Stream: Independent Submission
RFC: 8755
Category: Informational
Published: March 2020
ISSN: 2070-1721
Author: M. Jenkins
NSA

RFC 8755
Using Commercial National Security Algorithm Suite Algorithms in Secure/Multipurpose Internet Mail Extensions

Abstract

The United States Government has published the National Security Agency (NSA) Commercial National Security Algorithm (CNSA) Suite, which defines cryptographic algorithm policy for national security applications. This document specifies the conventions for using the United States National Security Agency's CNSA Suite algorithms in Secure/Multipurpose Internet Mail Extensions (S/MIME) as specified in RFC 8551. It applies to the capabilities, configuration, and operation of all components of US National Security Systems that employ S/MIME messaging. US National Security Systems are described in NIST Special Publication 800-59. It is also appropriate for all other US Government systems that process high-value information. It is made publicly available for use by developers and operators of these and any other system deployments.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8755.

Copyright Notice

Table of Contents

1. Introduction

This document specifies the conventions for using the United States National Security Agency's Commercial National Security Algorithm (CNSA) Suite algorithms [CNSA] in Secure/Multipurpose Internet Mail Extensions (S/MIME) [RFC8551]. It applies to the capabilities, configuration, and operation of all components of US National Security Systems that employ S/MIME messaging. US

National Security Systems are described in NIST Special Publication 800-59 [SP80059]. It is also appropriate for all other US Government systems that process high-value information. It is made publicly available for use by developers and operators of these and any other system deployments.

S/MIME makes use of the Cryptographic Message Syntax (CMS) [RFC5652] [RFC5083]. In particular, the signed-data, enveloped-data, and authenticated-enveloped-data content types are used. This document only addresses CNSA Suite compliance for S/MIME. Other applications of CMS are outside the scope of this document.

This document does not define any new cryptographic algorithm suites; instead, it defines a CNSA-compliant profile of S/MIME. Since many of the CNSA Suite algorithms enjoy uses in other environments as well, the majority of the conventions needed for these algorithms are already specified in other documents. This document references the source of these conventions, with some relevant details repeated to aid developers that choose to support the CNSA Suite. Where details have been repeated, the cited documents are authoritative.

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. The Commercial National Security Algorithm Suite

The National Security Agency (NSA) profiles commercial cryptographic algorithms and protocols as part of its mission to support secure, interoperable communications for US Government National Security Systems. To this end, it publishes guidance both to assist with the US Government transition to new algorithms and to provide vendors -- and the Internet community in general -- with information concerning their proper use and configuration.

Recently, cryptographic transition plans have become overshadowed by the prospect of the development of a cryptographically relevant quantum computer. The NSA has established the Commercial National Security Algorithm (CNSA) Suite to provide vendors and IT users near-term flexibility in meeting their cybersecurity interoperability requirements. The purpose behind this flexibility is to avoid having vendors and customers make two major transitions in a relatively short timeframe, as we anticipate a need to shift to quantum-resistant cryptography in the near future.

The NSA is authoring a set of RFCs, including this one, to provide updated guidance concerning the use of certain commonly available commercial algorithms in IETF protocols. These RFCs can be used in conjunction with other RFCs and cryptographic guidance (e.g., NIST Special Publications) to properly protect Internet traffic and data-at-rest for US Government National Security Systems.

3. Requirements and Assumptions

CMS values are generated using ASN.1 [X208], the Basic Encoding Rules (BER) [X209], and the Distinguished Encoding Rules (DER) [X509].

The elliptic curve used in the CNSA Suite is specified in [FIPS186] and appears in the literature under two different names. For the sake of clarity, we list both names below:

Curve	NIST Name	SECG Name	OID [FIPS186]
nistp384	P-384	secp384r1	1.3.132.0.34

Table 1

For CNSA Suite applications, public key certificates used to verify S/MIME signatures **MUST** be compliant with the CNSA Suite Certificate and Certificate Revocation List (CRL) profile specified in [RFC8603].

Within the CMS signed-data content type, signature algorithm identifiers are located in the signatureAlgorithm field of SignerInfo structures contained within the SignedData. In addition, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of countersignature attributes. Specific requirements for digital signatures are given in Section 5; compliant implementations **MUST** consider signatures not meeting these requirements as invalid.

