name,
     subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
     attributes    [0] Attributes{{ CRIAttributes }} }
```

The PreCertificationRequestInfo type is the same as the CertificationRequestInfo type, however the PreCertificationRequestInfo object will have different attributes than the CertificationRequestInfo. Specifically, the CertificationRequestInfo will include the AltSignatureValueAttr attribute, while the PreCertificationRequestInfo will not.

3.1. Creating Multiple Public-Key Algorithm CSRs

A multiple public-key algorithm CSR requires the applicant to generate two key pairs: one for the old algorithm (the conventional key pair), and another for the new algorithm (the alternative key pair). All actions taken by the applicant with regards to the conventional algorithm and key pair are unchanged during this process. Additional attributes are populated to prove that the applicant is in possession of the alternative private key.

The PreCertificationRequestInfo object MUST contain the SubjectAltPublicKeyInfoAttr attribute carrying the alternative public key and algorithm for the CSR being created.

The PreCertificationRequestInfo object MUST contain the AltSignatureAlgorithmAttr attribute, which specifies the algorithm identifier for the algorithm used to sign the PreCertificationRequestInfo object.

The alternative signature of the PreCertificationRequestInfo MUST be calculated using the alternative private key of the certificate request subject, which is the private key associated with the public key found in the subject’s SubjectAltPublicKeyInfoAttr attribute.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueAttr attribute to create the CertificationRequestInfo object.

The process of signing a multiple public-key algorithm CSR as described above can be summarized as follows:

a. Create a PreCertificationRequestInfo object, which is populated with all the data to be signed by the alternative private key, including the SubjectAltPublicKeyInfoAttr and AltSignatureAlgorithmAttr attributes.

b. Calculate the alternative signature on the DER encoding of the PreCertificationRequestInfo, using the certificate request subject's alternative private key with the algorithm specified in the AltSignatureAlgorithmAttr attribute.

c. Convert the PreCertificationRequestInfo to a CertificationRequestInfo by adding the calculated alternative signature to the PreCertificationRequestInfo object using the AltSignatureValueAttr attribute.

d. As per [RFC2986], calculate the conventional signature using the certificate request subject's conventional private key and create the CertificationRequest from the certificationRequestInfo, signatureAlgorithm and signature.

An upgraded system MAY issue both multiple public-key algorithm and single public-key algorithm CSRs depending on their policies. If the system issues a single public-key algorithm CSR, then that CSR MUST NOT contain any of the three attributes profiled in this section.

3.2. Verifying Multiple Public-Key Algorithm CSRs

The certificate issuer verifies the alternative signature of the multiple public-key algorithm CSR by reconstructing the PreCertificationRequestInfo object and using its ASN.1 DER encoding, alternative public key and alternative signature algorithm to verify the signature.

To verify the alternative signature of a multiple public-key algorithm CSR, the following steps are taken:

a. ASN.1 DER decode the certificationRequestInfo field of the CertificationRequest to get a CertificationRequestInfo object.

b. Remove the AltSignatureValueAttr attribute from the CertificationRequestInfo object and set aside the alternative signature. The object is now the same as the PreCertificationRequestInfo which the signature was generated on.

c. ASN.1 DER encode the PreCertificationRequestInfo object.

d. Using the algorithm specified in the AltSignatureAlgorithmAttr attribute of the PreCertificationRequestInfo, the alternative

public key from the CSR's SubjectAltPublicKeyInfoAttr attribute and the ASN.1 DER encoded PreCertificationRequestInfo, verify the alternative signature from (b).

During the process of ASN.1 DER decoding the CertificationRequestInfo, removing the AltSignatureValueAttr attribute from the PreCertificationRequestInfo, and ASN.1 DER encoding the PreCertificationRequestInfo, the relative ordering of the remaining attributes is not modified. This is due to the DER encoding rules applied during signature generation as specified in RFC2986. Thus, the resulting ASN.1 DER encoded PreCertificationRequestInfo is identical to the one the issuer used to generate the alternative signature.

4. Multiple Public-Key Algorithm Certificates

An X.509 digital certificate is a sequence of three fields as defined in [RFC5280].

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING }
```

An X.509v3 certificate's signature is calculated on the ASN.1 DER encoding of the TBSCertificate object as defined in Section 4.1 of [RFC5280]. In this way, a CA certifies the validity of the information in the tbsCertificate field, in particular the binding between the conventional public key material and the subject of the certificate.

