Stream: Internet Engineering Task Force (IETF)
RFC: 9481
Updates: 4210
Category: Standards Track
Published: November 2023
ISSN: 2070-1721
Authors: H. Brockhaus *Siemens*, H. Aschauer *Siemens*, M. Ounsworth *Entrust*, J. Gray *Entrust*

RFC 9481
Certificate Management Protocol (CMP) Algorithms

Abstract

This document describes the conventions for using several cryptographic algorithms with the Certificate Management Protocol (CMP). CMP is used to enroll and further manage the lifecycle of X.509 certificates. This document also updates the algorithm use profile from Appendix D.2 of RFC 4210.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
   1.1. Terminology 3
2. Message Digest Algorithms 4
   2.1. SHA2 4
   2.2. SHAKE 4
3. Signature Algorithms 5
   3.1. RSA 5
   3.2. ECDSA 6
   3.3. EdDSA 7
4. Key Management Algorithms 8
   4.1. Key Agreement Algorithms 8
      4.1.1. Diffie-Hellman 9
      4.1.2. ECDH 9
   4.2. Key Transport Algorithms 11
      4.2.1. RSA 11
   4.3. Symmetric Key-Encryption Algorithms 11
      4.3.1. AES Key Wrap 12
   4.4. Key Derivation Algorithms 12
      4.4.1. PBKDF2 13
5. Content-Encryption Algorithms 13
   5.1. AES-CBC 13
6. Message Authentication Code Algorithms 14
   6.1. Password-Based MAC 14
      6.1.1. PasswordBasedMac 14
      6.1.2. PBMAC1 15
   6.2. Symmetric Key-Based MAC 15
      6.2.1. SHA2-Based HMAC 15

6.2.2. AES-GMAC 16

6.2.3. SHAKE-Based KMAC 16

7. Algorithm Use Profiles 17

7.1. Algorithm Profile for PKI Management Message Profiles in RFC 4210 20

7.2. Algorithm Profile for Lightweight CMP Profile 22

8. IANA Considerations 23

9. Security Considerations 23

10. References 24

10.1. Normative References 24

10.2. Informative References 27

Acknowledgements 27

Authors' Addresses 27

1. Introduction

Appendix D.2 of [RFC4210] contains a set of algorithms that is mandatory to be supported by implementations conforming to [RFC4210]. These algorithms were appropriate at the time CMP was released, but as cryptographic algorithms weaken over time, some of them should no longer be used. In general, new attacks are emerging due to research in cryptanalysis or an increase in computing power. New algorithms were introduced that are more resistant to today's attacks.

This document lists current cryptographic algorithms that can be used with CMP to offer an easier way to maintain the list of suitable algorithms over time.

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.

In the following sections, ASN.1 values and types are used to indicate where algorithm identifier and output values are provided. These ASN.1 values and types are defined in CMP [RFC4210], Certificate Request Message Format (CRMF) [RFC4211], CMP Updates [RFC9480], and Cryptographic Message Syntax (CMS) [RFC5652].

2. Message Digest Algorithms

This section provides references to object identifiers and conventions to be employed by CMP implementations that support SHA2 or SHAKE message digest algorithms.

Digest algorithm identifiers are located in the:

- hashAlg field of OOBCertHash and CertStatus,
- owf field of Challenge, PBMParameter, and DHBMParameter,
- digestAlgorithms field of SignedData, and
- digestAlgorithm field of SignerInfo.

Digest values are located in the:

- hashVal field of OOBCertHash,
- certHash field of CertStatus, and
- witness field of Challenge.

In addition, digest values are input to signature algorithms.

2.1. SHA2

The SHA2 algorithm family is defined in FIPS Pub 180-4 [NIST.FIPS.180-4].

The message digest algorithms SHA-224, SHA-256, SHA-384, and SHA-512 are identified by the following OIDs:

```
id-sha224 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 4 }
id-sha256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 1 }
id-sha384 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 2 }
id-sha512 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistalgorithm(4)
   hashalgs(2) 3 }
```

Specific conventions to be considered are specified in Section 2 of [RFC5754].

2.2. SHAKE

The SHA-3 family of hash functions is defined in FIPS Pub 202 [NIST.FIPS.202] and consists of the fixed output length variants SHA3-224, SHA3-256, SHA3-384, and SHA3-512, as well as the extendable-output functions (XOFs) SHAKE128 and SHAKE256. Currently, SHAKE128 and

SHAKE256 are the only members of the SHA3-family that are specified for use in X.509 certificates [RFC8692] and CMS [RFC8702] as one-way hash functions for use with RSASSA-PSS and ECDSA.