Implementations based on Elliptic Curve Cryptography (ECC) also require specification of schemes for key derivation and key wrap. Requirements for these schemes are in Sections 6.1.1 and 6.1.2, respectively.

RSA key pairs (public, private) are identified by the modulus size expressed in bits; RSA-3072 and RSA-4096 are computed using moduli of 3072 bits and 4096 bits, respectively.

RSA signature key pairs used in CNSA Suite-compliant implementations are either RSA-3072 or RSA-4096. The RSA exponent e **MUST** satisfy $2^{16} < e < 2^{256}$ and be odd per [FIPS186].

It is recognized that, while the vast majority of RSA signatures are currently made using the RSASSA-PKCS1-v1_5 algorithm, the preferred RSA signature scheme for new applications is RSASSA-PSS. CNSA Suite-compliant X.509 certificates will be issued in accordance with [RFC8603], and while those certificates must be signed and validated using RSASSA-PKCS1-v1_5, the subject's RSA key pair can be used to generate and validate signatures appropriate for either signing scheme. Where use of RSASSA-PSS is indicated in this document, the parameters in Section 5.2.2 apply.

This document assumes that the required trust anchors have been securely provisioned to the client.

All implementations use SHA-384 for hashing and either AES-CBC or AES-GCM for encryption, the requirements for which are given in Section 4 and Section 7, respectively.

4. SHA-384 Message Digest Algorithm

SHA-384 is the sole CNSA Suite message digest algorithm. [RFC5754] specifies the conventions for using SHA-384 with the Cryptographic Message Syntax (CMS). CNSA Suite-compliant S/MIME implementations **MUST** follow the conventions in [RFC5754].

Within the CMS signed-data content type, message digest algorithm identifiers are located in the SignedData digestAlgorithms field and the SignerInfo digestAlgorithm field.

The SHA-384 message digest algorithm is defined in FIPS Pub 180 [FIPS180]. The algorithm identifier for SHA-384 is defined in [RFC5754] as follows:

```
id-sha384  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistalgorithm(4) hashalgs(2) 2 }
```

For SHA-384, the AlgorithmIdentifier parameters field is **OPTIONAL**, and if present, the parameters field **MUST** contain a NULL. As specified in [RFC5754], implementations **MUST** generate SHA-384 AlgorithmIdentifiers with absent parameters. Implementations **MUST** accept SHA-384 AlgorithmIdentifiers with absent parameters or with NULL parameters.

5. Digital Signature

5.1. ECDSA Signature

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the CNSA Suite digital signature algorithm based on ECC. [RFC5753] specifies the conventions for using ECDSA with the Cryptographic Message Syntax (CMS). CNSA Suite-compliant S/MIME implementations **MUST** follow the conventions in [RFC5753].

[RFC5480] defines the signature algorithm identifier used in CMS for ECDSA with SHA-384 as follows:

```
ecdsa-with-SHA384  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) ansi-X9-62(10045) signatures(4)
   ecdsa-with-sha2(3) 3 }
```

When the ecdsa-with-SHA384 algorithm identifier is used, the AlgorithmIdentifier parameters field **MUST** be absent.

When signing, the ECDSA algorithm generates two values, commonly called r and s. These two values **MUST** be encoded using the ECDSA-Sig-Value type specified in [RFC5480]:

```
ECDSA-Sig-Value  ::=  SEQUENCE {
   r  INTEGER,
   s  INTEGER }
```

5.2. RSA Signature

The RSA signature generation process and the encoding of the result is either RSASSA-PKCS1-v1_5 or RSA-PSS, as described in detail in PKCS #1 version 2.2 [RFC8017].

5.2.1. RSA-PKCS1-v1_5

[RFC5754] defines the signature algorithm identifier used in CMS for an RSA signature with SHA-384 as follows:

```
sha384WithRSAEncryption  OBJECT IDENTIFIER  ::= { iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 12 }
```

When the sha384WithRSAEncryption algorithm identifier is used, the parameters **MUST** be NULL. Implementations **MUST** accept the parameters being absent as well as present.