The alternative signature is calculated on the ASN.1 DER encoding of the similar, but not identical, PreTBSCertificate defined below. This signature also certifies the validity of the information in the tbsCertificate field. In particular, the binding between the alternative public key material and the subject of the certificate is validated.

```
PreTBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }
```

The PreTBSCertificate type is similar to the TBSCertificate type, except that the PreTBSCertificate does not include the signature field (the third element in the TBSCertificate sequence). In a TBSCertificate the signature field contains the AlgorithmIdentifier of the algorithm which will be used to sign the final certificate, and this value might not be known at the time that the alternative signature is calculated. Additionally, since the AlgorithmIdentifier of the signature field is associated with the final signatureValue field in the certificate, it is outside the scope of the alternative public-key algorithm and does not need to be protected by the alternative signature.

The PreTBSCertificate object also does not contain the AltSignatureValueExt extension in its extension list, while the TBSCertificate will. Since the alternative signature is calculated on the encoding of the PreTBSCertificate it cannot be included in the PreTBSCertificate.

4.1. Creating Multiple Public-Key Algorithm Certificates

If a CA is issuing a subject certificate and the issuer certificate or root of trust contains an alternative public key, then the CA SHOULD add an alternative signature to the subject certificate. Failure to do so could result in a verifier rejecting the certificate as being malformed, especially if the verifier is concerned about quantum-enabled adversaries. This is discussed further in Section 8.1.

A multiple public-key algorithm certificate MAY contain the SubjectAltPublicKeyInfoExt extension. If the certificate's subject has an alternative public key which they wish to bind to their identity, then the public key and algorithm MUST be placed in the SubjectAltPublicKeyInfoExt extension. However, if the certificate's

subject has no such alternative public key (e.g. the subject's application has not been upgraded to support multiple public-key algorithms) then the SubjectAltPublicKeyInfoExt extension will not be added to the certificate.

If a CA is issuing a certificate with an alternative signature, the extensions field of the PreTBSCertificate MUST contain the AltSignatureAlgorithmExt extension, which specifies the algorithm identifier for the algorithm used to sign the PreTBSCertificate.

The alternative signature of the PreTBSCertificate MUST be calculated using the alternative private key of the Issuer, which is the private key associated with the public key found in the Issuer's SubjectAltPublicKeyInfoExt extension.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueExt extension to the extensions list of the PreTBSCertificate, resulting in the TBSCertificate.

The process of signing an X.509v3 multiple public-key algorithm certificate as described above can be summarized as follows:

a. Create a PreTBSCertificate object, which is populated with all the data to be signed by the alternative private key, including the SubjectAltPublicKeyInfoExt and AltSignatureAlgorithmExt extensions.

b. Calculate the alternative signature on the DER encoding of the PreTBSCertificate, using the Issuer's alternative private key with the algorithm specified in the AltSignatureAlgorithmExt extension.

c. Add the calculated alternative signature to the PreTBSCertificate object using the AltSignatureValueExt extension.

d. Convert the PreTBSCertificate to a TBSCertificate by adding the signature field and populating it with the algorithm identifier of the conventional algorithm to be used to sign the certificate.

e. As per [RFC5280], calculate the conventional signature using the conventional private key associated with the Issuer's certificate and create the certificate from the tbsCertificate, signatureAlgorithm and signature.

If the upgraded CA's policy allows it to process single public-key algorithm CSRs and issue single public-key algorithm certificates, and the issuer's certificate has an alternative public key, and the

CA receives a single-algorithm CSR, the CA SHOULD still include properly calculated AltSignatureValueExt and AltSignatureAlgorithmExt extensions in the certificate. This ensures that when an upgraded system verifies the subject's certificate and sees that the issuer certificate contains the SubjectAltPublicKeyInfoExt extension that it will verify the subject's alternative signature. Otherwise it might treat the subject's certificate as invalid. This is discussed further in the Security Considerations section.

Note - A certificate issuer would typically mark the SubjectAltPublicKeyInfoExt, AltSignatureAlgorithmExt and AltSignatureValueExt extensions as non-critical, allowing the multiple public-key algorithm certificate to be treated like a regular certificate by non-upgraded entities. However, the issuer MAY mark the extensions as critical, for example if it is part of a PKI which requires entities to understand both the conventional and alternative signatures.