SHAKE is an extendable-output function, and FIPS Pub 202 [NIST.FIPS.202] prohibits using SHAKE as a general-purpose hash function. When SHAKE is used in CMP as a message digest algorithm, the output length **MUST** be 256 bits for SHAKE128 and 512 bits for SHAKE256.

The message digest algorithms SHAKE128 and SHAKE256 are identified by the following OIDs:

```
id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
   hashalgs(2) 11 }
id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
   us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
   hashalgs(2) 12 }
```

Specific conventions to be considered are specified in Section 3.1 of [RFC8702].

3. Signature Algorithms

This section provides references to object identifiers and conventions to be employed by CMP implementations that support signature algorithms like RSA, ECDSA, or EdDSA.

The signature algorithm is referred to as MSG_SIG_ALG in Appendices D and E of [RFC4210], in the Lightweight CMP Profile [RFC9483], and in Section 7.2.

Signature algorithm identifiers are located in the:

- protectionAlg field of PKIHeader,
- algorithmIdentifier field of POPOSigningKey, and
- signatureAlgorithm field of CertificationRequest, SignKeyPairTypes, and SignerInfo

Signature values are located in the:

- protection field of PKIMessage,
- signature field of POPOSigningKey, and
- signature field of CertificationRequest and SignerInfo.

3.1. RSA

The RSA (RSASSA-PSS and PKCS #1 version 1.5) signature algorithm is defined in [RFC8017].

The algorithm identifier for RSASAA-PSS signatures used with SHA2 message digest algorithms is identified by the following OID:

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

Specific conventions to be considered are specified in [RFC4056].

The signature algorithm RSASSA-PSS used with SHAKE message digest algorithms is identified by the following OIDs:

```
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 }
```

Specific conventions to be considered are specified in Section 3.2.1 of [RFC8702].

The signature algorithm PKCS #1 version 1.5 used with SHA2 message digest algorithms is identified by the following OIDs:

```
sha224WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 14 }
sha256WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 11 }
sha384WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 12 }
sha512WithRSAEncryption  OBJECT IDENTIFIER  ::=  { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 13 }
```

Specific conventions to be considered are specified in Section 3.2 of [RFC5754].

3.2. ECDSA

The ECDSA signature algorithm is defined in FIPS Pub 186-5 [NIST.FIPS.186-5].

The signature algorithm ECDSA used with SHA2 message digest algorithms is identified by the following OIDs:

```
ecdsa-with-SHA224 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 1 }
ecdsa-with-SHA256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 2 }
ecdsa-with-SHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 3 }
ecdsa-with-SHA512 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 4 }
```

As specified in [RFC5480], the NIST-recommended curves are identified by the following OIDs:

```
secp192r1 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) curves(3) prime(1) 1 }
secp224r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 33 }
secp256r1 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) curves(3) prime(1) 7 }
secp384r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 34 }
secp521r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 35 }
```

Specific conventions to be considered are specified in Section 3.3 of [RFC5754].

The signature algorithm ECDSA used with SHAKE message digest algorithms is identified by the following OIDs:

```
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 }
```

Specific conventions to be considered are specified in Section 3.2.2 of [RFC8702].

3.3. EdDSA

The EdDSA signature algorithm is defined in Section 3.3 of [RFC8032] and FIPS Pub 186-5 [NIST.FIPS.186-5].

The signature algorithm Ed25519 that **MUST** be used with SHA-512 message digest algorithms is identified by the following OIDs:

```
id-Ed25519 OBJECT IDENTIFIER  ::=  { iso(1)
   identified-organization(3) thawte(101) 112 }
```

The signature algorithm Ed448 that **MUST** be used with SHAKE256 message digest algorithms is identified by the following OIDs:

```
id-Ed448 OBJECT IDENTIFIER  ::=  { iso(1)
   identified-organization(3) thawte(101) 113 }
```

Specific conventions to be considered are specified in [RFC8419].

Note: The hash algorithm used to calculate the certHash in certConf messages **MUST** be SHA512 if the certificate to be confirmed has been signed using Ed25519 or SHAKE256 with d=512 if the certificate to be confirmed has been signed using Ed448.

4. Key Management Algorithms

CMP utilizes the following general key management techniques: key agreement, key transport, and passwords.

CRMF [RFC4211] and CMP Updates [RFC9480] promote the use of CMS EnvelopedData [RFC5652] by deprecating the use of EncryptedValue.