5.2.2. RSA-PSS

[RFC4056] defines the signature algorithm identifier used in CMS for an RSA-PSS signature as follows (presented here in expanded form):

```
RSASSA-PSS  OBJECT IDENTIFIER  ::= { iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }
```

The parameters field of an AlgorithmIdentifier that identifies RSASSA-PSS is defined in [RFC4055] as follows:

```
 RSASSA-PSS-params  ::=  SEQUENCE  {
    hashAlgorithm      [0] HashAlgorithm DEFAULT
                           sha1Identifier,
    maskGenAlgorithm   [1] MaskGenAlgorithm DEFAULT
                           mgf1SHA1Identifier,
    saltLength         [2] INTEGER DEFAULT 20,
    trailerField       [3] INTEGER DEFAULT 1  }
```

The AlgorithmIdentifier parameters field **MUST** contain RSASSA-PSS-params with the following values:

- The hash algorithm **MUST** be id-sha384 as defined in [RFC8017];
- The mask generation function **MUST** use the algorithm identifier mfg1SHA384Identifier as defined in [RFC4055];
- The salt length **MUST** be 48 octets (the same length as the SHA-384 output); and

- The trailerField **MUST** have value 1.

6. Key Establishment

6.1. Elliptic Curve Key Agreement

Elliptic Curve Diffie-Hellman (ECDH) is the CNSA Suite key agreement algorithm. Since S/MIME is used in store-and-forward communications, ephemeral-static ECDH is always employed. This means that the message originator possesses an ephemeral ECDH key pair and that the message recipient possesses a static ECDH key pair whose public key is provided in an X.509 certificate. The certificate used to obtain the recipient's public key **MUST** be compliant with [RFC8603].

When a key agreement algorithm is used, the following steps are performed:

1. A content-encryption key (CEK) for a particular content-encryption algorithm is generated at random.
2. The recipient's public key and sender's private key are used with a key agreement scheme to generate a shared secret (Z).
3. The shared secret is used with a key derivation function (KDF) to produce a key-encryption key (KEK).
4. The KEK is used with a key wrap algorithm to encrypt the CEK.

Key derivation is discussed in Section 6.1.1. Key wrapping is discussed in Section 6.1.2.

Section 3.1 of [RFC5753] specifies the conventions for using ECDH with the CMS. CNSA Suite-compliant S/MIME implementations **MUST** follow these conventions.

Within the CMS enveloped-data and authenticated-enveloped-data content types, key agreement algorithm identifiers are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm field.

The keyEncryptionAlgorithm field comprises two fields, an algorithm field and a parameter field. The algorithm field **MUST** identify dhSinglePass-stdDH-sha384kdf-scheme. The algorithm identifier for the dhSinglePass-stdDH-sha384kdf-scheme, repeated from Section 7.1.4 of [RFC5753], is (presented here in expanded form):

```
dhSinglePass-stdDH-sha384kdf-scheme  OBJECT IDENTIFIER  ::=
    { iso(1) identified-organization(3) certicom(132)
      schemes(1) 11 2 }
```

The keyEncryptionAlgorithm parameter field **MUST** be constructed as described in Section 6.1.2.

6.1.1. Key Derivation Functions

KDFs based on SHA-384 are used to derive a pairwise key-encryption key from the shared secret produced by ephemeral-static ECDH. Sections 7.1.8 and 7.2 in [RFC5753] specify the CMS conventions for using a KDF with the shared secret generated during ephemeral-static ECDH. CNSA Suite-compliant S/MIME implementations **MUST** follow these conventions.

As specified in Section 7.1.8 of [RFC5753], the ANSI-X9.63-KDF described in Section 3.6.1 of [SEC1] and based on SHA-384 **MUST** be used.

As specified in Section 7.2 of [RFC5753], when using ECDH with the CMS enveloped-data or authenticated-enveloped-data content type, the derivation of key-encryption keys makes use of the ECC-CMS-SharedInfo type:

```
ECC-CMS-SharedInfo  ::=  SEQUENCE {
   keyInfo      AlgorithmIdentifier,
   entityUInfo  [0] EXPLICIT OCTET STRING OPTIONAL,
   suppPubInfo  [2] EXPLICIT OCTET STRING }
```

In the CNSA Suite for S/MIME, the fields of ECC-CMS-SharedInfo are used as follows:

- keyInfo contains the object identifier of the key-encryption algorithm used to wrap the content-encryption key. If AES-256 Key Wrap is used, then the keyInfo will contain id-aes256-wrap-pad, and the parameters will be absent.
- entityUInfo optionally contains a random value provided by the message originator. If user keying material (ukm) is included in the KeyAgreeRecipientInfo, then the entityUInfo **MUST** be present, and it **MUST** contain the ukm value. If the ukm is not present, then the entityUInfo **MUST** be absent.
- suppPubInfo contains the length of the generated key-encryption key in bits, represented as a 32-bit unsigned number, as described in [RFC2631]. When a 256-bit AES key is used, the length **MUST** be 0x00000100.