4.2. Verifying Multiple Public-Key Algorithm Certificates

Users wishing to verify a multiple public-key algorithm certificate using the alternative public-key algorithm will need to convert the tbsCertificate field in the certificate to a PreTBSCertificate object identical to the PreTBSCertificate object which the issuer used to create the alternative signature. Then the user can use the issuer's alternative public key with the alternative signature algorithm to verify the alternative signature of the PreTBSCertificate.

To verify the alternative signature of the multiple public-key algorithm certificate, the following steps are taken:

a. ASN.1 DER decode the tbsCertificate field of the certificate to get a TBSCertificate object.

b. Remove the AltSignatureValueExt extension from the TBSCertificate object and set aside the alternative signature.

c. Remove the signature field from the TBSCertificate object, converting it to a PreTBSCertificate object.

d. ASN.1 DER encode the PreTBSCertificate object.

e. Using the algorithm specified in the AltSignatureAlgorithmExt extension of the PreTBSCertificate, the alternative public key from the Issuer's SubjectAltPublicKeyInfoExt extension and the ASN.1 DER encoded PreTBSCertificate, verify the alternative signature from (b).

The issuer's alternative public key comes from the issuing certificate's SubjectAltPublicKeyInfoExt extension, unless the issuer is a trust anchor. In that case, the trust anchor's alternative public key may come from a self-signed certificate's SubjectAltPublicKeyInfoExt extension, or it may come from elsewhere. [RFC5280] section 6.1.1 (d) lists the trust anchor information as including:

a. the trusted issuer name,

b. the trusted public key algorithm,

c. the trusted public key, and

d. optionally, the trusted public key parameters associated with the public key.

When validating a multiple public-key algorithm certificate, the trust anchor information also includes:

a. the trusted alternative public key algorithm,

b. the trusted alternative public key, and

c. optionally, the trusted alternative public key parameters associated with the alternative public key.

During the process of ASN.1 DER decoding the TBSCertificate, removing the AltSignatureValueExt extension from the PreTBSCertificate and ASN.1 DER encoding the PreTBSCertificate, the relative ordering of the remaining extensions is not modified. Thus, the resulting ASN.1 DER encoded PreTBSCertificate is identical to the one the issuer used to generate the alternative signature.

A certificate that contains an AltSignatureValueExt extension but does not contain an AltSignatureAlgorithmExt extension cannot be verified under the alternative public-key algorithm and so SHOULD be rejected as being malformed. Similarly, a certificate that contains an AltSignatureAlgorithmExt extension but does not contain an AltSignatureValueExt extension SHOULD be rejected.

A certificate MAY have AltSignatureValueExt and AltSignatureAlgorithmExt extensions without having a SubjectAltPublicKeyInfoExt extension. This case could arise if a non-upgraded subject requests a certificate from an upgraded CA who has a multiple public-key algorithm CA certificate.

If an issuer certificate or root of trust has an alternative public key, but a subject certificate issued by the issuer certificate or root of trust doesn't contain an alternative signature then the verifier SHOULD reject the subject certificate. This is especially important if the verifier is concerned about quantum-enabled adversaries. This is discussed further in the Section 8.1. Accepting such a subject certificate SHOULD be limited to cases where the verifier has been explicitly configured to ignore missing alternative signatures for a given issuing CA, for subject certificates matching a given wildcard, or similar whitelisting mechanisms.

5. Multiple Public-Key Algorithm Certificate Revocation Lists

In certain situations, certificates must be revoked and no longer used. This can happen for a variety of reasons including, but not limited to: key compromise, CA compromise, or due to a change in affiliation. Roughly speaking, Certificate Revocation Lists (CRLs) are authenticated lists of revoked certificates.

An X.509v2 Certificate Revocation List (CRL) is a sequence of three fields as defined in [RFC5280].

```
CertificateList  ::=  SEQUENCE  {
    tbsCertList          TBSCertList,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue       BIT STRING  }
```

An X.509v2 CRL's signature is calculated on the ASN.1 DER encoding of the TBSCertList object as defined in Section 5.1 of [RFC5280].