4.1. Key Agreement Algorithms

The key agreement algorithm is referred to as PROT_ENC_ALG in Appendices D and E of [RFC4210] and as KM_KA_ALG in the Lightweight CMP Profile [RFC9483] and Section 7.

Key agreement algorithms are only used in CMP when using CMS EnvelopedData [RFC5652] together with the key agreement key management technique. When a key agreement algorithm is used, a key-encryption algorithm (Section 4.3) is needed next to the content-encryption algorithm (Section 5).

Key agreement algorithm identifiers are located in the:

- keyEncryptionAlgorithm field of KeyAgreeRecipientInfo.

Key wrap algorithm identifiers are located in the:

- KeyWrapAlgorithm parameters within keyEncryptionAlgorithm field of KeyAgreeRecipientInfo.

Wrapped content-encryption keys are located in the:

- encryptedKey field of RecipientEncryptedKeys.

4.1.1. Diffie-Hellman

The Diffie-Hellman (DH) key agreement is defined in [RFC2631] and **SHALL** be used in the ephemeral-static variants, as specified in [RFC3370]. Static-static variants **SHALL NOT** be used.

The DH key agreement algorithm is identified by the following OID:

```
id-alg-ESDH OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 5 }
```

Specific conventions to be considered are specified in Section 4.1 of [RFC3370].

4.1.2. ECDH

The Elliptic Curve Diffie-Hellman (ECDH) key agreement is defined in [RFC5753] and **SHALL** be used in the ephemeral-static variant, as specified in [RFC5753], or the 1-Pass Elliptic Curve Menezes-Qu-Vanstone (ECMQV) variant, as specified in [RFC5753]. Static-static variants **SHALL NOT** be used.

The ECDH key agreement algorithm used together with NIST-recommended SECP curves are identified by the following OIDs:

```
dhSinglePass-stdDH-sha224kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 11(11) 0 }
dhSinglePass-stdDH-sha256kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 11(11) 1 }
dhSinglePass-stdDH-sha384kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 11(11) 2 }
dhSinglePass-stdDH-sha512kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 11(11) 3 }
dhSinglePass-cofactorDH-sha224kdf-scheme OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) certicom(132) schemes(1)
   14(14) 0 }
dhSinglePass-cofactorDH-sha256kdf-scheme OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) certicom(132) schemes(1)
   14(14) 1 }
dhSinglePass-cofactorDH-sha384kdf-scheme OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) certicom(132) schemes(1)
   14(14) 2 }
dhSinglePass-cofactorDH-sha512kdf-scheme OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) certicom(132) schemes(1)
   14(14) 3 }
mqvSinglePass-sha224kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 15(15) 0 }
mqvSinglePass-sha256kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 15(15) 1 }
mqvSinglePass-sha384kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 15(15) 2 }
mqvSinglePass-sha512kdf-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) schemes(1) 15(15) 3 }
```

As specified in [RFC5480], the NIST-recommended SECP curves are identified by the following OIDs:

```
secp192r1 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) curves(3) prime(1) 1 }
secp224r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 33 }
secp256r1 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) ansi-X9-62(10045) curves(3) prime(1) 7 }
secp384r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 34 }
secp521r1 OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) certicom(132) curve(0) 35 }
```

Specific conventions to be considered are specified in [RFC5753].

The ECDH key agreement algorithm used together with curve25519 or curve448 is identified by the following OIDs:

```
   id-X25519 OBJECT IDENTIFIER ::= { iso(1)
      identified-organization(3) thawte(101) 110 }
   id-X448 OBJECT IDENTIFIER ::= { iso(1)
      identified-organization(3) thawte(101) 111 }
```

Specific conventions to be considered are specified in [RFC8418].

4.2. Key Transport Algorithms

The key transport algorithm is also referred to as PROT_ENC_ALG in Appendices D and E of [RFC4210] and as KM_KT_ALG in the Lightweight CMP Profile [RFC9483] and Section 7.

Key transport algorithms are only used in CMP when using CMS [RFC5652] EnvelopedData together with the key transport key management technique.

Key transport algorithm identifiers are located in the:

- keyEncryptionAlgorithm field of KeyTransRecipientInfo.

Key transport encrypted content-encryption keys are located in the:

- encryptedKey field of KeyTransRecipientInfo.

4.2.1. RSA

The RSA key transport algorithm is the RSA encryption scheme defined in [RFC8017].

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 algorithm identifier for RSAES-OAEP is:

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

Further conventions to be considered for PKCS #1 v1.5 are specified in Section 4.2.1 of [RFC3370] and for RSAES-OAEP in [RFC3560].