ECC-CMS-SharedInfo is DER encoded and is used as input to the key derivation function, as specified in Section 3.6.1 of [SEC1]. Note that ECC-CMS-SharedInfo differs from the OtherInfo specified in [RFC2631]. Here, a counter value is not included in the keyInfo field because the KDF specified in [SEC1] ensures that sufficient keying data is provided.

The KDF specified in Section 3.6.1 of [SEC1] describes how to generate an essentially arbitrary amount of keying material from a shared secret, Z, produced by ephemeral-static ECDH. To generate an L-bit key-encryption key (KEK), blocks of key material (KM) are computed by incrementing Counter appropriately until enough material has been generated:

```
KM(Counter) = Hash ( Z || Counter || ECC-CMS-SharedInfo )
```

The KM blocks are concatenated left to right as they are generated, and the first (leftmost) L bits are used as the KEK:

```
KEK = the leftmost L bits of
         [KM ( counter=1 ) || KM ( counter=2 ) ...]
```

In the CNSA Suite for S/MIME, the elements of the KDF are defined as follows:

- Hash is a one-way hash function. The SHA-384 hash **MUST** be used.
- Z is the shared secret value generated during ephemeral-static ECDH. Z **MUST** be exactly 384 bits, i.e., leading zero bits **MUST** be preserved.
- Counter is a 32-bit unsigned number represented in network byte order. Its initial value **MUST** be 0x00000001 for any key derivation operation.
- ECC-CMS-SharedInfo is composed as described above. It **MUST** be DER encoded.

In the CNSA Suite for S/MIME, exactly one iteration is needed; the Counter is not incremented. The key-encryption key (KEK) **MUST** be the first (leftmost) 256 bits of the SHA-384 output value:

```
KEK = the leftmost 256 bits of
         SHA-384 ( Z || 0x00000001 || ECC-CMS-SharedInfo )
```

Note that the only source of secret entropy in this computation is Z.

6.1.2. AES Key Wrap

The AES Key Wrap with Padding key-encryption algorithm, as specified in [RFC5649] and [SP80038F], is used to encrypt the content-encryption key with a pairwise key-encryption key that is generated using ephemeral-static ECDH. Section 8 of [RFC5753] specifies the CMS conventions for using AES Key Wrap with a pairwise key generated through ephemeral-static ECDH. CNSA Suite-compliant S/MIME implementations **MUST** follow these conventions.

Within the CMS enveloped-data content type, key wrap algorithm identifiers are located in the KeyWrapAlgorithm parameters within the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm field.

The KeyWrapAlgorithm **MUST** be id-aes256-wrap-pad. The required algorithm identifier, specified in [RFC5649], is:

```
id-aes256-wrap-pad  OBJECT IDENTIFIER ::=  { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 48 }
```

6.2. RSA Key Transport

RSA encryption (RSA) is the CNSA Suite key transport algorithm. The RSA key transport algorithm is the RSA encryption scheme defined in [RFC8017], where the message to be encrypted is the content-encryption key.

The recipient of an S/MIME message possesses an RSA key pair whose public key is represented by an X.509 certificate. The certificate used to obtain the recipient's public key **MUST** be compliant with [RFC8603]. These certificates are suitable for use with either RSAES-OAEP or RSAES-PKCS1-v1_5.

6.2.1. RSAES-PKCS1-v1_5

Section 4.2 of [RFC3370] specifies the conventions for using RSAES-PKCS1-v1_5 with the CMS. S/MIME implementations employing this form of key transport **MUST** follow these conventions.

Within the CMS enveloped-data and authenticated-enveloped-data content types, key transport algorithm identifiers are located in the EnvelopedData RecipientInfos KeyTransRecipientInfo keyEncryptionAlgorithm field.