The alternative signature is calculated on the ASN.1 DER encoding of the similar, but not identical, PreTBSCertList object defined here.

```
PreTBSCertList  ::=  SEQUENCE  {
     version                 Version OPTIONAL,
                                  -- if present, MUST be v2
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time OPTIONAL,
     revokedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          revocationDate          Time,
          crlEntryExtensions      Extensions OPTIONAL
                                   -- if present, version MUST be v2
                                  }  OPTIONAL,
     crlExtensions           [0]  EXPLICIT Extensions OPTIONAL
                                   -- if present, version MUST be v2
                                  }
```

The PreTBSCertList object is similar to the TBSCertList object, except that the PreTBSCertList does not include the signature field (the second element in the TBSCertList sequence). In a TBSCertList the signature field contains the AlgorithmIdentifier of the algorithm which will sign the final certificate revocation list, and this value might not be known at the time that the alternative signature is calculated. Additionally, since the AlgorithmIdentifier of the signature field is associated with the final signatureValue field in the CRL, it is outside the scope of the alternative public-key algorithm and does not need to be protected by the alternative signature.

The PreTBSCertList object also does not contain the AltSignatureValueExt extension in its extension list, while the TBSCertList will. Since the alternative signature is calculated on the encoding of the PreTBSCertList, it cannot be included in the TBSCertList.

If a multiple public-key algorithm certificate is revoked, whether because the classical key is compromised, the alternative key is compromised or or other reason, both the classical and alternative keys SHOULD be considered revoked. This avoids any unneeded complexity in dealing with a certificate where one key is compromised but the other isn’t.

5.1. Creating Multiple Public-Key Algorithm Certificate Revocation Lists

To create a multiple public-key algorithm CRL, one creates a CRL as specified in Section 5 of [RFC5280] and includes the additional extensions as specified in this section.

If the CRL issuer's certificate has a SubjectAltPublicKeyInfoExt extension, the CRL SHOULD be created with an alternative signature. Otherwise, some upgraded systems may fail to validate the CRL because it is not trusted under the alternative public-key algorithm.

The extensions field of the PreTBSCertList MUST contain the AltSignatureAlgorithmExt extension, which specifies the algorithm identifier for the algorithm used to sign the PreTBSCertList.

The alternative signature of the PreTBSCertList MUST be calculated using the alternative private key of the CRL issuer, which is the private key associated with the public key found in the CRL issuer X.509v3 certificate's SubjectAltPublicKeyInfoExt extension.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueExt extension to the extensions list of the PreTBSCertList, resulting in the TBSCertList.

The process of signing an X.509v2 multiple public-key algorithm CRL as described above can be summarized as follows:

a. Create a TBSCertList object, which is populated with all the data to be signed by the alternative private key, including the AltSignatureAlgorithmExt extension.

b. Calculate the alternative signature on the DER encoding of the PreTBSCertList, using the CRL issuer's alternative private key with the algorithm specified in the AltSignatureAlgorithmExt extension.

c. Add the calculated alternative signature to the PreTBSCertList object using the AltSignatureValueExt extension.

d. Convert the PreTBSCertList to a TBSCertList by adding the signature field and populating it with the algorithm identifier of the conventional algorithm to be used to sign the certificate.

e. As per [RFC5280], calculate the conventional signature using the conventional private key associated with the CRL issuer's certificate and create the CRL from the tbsCertList, signatureAlgorithm and signature.

Note - A CRL issuer would typically mark the AltSignatureAlgorithmExt and AltSignatureValueExt extensions as non-critical, allowing the multiple public-key algorithm CRL to be treated like a regular CRL by non-upgraded entities. However, the issuer may be part of a PKI which requires entities to understand both the conventional and

alternative signatures, in which case it would mark the extensions as critical.

5.2. Verifying Multiple Public-Key Algorithm Certificate Revocation Lists

Users wishing to verify the alternative signature of a multiple public-key algorithm CRL will need to convert the tbsCertList field in the CRL to a PreTBSCertList identical to the PreTBSCertList which the issuer used to create the alternative signature. Then the user can use the CRL issuer certificate's alternative public key with the alternative signature algorithm to verify the alternative signature of the PreTBSCertList.

To verify the alternative signature of the multiple public-key algorithm CRL, the following steps are taken:

a. ASN.1 DER decode the tbsCertList field of the certificate to get a TBSCertList object.

b. Remove the AltSignatureValueExt extension from the TBSCertList object and set aside the alternative signature.

c. Remove the signature field from the TBSCertList object, converting it to a PreTBSCertList object.

d. ASN.1 DER encode the PreTBSCertList object.

e. Using the algorithm specified in the AltSignatureAlgorithmExt extension of the PreTBSCertList, the alternative public key from the CRL issuer certificate's SubjectAltPublicKeyInfoExt extension and the ASN.1 DER encoded PreTBSCertList, verify the alternative signature from (b).