4.3. Symmetric Key-Encryption Algorithms

The symmetric key-encryption algorithm is also referred to as KM_KW_ALG in Section 7.2 and the Lightweight CMP Profile [RFC9483].

As the symmetric key-encryption key management technique is not used by CMP, the symmetric key-encryption algorithm is only needed when using the key agreement or password-based key management technique with CMS [RFC5652] EnvelopedData.

Key wrap algorithm identifiers are located in the:

- parameters field of the KeyEncryptionAlgorithmIdentifier of KeyAgreeRecipientInfo and PasswordRecipientInfo.

Wrapped content-encryption keys are located in the:

- encryptedKey field of RecipientEncryptedKeys (for key agreement) and PasswordRecipientInfo (for password-based key management).

4.3.1. AES Key Wrap

The AES encryption algorithm is defined in FIPS Pub 197 [NIST.FIPS.197], and the key wrapping is defined in [RFC3394].

AES key encryption has the algorithm identifier:

```
    id-aes128-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
       country(16) us(840) organization(1) gov(101) csor(3)
       nistAlgorithm(4) aes(1) 5 }
    id-aes192-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
       country(16) us(840) organization(1) gov(101) csor(3)
       nistAlgorithm(4) aes(1) 25 }
    id-aes256-wrap OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
       country(16) us(840) organization(1) gov(101) csor(3)
       nistAlgorithm(4) aes(1) 45 }
```

The underlying encryption functions for the key wrap and content-encryption algorithms (as specified in Section 5) 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); see [RFC8551].

Further conventions to be considered for AES key wrap are specified in Section 2.2 of [RFC3394] and Section 2.3.2 of [RFC3565].

4.4. Key Derivation Algorithms

The key derivation algorithm is also referred to as KM_KD_ALG in Section 7.2 and the Lightweight CMP Profile [RFC9483].

Key derivation algorithms are only used in CMP when using CMS EnvelopedData [RFC5652] together with the password-based key management technique.

Key derivation algorithm identifiers are located in the:

- keyDerivationAlgorithm field of PasswordRecipientInfo.

When using the password-based key management technique with EnvelopedData as specified in CMP Updates [RFC9480] together with PKIProtection based on the message authentication code (MAC), the salt for the password-based MAC and key derivation function (KDF) must be chosen independently to ensure usage of independent symmetric keys.

4.4.1. PBKDF2

Password-based key derivation function 2 (PBKDF2) is defined in [RFC8018].

PBKDF2 has the algorithm identifier:

```
id-PBKDF2 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
   rsadsi(113549) pkcs(1) pkcs-5(5) 12 }
```

Further conventions to be considered for PBKDF2 are specified in Section 4.4.1 of [RFC3370] and Section 5.2 of [RFC8018].

5. Content-Encryption Algorithms

The content-encryption algorithm is also referred to as PROT_SYM_ALG in Appendices D and E of [RFC4210], in the Lightweight CMP Profile [RFC9483], and in Section 7.

Content-encryption algorithms are used in CMP when using CMS EnvelopedData [RFC5652] to transport a signed private key package in case of central key generation or key archiving, a certificate to facilitate implicit proof-of-possession, or a revocation passphrase in encrypted form.

Content-encryption algorithm identifiers are located in the:

- contentEncryptionAlgorithm field of EncryptedContentInfo.

Encrypted content is located in the:

- encryptedContent field of EncryptedContentInfo.

5.1. AES-CBC

The AES encryption algorithm is defined in FIPS Pub 197 [NIST.FIPS.197].

AES Cipher Block Chaining (AES-CBC) content encryption has the algorithm identifier:

```
    id-aes128-CBC OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
       country(16) us(840) organization(1) gov(101) csor(3)
       nistAlgorithm(4) aes(1) 2 }
    id-aes192-CBC OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
       country(16) us(840) organization(1) gov(101) csor(3)
       nistAlgorithm(4) aes(1)22 }
    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 }
```

Specific conventions to be considered for AES-CBC content encryption are specified in [RFC3565].

6. Message Authentication Code Algorithms

The message authentication code (MAC) is either used for shared secret-based CMP message protection or together with the password-based key derivation function (PBKDF2).

The MAC algorithm is also referred to as MSG_MAC_ALG in Section 7, Appendices D and E of [RFC4210], and the Lightweight CMP Profile [RFC9483].

6.1. Password-Based MAC

Password-based message authentication code (MAC) algorithms combine the derivation of a symmetric key from a password or other shared secret information and a symmetric key-based MAC function, as specified in Section 6.2, using this derived key.