The algorithm identifier for RSA (PKCS #1 v1.5) is:

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

The AlgorithmIdentifier parameters field **MUST** be present, and the parameters field **MUST** contain NULL.

6.2.2. RSAES-OAEP

[RFC3560] specifies the conventions for using RSAES-OAEP with the CMS. CNSA Suite-compliant S/MIME implementations employing this form of key transport **MUST** follow these conventions.

Within the CMS enveloped-data and authenticated-enveloped-data content types, key transport algorithm identifiers are located in the EnvelopedData RecipientInfos KeyTransRecipientInfo keyEncryptionAlgorithm field.

The algorithm identifier for RSA (OAEP) is:

```
id-RSAES-OAEP  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 7 }
```

The parameters field of an AlgorithmIdentifier that identifies RSAES-OAEP is defined in [RFC4055] as follows:

```
RSAES-OAEP-params  ::=  SEQUENCE  {
   hashFunc          [0] AlgorithmIdentifier DEFAULT
                            sha1Identifier,
   maskGenFunc       [1] AlgorithmIdentifier DEFAULT
                            mgf1SHA1Identifier,
   pSourceFunc       [2] AlgorithmIdentifier DEFAULT
                            pSpecifiedEmptyIdentifier  }

pSpecifiedEmptyIdentifier  AlgorithmIdentifier  ::=
                     { id-pSpecified, nullOctetString }

nullOctetString  OCTET STRING (SIZE (0))  ::=  { ''H }
```

The AlgorithmIdentifier parameters field **MUST** be present, and the parameters field **MUST** contain RSAES-OAEP-params with values as follows:

- The hashFunc algorithm must be id-sha384 as defined in [RFC8017];
- The mask generation function must use the algorithm identifier mfg1SHA384Identifier as defined in [RFC4055];
- The pSourceFunc field must be absent.

The SMIMECapabilities signed attribute is used to specify a partial list of algorithms that the software announcing the SMIMECapabilities can support. If the SMIMECapabilities signed attribute is included to announce support for the RSAES-OAEP algorithm, it **MUST** be constructed as defined in Section 5 of [RFC3560], with the sequence representing the rSAES-OAEP-SHA384-Identifier.

7. Content Encryption

AES-GCM is the preferred mode for CNSA Suite applications, as described in the Security Considerations (Section 8). AES-CBC is acceptable where AES-GCM is not yet available.

7.1. AES-GCM Content Encryption

CNSA Suite-compliant S/MIME implementations using the authenticated-enveloped-data content type [RFC5083] **MUST** use AES [FIPS197] in Galois Counter Mode (GCM) [SP80038D] as the content-authenticated encryption algorithm and **MUST** follow the conventions for using AES-GCM with the CMS defined in [RFC5084].

Within the CMS authenticated-enveloped-data content type, content-authenticated encryption algorithm identifiers are located in the AuthEnvelopedData EncryptedContentInfo contentEncryptionAlgorithm field. The content-authenticated encryption algorithm is used to encipher the content located in the AuthEnvelopedData EncryptedContentInfo encryptedContent field.

The AES-GCM content-authenticated encryption algorithm is described in [FIPS197] and [SP80038D]. The algorithm identifier for AES-256 in GCM mode is:

```
id-aes256-GCM  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 46 }
```

The AlgorithmIdentifier parameters field **MUST** be present, and the parameters field must contain GCMParameters:

```
GCMParameters ::= SEQUENCE {
  aes-nonce        OCTET STRING,
  aes-ICVlen       AES-GCM-ICVlen DEFAULT 12 }
```

The authentication tag length (aes-ICVlen) **SHALL** be 16 (indicating a tag length of 128 bits).

The initialization vector (aes-nonce) **MUST** be generated in accordance with Section 8.2 of [SP80038D]. AES-GCM loses security catastrophically if a nonce is reused with a given key on more than one distinct set of input data. Therefore, a fresh content-authenticated encryption key **MUST** be generated for each message.

7.2. AES-CBC Content Encryption

CNSA Suite-compliant S/MIME implementations using the enveloped-data content type **MUST** use AES-256 [FIPS197] in Cipher Block Chaining (CBC) mode [SP80038A] as the content-encryption algorithm and **MUST** follow the conventions for using AES with the CMS defined in [RFC3565].