During the process of ASN.1 DER decoding the TBSCertList, removing the AltSignatureValueExt extension from the PreTBSCertList and ASN.1 DER encoding the PreTBSCertList, the relative ordering of the remaining extensions will not be modified. Thus, the resulting ASN.1 DER encoded PreTBSCertList is identical to the one the issuer used to generate the alternative signature.

In addition to verifying the alternative signature of a CRL, an implementation also needs to validate the CRL issuer's certificate and the certificate chain it is a part of. Implementations SHOULD use the same method as profiled in Section 6 of [RFC5280] with the following modifications to the CRL processing algorithm of that document's Section 6.3.3. Step (f) of the CRL processing algorithm requires certificate path validation for the issuer of the complete

CRL. To validate multiple public-key algorithm CRLs, upgraded entities SHOULD additionally verify the alternative signatures along the path as described in Section 4.2 of this document. Step (g) of the CRL Processing algorithm requires the verification of a single signature on the complete CRL. To verify multiple public-key algorithm CRLs, this step MUST be modified to instead verify dual signatures on the complete CRL. Similarly, in step (h) of the same algorithm, if use-deltas is set and if the delta CRL is a multiple public-key algorithm CRL, then the verifying peer should validate the signature on the delta CRL via the method described above, and use standard practice otherwise - using the public key(s) validated in step (f).

6. Acknowledgements

The authors would like to thank Philip Lafrance and John Gray for their valuable contributions.

7. IANA Considerations

Extensions in certificates and CRLs are identified using object Identifiers (OIDs). The creation and delegation of these arcs is to be determined.

8. Security Considerations

Many of the security considerations for this document closely follow those of [RFC5280]. However, the use of the extensions introduced in this document does bring rise to additional considerations.

The motivation behind this document is to provide a method of upgrading PKIs and dependent systems to achieve quantum-safe state. However, state-of-the-art quantum-safe signature schemes tend to have large signature or key sizes. As such, their inclusion on CSRs, certificates, or CRLs means that the sizes of these data structures will significantly increase. This could potentially cause problems in protocols or implementations expecting more reasonable sizes. Even if enterprises choose instead to upgrade their PKI to new, but still classically secure signature algorithms, these algorithms can also be expected to have large signature or key sizes; often a by-product of an increased level of security is larger signatures or key sizes.

There is a great deal of flexibility inherent to the use of the extensions introduced in this document. Their design is such that a clean separation is made between the old and new signatures. The new signatures have no dependency on the old signatures and no understanding of the new signatures is required to compute or verify

the old signature. As such, one could rely on the conventional signature only, the alternative signature only, or both, depending on the policies of the entity.

It is paramount that all private keying material be kept secret; a subject covered in the Security Considerations section of [RFC5280]. If the PKI is upgraded to use quantum-safe technologies, then it is of key importance to ensure that all private materials are protected against quantum-enabled adversaries as well. How such a feat is accomplished is outside the scope of this document. Additionally, issues such as re-keying or key management are outside the scope of this document.

Typically, the SubjectAltPublicKeyInfoExt, AltSignatureAlgorithmExt and AltSignatureValueExt extensions will be marked as non-critical so that a non-upgraded system could treat a multiple public-key algorithm certificate or CSR as a conventional certificate. However, a PKI could choose to enforce the usage of both conventional and alternative public-key algorithms, in which case it MAY mark these extensions as critical. The reasons why a PKI may want to do this are outside the scope of this document.

8.1. Post-Quantum Security Considerations

While this document is intended to facilitate transitioning a PKI from a classical public-key algorithm to a quantum-safe public-key algorithm, with the transition completing before the development of quantum computers capable of breaking classical public-key algorithms, it is worth discussing security considerations if multiple public-key algorithm certificates are used in the presence of a quantum-enabled adversary.