MAC algorithm identifiers are located in the:

- protectionAlg field of PKIHeader.

Message authentication code values are located in the:

- PKIProtection field of PKIMessage.

6.1.1. PasswordBasedMac

The PasswordBasedMac algorithm is defined in Section 5.1.3.1 of [RFC4210], Section 4.4 of [RFC4211], and "Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)" [RFC9045].

The PasswordBasedMac algorithm is identified by the following OID:

```
    id-PasswordBasedMac OBJECT IDENTIFIER ::= { iso(1) member-body(2)
       us(840) nt(113533) nsn(7) algorithms(66) 13 }
```

Further conventions to be considered for PasswordBasedMac are specified in Section 5.1.3.1 of [RFC4210], Section 4.4 of [RFC4211], and "Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)" [RFC9045].

6.1.2. PBMAC1

Password-Based Message Authentication Code 1 (PBMAC1) is defined in [RFC8018]. PBMAC1 combines a password-based key derivation function, like PBKDF2 (Section 4.4.1), with an underlying symmetric key-based message authentication scheme.

PBMAC1 has the following OID:

```
   id-PBMAC1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
      rsadsi(113549) pkcs(1) pkcs-5(5) 14 }
```

Specific conventions to be considered for PBMAC1 are specified in Section 7.1 and Appendix A.5 of [RFC8018].

6.2. Symmetric Key-Based MAC

Symmetric key-based message authentication code (MAC) algorithms are used for deriving the symmetric encryption key when using PBKDF2, as described in Section 4.4.1, as well as with password-based MAC, as described in Section 6.1.

MAC algorithm identifiers are located in the:

- protectionAlg field of PKIHeader,
- messageAuthScheme field of PBMAC1,
- mac field of PBMParameter, and
- prf field of PBKDF2-params.

MAC values are located in the:

- PKIProtection field of PKIMessage.

6.2.1. SHA2-Based HMAC

The HMAC algorithm is defined in [RFC2104] and FIPS Pub 198-1 [NIST.FIPS.198-1].

The HMAC algorithm used with SHA2 message digest algorithms is identified by the following OIDs:

```
id-hmacWithSHA224 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 8 }
id-hmacWithSHA256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 9 }
id-hmacWithSHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 10 }
id-hmacWithSHA512 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) digestAlgorithm(2) 11 }
```

Specific conventions to be considered for SHA2-based HMAC are specified in Section 3.1 of [RFC4231].

6.2.2. AES-GMAC

The AES-GMAC algorithm is defined in FIPS Pub 197 [NIST.FIPS.197] and NIST SP 800-38d [NIST.SP.800-38d].

Note: The AES-GMAC **MUST NOT** be used twice with the same parameter set, especially the same nonce. Therefore, it **MUST NOT** be used together with PBKDF2. When using it with PBMAC1, it **MUST** be ensured that the AES-GMAC is only used as a message authentication scheme and not for the key derivation function PBKDF2.

The AES-GMAC algorithm is identified by the following OIDs:

```
id-aes128-GMAC OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 9 }
id-aes192-GMAC OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 29 }
id-aes256-GMAC OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) aes(1) 49 }
```

Specific conventions to be considered for the AES-GMAC are specified in [RFC9044].

6.2.3. SHAKE-Based KMAC

The KMAC algorithm is defined in [RFC8702] and FIPS SP 800-185 [NIST.SP.800-185]].

The SHAKE-based KMAC algorithm is identified by the following OIDs:

```
id-KMACWithSHAKE128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) hashAlgs(2) 19 }
id-KMACWithSHAKE256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
   country(16) us(840) organization(1) gov(101) csor(3)
   nistAlgorithm(4) hashAlgs(2) 20 }
```

Specific conventions to be considered for KMAC with SHAKE are specified in Section 3.4 of [RFC8702].

7. Algorithm Use Profiles

This section provides profiles of algorithms and respective conventions for different application use cases.

Recommendations like those described in Table 2 of NIST SP 800-57 "Recommendation for Key Management" [NIST.SP.800-57pt1r5] and Section 4.6 of ECRYPT "Algorithms, Key Size and Protocols Report (2018)" [ECRYPT.CSA.D5.4] provide general information on current cryptographic algorithms.