Within the CMS enveloped-data content type, content-encryption algorithm identifiers are located in the EnvelopedData EncryptedContentInfo contentEncryptionAlgorithm field. The content-encryption algorithm is used to encipher the content located in the EnvelopedData EncryptedContentInfo encryptedContent field.

The AES-CBC content-encryption algorithm is described in [FIPS197] and [SP80038A]. The algorithm identifier for AES-256 in CBC mode is:

```
id-aes256-CBC  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 42 }
```

The AlgorithmIdentifier parameters field **MUST** be present, and the parameters field must contain AES-IV:

```
AES-IV  ::=  OCTET STRING (SIZE(16))
```

The 16-octet initialization vector is generated at random by the originator. See [RFC4086] for guidance on generation of random values.

8. Security Considerations

This document specifies the conventions for using the NSA's CNSA Suite algorithms in S/MIME. All of the algorithms and algorithm identifiers have been specified in previous documents.

See [RFC4086] for guidance on generation of random values.

The security considerations in [RFC5652] discuss the CMS as a method for digitally signing data and encrypting data.

The security considerations in [RFC3370] discuss cryptographic algorithm implementation concerns in the context of the CMS.

The security considerations in [RFC5753] discuss the use of elliptic curve cryptography (ECC) in the CMS.

The security considerations in [RFC3565] discuss the use of AES in the CMS.

The security considerations in [RFC8551] apply to this profile, particularly the recommendation to use authenticated encryption modes (i.e., use authenticated-enveloped-data with AES-GCM rather than enveloped-data with AES-CBC).

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[CNSA] Committee for National Security Systems, "Use of Public Standards for Secure Information Sharing", CNSS Policy 15, October 2016, <https://www.cnss.gov/CNSS/Issuances/Policies.cfm>.

[FIPS180] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standard 180-4, August 2015, <https://csrc.nist.gov/publications/detail/fips/180/4/final>.

[FIPS186] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", DOI 10.6028/NIST.FIPS.186-4, FIPS PUB 186-4, July 2013, <https://csrc.nist.gov/publications/detail/fips/186/4/final>.

[FIPS197] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", DOI 10.6028/NIST.FIPS.197, FIPS PUB 197, November 2001, <https://csrc.nist.gov/publications/detail/fips/197/final>.

[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>.

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

[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>.

[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>.

[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>.

[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>.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <https://www.rfc-editor.org/info/rfc5649>.

[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>.

[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>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8603] Jenkins, M. and L. Zieglar, "Commercial National Security Algorithm (CNSA) Suite Certificate and Certificate Revocation List (CRL) Profile", RFC 8603, DOI 10.17487/RFC8603, May 2019, <https://www.rfc-editor.org/info/rfc8603>.

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

[SP80038A] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", DOI 10.6028/NIST.SP.800-38A, Special Publication 800-38A, December 2001, <https://csrc.nist.gov/publications/detail/sp/800-38a/final>.

[SP80038D] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", DOI 10.6028/NIST.SP.800-38D, Special Publication 800-38D, November 2007, <https://csrc.nist.gov/publications/detail/sp/800-38d/final>.

[SP80038F] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping", DOI 10.6028/NIST.SP.800-38F, Special Publication 800-38F, December 2012, <https://csrc.nist.gov/publications/detail/sp/800-38f/final>.

[X208] CCITT, "Specification of Abstract Syntax Notation One (ASN.1)", CCITT Recommendation X.208, 1988, <https://www.itu.int/rec/T-REC-X.208-198811-W/en>.

[X209] CCITT, "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", CCITT Recommendation X.209, 1988, <https://www.itu.int/rec/T-REC-X.209-198811-W/en>.

[X509] CCITT, "The Directory - Authentication Framework", CCITT Recommendation X.509, 1988, <https://www.itu.int/rec/T-REC-X.509-198811-S>.

10.2. Informative References

[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>.

[SP80059] Barker, W., "Guideline for Identifying an Information System as a National Security System", DOI 10.6028/NIST.SP.800-59, Special Publication 800-59, August 2003, <https://csrc.nist.gov/publications/detail/sp/800-59/final>.

Author's Address

Michael Jenkins
National Security Agency
Email: mjjenki@nsa.gov