A quantum-enabled adversary is expected to be able to forge signatures for certificates and CRLs using classically secure signature algorithms. Thus, a CA SHOULD add an alternative signature to any certificate it issues if the issuing certificate contains a SubjectAltPublicKeyInfoExt extension. If the trust anchor is not a certificate, the alternative signature SHOULD be added if the trust anchor has an associated alternative public key which could be used for verification. Similarly, when verifying certificates or CRLs an application SHOULD reject certificates or CRLs if they don’t contain an alternative signature but the issuer certificate does contain a SubjectAltPublicKeyInfoExt or the trust anchor has an alternative public key. If the CA does not add the alternative signature in these cases, and an upgraded application does not take this precaution when verifying, then a quantum-enabled adversary could create a certificate or CRL without an alternative signature, and

forge the conventional signature of any issuer, causing upgraded applications to accept forged credentials.

If an upgraded relying party processing a non-multiple public-key algorithm CRL encounters a multiple public-key algorithm certificate (containing an AltSignatureValueExt extension) in the list of revoked certificates, it SHOULD NOT treat that certificate as revoked. If the conventional signature of the CRL uses a non-quantum-safe signature algorithm (e.g. RSA or ECDSA), a quantum-enabled attacker may have forged the CRL, thereby revoking certificates that the CA didn't intend to revoke. If one of those certificates has the multiple public-key algorithm extension then it was intended to be processed using the alternative public-key algorithm and should not be revoked based on only the results of the conventional public-key algorithm.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

Appendix A. ASN.1 Structures and OIDs

This appendix includes all of the ASN.1 type and value definitions introduced in this document.

```
   DEFINITIONS IMPLICIT TAGS ::= BEGIN

   -- EXPORTS All --
   -- IMPORTS NONE --

   -- Object Identifiers for the certificate extensions introduced in
   -- Section 4.

   id-subjectAltPublicKeyInfo OBJECT IDENTIFIER  ::=  { TBD }

   id-altSignatureAlgorithm OBJECT IDENTIFIER  ::=  { TBD }

   id-altSignatureValue OBJECT IDENTIFIER  ::=  { TBD }

   -- X.509 Certificate extensions

   SubjectAltPublicKeyInfoExt  ::=  SEQUENCE  {
        algorithm                  AlgorithmIdentifier,
        subjectAltPublicKey        BIT STRING }

   AltSignatureAlgorithmExt  ::=  AlgorithmIdentifier

   AltSignatureValueExt  ::=  BIT STRING

   -- attribute data types

   subjectAltPublicKeyInfoAttr ATTRIBUTE  ::=  {
        WITH SYNTAX SubjectPublicKeyInfo
        ID id-subjectAltPublicKeyInfo }

   altSignatureAlgorithmAttr ATTRIBUTE  ::=  {
        WITH SYNTAX AlgorithmIdentifier
        ID id-altSignatureAlgorithm }

   altSignatureValueAttr ATTRIBUTE  ::=  {
        WITH SYNTAX BIT STRING
        EQUALITY MATCHING RULE bitStringMatch
        ID id-altSignatureValue }

   END
```

Appendix B. Upgrading PKI and Dependent Systems

One way to upgrade these systems is to employ a "top down" approach: First the root CA is upgraded, then the same is done for any subordinate CAs, and finally for end entities. The dependent applications can then be upgraded in phases, where the upgraded applications can switch to using the new public-key algorithms while

non-upgraded systems can continue using the old public-key algorithms.

Appendix C. Options for Alternative Algorithms

Out of all branches of mathematics thought to be suitable for quantum-safe cryptographic algorithm development, the theory of hash functions, specifically hash-based signatures are currently the most trusted in regard to their quantum security assurances. While the private key state management makes using them challenging in some high-frequency use cases, they are very well suited for roots of trust and code signing; hash-based algorithms can already be used to upgrade CA certificates. Furthermore, the option will be available to use stateless digital signatures in end-entity certificates when they become available.

Authors' Addresses

Alexander Truskovsky
ISARA Corporation
560 Westmount Rd N
Waterloo, Ontario N2L 0A9
Canada

Email: alexander.truskovsky@isara.com

Daniel Van Geest
ISARA Corporation
560 Westmount Rd N
Waterloo, Ontario N2L 0A9
Canada

Email: daniel.vangeest@isara.com

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sfluhrer@cisco.com

Panos Kampanakis
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: pkampana@cisco.com

Mike Ounsworth
Entrust Datacard, Ltd
1000 Innovation Drive
Kanata, Ontario K2K 3E7
Canada

Email: mike.ounsworth@entrustdatacard.com

Serge Mister
Entrust Datacard, Ltd
1000 Innovation Drive
Kanata, Ontario K2K 3E7
Canada

Email: serge.mister@entrustdatacard.com