The overall cryptographic strength of CMP implementations will depend on several factors, including:

- capabilities of the end entity: What kind of algorithms does the end entity support? The cryptographic strength of the system **SHOULD** be at least as strong as the algorithms and keys used for the certificate being managed.
- algorithm profile: The overall strength of the profile will be the strength of the weakest algorithm it contains.
- message protection: The overall strength of the CMP message protection.
  - MAC-based protection: The entropy of the shared secret information or password when MAC-based message protection is used (MSG_MAC_ALG).
  - signature-based protection: The strength of the key pair and signature algorithm when signature-based protection is used (MSG_SIG_ALG).
  - protection of centrally generated keys: The strength of the algorithms used for the key management technique (Section 7.1: PROT_ENC_ALG or Section 7.2: KM_KA_ALG, KM_KT_ALG, KM_KD_ALG) and the encryption of the content-encryption key and private key (Section 7.1: SYM_PENC_ALG, PROT_SYM_ALG or Section 7.2: KM_KW_ALG, PROT_SYM_ALG).

The following table shows the algorithms listed in this document sorted by their bits of security. If an implementation intends to enroll and manage certificates for keys of a specific security, it **SHALL** implement and use algorithms of at least that strength for the respective PKI management operation. If one row does not provide a suitable algorithm, the implementer **MUST** choose one offering more bits of security.

Bits of Security	RSA or DH	Elliptic Curve Cryptography	Hash Function or XOF with Specified Output Length (d)	Symmetric Encryption
112	RSA2048, DH(2048)	ECDSA/ECDH (secp224r1)	SHA-224	
128	RSA3072, DH(3072)	ECDSA/ECDH (secp256r1), Ed25519/X25519 (curve25519)	SHA-256, SHAKE128(d=256)	AES-128
192		ECDSA/ECDH (secp384r1)	SHA-384	AES-192
224		Ed448/X448 (curve448)		
256		ECDSA/ECDH (secp521r1)	SHA-512, SHAKE256(d=512)	AES-256

Table 1: Cryptographic Algorithms Sorted by Their Bits of Security

The following table shows the cryptographic algorithms sorted by their usage in CMP and with more details.

Bits of Security	Key Types to Be Certified	CMP Protection MSG_SIG_ALG, MSG_MAC_ALG	Key Management Technique PROT_ENC_ALG or KM_KA_ALG, KM_KT_ALG, KM_KD_ALG	Key-Wrap and Symmetric Encryption PROT_SYM_ALG, SYM_PENC_ALG or KM_KW_ALG
112	RSA2048, secp224r1	RSASSA-PSS (2048, SHA-224 or SHAKE128 (d=256)), RSAEncryption (2048, SHA-224), ECDSA (secp224r1, SHA-224 or SHAKE128 (d=256)), PBMAC1 (HMAC-SHA-224)	DH(2048), RSAES-OAEP (2048, SHA-224), RSAEncryption (2048, SHA-224), ECDH (secp224r1, SHA-224), PBKDF2 (HMAC-SHA-224)	
128	RSA3072, secp256r1, curve25519	RSASSA-PSS (3072, SHA-256 or SHAKE128 (d=256)), RSAEncryption (3072, SHA-256), ECDSA (secp256r1, SHA-256 or SHAKE128 (d=256)), Ed25519 (SHA-512), PBMAC1 (HMAC-SHA-256)	DH(3072), RSAES-OAEP (3072, SHA-256), RSAEncryption (3072, SHA-256), ECDH (secp256r1, SHA-256), X25519, PBKDF2 (HMAC-SHA-256)	AES-128
192	secp384r1	ECDSA (secp384r1, SHA-384), PBMAC1 (HMAC-SHA-384)	ECDH (secp384r1, SHA-384), PBKDF2 (HMAC-SHA-384)	AES-192
224	curve448	Ed448 (SHAKE256)	X448	

Bits of Security	Key Types to Be Certified	CMP Protection MSG_SIG_ALG, MSG_MAC_ALG	Key Management Technique PROT_ENC_ALG or KM_KA_ALG, KM_KT_ALG, KM_KD_ALG	Key-Wrap and Symmetric Encryption PROT_SYM_ALG, SYM_PENC_ALG or KM_KW_ALG
256	secp521r1	ECDSA (secp521r1, SHA-512 or SHAKE256 (d=512)), PBMAC1 (HMAC-SHA-512)	ECDH (secp521r1, SHA-512), PBKDF2 (HMAC-SHA-512)	AES-256

Table 2: Cryptographic Algorithms Sorted by Their Bits of Security and Usage by CMP

To avoid consuming too many computational resources, choosing a set of algorithms offering roughly the same level of security is recommended. Below are several algorithm profiles that are balanced, assuming the implementer chooses MAC secrets and/or certificate profiles of at least equivalent strength.

7.1. Algorithm Profile for PKI Management Message Profiles in RFC 4210

The following table updates the definitions of algorithms used within PKI Management Message Profiles, as defined in Appendix D.2 of [RFC4210].

The columns in the table are:

Name: An identifier used for message profiles

Use: Description of where and for what the algorithm is used

Mandatory: Algorithms that **MUST** be supported by conforming implementations

Optional: Algorithms that are **OPTIONAL** to support

Deprecated: Algorithms from [RFC4210] that **SHOULD NOT** be used any more

Name	Use	Mandatory	Optional	Deprecated
MSG_SIG_ALG	protection of PKI messages using signatures	RSA	ECDSA, EdDSA	DSA, combinations with MD5 and SHA-1

Name	Use	Mandatory	Optional	Deprecated
MSG_MAC_ALG	protection of PKI messages using MACs	PBMAC1	PasswordBasedMac, HMAC, KMAC	X9.9
SYM_PENC_ALG	symmetric encryption of an end entity's private key where the symmetric key is distributed out of band	AES-wrap		3-DES(3-key-EDE, CBC Mode), RC5, CAST-128
PROT_ENC_ALG	asymmetric algorithm used for encryption of (symmetric keys for encryption of) private keys transported in PKIMessages	DH	ECDH, RSA	
PROT_SYM_ALG	symmetric encryption algorithm used for encryption of private key bits (a key of this type is encrypted using PROT_ENC_ALG)	AES-CBC		3-DES(3-key-EDE, CBC Mode), RC5, CAST-128

Table 3: Algorithms Used within Appendix D.2 of RFC 4210

The following are the mandatory algorithm identifiers and specifications:

RSA: sha256WithRSAEncryption with 2048 bit, see Section 3.1

PasswordBasedMac: id-PasswordBasedMac, see Section 6.1 (with id-sha256 as the owf parameter, see Section 2.1 and id-hmacWithSHA256 as the mac parameter, see Section 6.2.1)

PBMAC1: id-PBMAC1, see Section 6.1.2 (with id-PBKDF2 as the key derivation function, see Section 4.4.1 and id-hmacWithSHA256 as the message authentication scheme, see Section 6.2.1). It is **RECOMMENDED** to prefer the usage of PBMAC1 instead of PasswordBasedMac.

DH: id-alg-ESDH, see Section 4.1.1

AES-wrap: id-aes128-wrap, see Section 4.3.1

AES-CBC: id-aes128-CBC, see Section 5.1

7.2. Algorithm Profile for Lightweight CMP Profile

The following table contains definitions of algorithms that **MAY** be supported by implementations of the Lightweight CMP Profile [RFC9483].

As the set of algorithms to be used for implementations of the Lightweight CMP Profile heavily depends on the PKI management operations implemented, the certificates used for message protection, and the certificates to be managed, this document will not specify a specific set that is mandatory to support for conforming implementations.

The columns in the table are:

Name: An identifier used for message profiles

Use: Description of where and for what the algorithm is used

Examples: Lists the algorithms, as described in this document. The list of algorithms depends on the set of PKI management operations to be implemented.

Note: It is **RECOMMENDED** to prefer the usage of PBMAC1 instead of PasswordBasedMac.

Name	Use	Examples
MSG_SIG_ALG	protection of PKI messages using signatures and for SignedData, e.g., a private key transported in PKIMessages	RSA, ECDSA, EdDSA
MSG_MAC_ALG	protection of PKI messages using MACing	PasswordBasedMac (see Section 9), PBMAC1, HMAC, KMAC
KM_KA_ALG	asymmetric key agreement algorithm used for agreement of a symmetric key for use with KM_KW_ALG	DH, ECDH
KM_KT_ALG	asymmetric key-encryption algorithm used for transport of a symmetric key for PROT_SYM_ALG	RSA
KM_KD_ALG	symmetric key derivation algorithm used for derivation of a symmetric key for use with KM_KW_ALG	PBKDF2
KM_KW_ALG	algorithm to wrap a symmetric key for PROT_SYM_ALG	AES-wrap

Name	Use	Examples
PROT_SYM_ALG	symmetric content-encryption algorithm used for encryption of EnvelopedData, e.g., a private key transported in PKIMessages	AES-CBC

Table 4: Algorithms Used within the Lightweight CMP Profile

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

This document lists many cryptographic algorithms usable with CMP to offer implementers a more up-to-date choice. Finally, the algorithms to be supported also heavily depend on the certificates and PKI management operations utilized in the target environment. The algorithm with the lowest security strength and the entropy of shared secret information defines the security of the overall solution; see Section 7.

When using MAC-based message protection, the use of PBMAC1 is preferable to that of PasswordBasedMac. First, PBMAC1 is a well-known scrutinized algorithm, which is not true for PasswordBasedMac. Second, the PasswordBasedMac algorithm as specified in Section 4.4 of [RFC4211] is essentially PBKDF1 (as defined in Section 5.1 of [RFC8018]) with an HMAC step at the end. Here, we update to use the PBKDF2-HMAC construct defined as PBMAC1 in [RFC8018]. PBKDF2 is superior to PBKDF1 in an improved internal construct for iterated hashing and in removing PBKDF1's limitation of only being able to derive keys up to the size of the underlying hash function. Additionally, PBKDF1 is not recommended for new applications as stated in Section 5.1 of [RFC8018] and is no longer an approved algorithm by most standards bodies. Therefore, it presents difficulties to implementers who are submitting their CMP implementations for certification, hence moving to a PBKDF2-based mechanism. This change is in alignment with [RFC9045], which updates [RFC4211] to allow the use of PBMAC1 in CRMF.

The AES-GMAC **MUST NOT** be used as the pseudorandom function (PRF) in PBKDF2; the use of the AES-GMAC more than once with the same key and the same nonce will break the security.

In Section 7 of this document, there is also an update to Appendix D.2 of [RFC4210] and a set of algorithms that **MAY** be supported when implementing the Lightweight CMP Profile [RFC9483]. It is recognized that there may be older CMP implementations in use that conform to the algorithm use profile from Appendix D.2 of [RFC4210]. For example, the use of AES is now mandatory for PROT_SYM_ALG, while 3-DES was mandatory in [RFC4210]. Therefore, it is expected that many CMP systems may already support the recommended algorithms in this specification. In such systems, the weakened algorithms should be disabled from further use. If critical systems cannot be immediately updated to conform to the recommended algorithm use profile, it is recommended that a plan to migrate the infrastructure to conforming profiles be adopted as soon as possible.

Symmetric key-based MAC algorithms as described in Section 6.2 **MAY** be used as MSG_MAC_ALG. The implementer **MUST** choose a suitable PRF and ensure that the key has sufficient entropy to match the overall security level of the algorithm profile. These considerations are outside the scope of the profile.

10. References

10.1. Normative References

[NIST.FIPS.180-4] Dang, Q. H. and NIST, "Secure Hash Standard", NIST Federal Information Processing Standards Publications 180-4, DOI 10.6028/NIST.FIPS.180-4, July 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[NIST.FIPS.186-5] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS PUB 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf>.

[NIST.FIPS.197] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", NIST FIPS 197, DOI 10.6028/NIST.FIPS.197, November 2001, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>.

[NIST.FIPS.198-1] National Institute of Standards and Technology (NIST), "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198-1, DOI 10.6028/NIST.FIPS.198-1, July 2008, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf>.

[NIST.FIPS.202] Dworkin, M. J. and NIST, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST Federal Information Processing Standards Publications 202, DOI 10.6028/NIST.FIPS.202, July 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.

[NIST.SP.800-185]] Kelsey, J., Change, S., Perlner, R., and NIST, "SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash", NIST Special Publications (General) 800-185, DOI 10.6028/NIST.SP.800-185, December 2016, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

[NIST.SP.800-38d] Dworkin, M J. and NIST, "Recommendation for block cipher modes of operation :GaloisCounter Mode (GCM) and GMAC", NIST Special Publications (General) 800-38d, DOI 10.6028/NIST.SP.800-38d, 2007, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[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>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <https://www.rfc-editor.org/info/rfc4231>.

[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>.

[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>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[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>.

[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>.

[RFC8418] Housley, R., "Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)", RFC 8418, DOI 10.17487/RFC8418, August 2018, <https://www.rfc-editor.org/info/rfc8418>.

[RFC8419] Housley, R., "Use of Edwards-Curve Digital Signature Algorithm (EdDSA) Signatures in the Cryptographic Message Syntax (CMS)", RFC 8419, DOI 10.17487/RFC8419, August 2018, <https://www.rfc-editor.org/info/rfc8419>.

[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>.

[RFC8692] Kampanakis, P. and Q. Dang, "Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA Using SHAKEs", RFC 8692, DOI 10.17487/RFC8692, December 2019, <https://www.rfc-editor.org/info/rfc8692>.

[RFC8702] Kampanakis, P. and Q. Dang, "Use of the SHAKE One-Way Hash Functions in the Cryptographic Message Syntax (CMS)", RFC 8702, DOI 10.17487/RFC8702, January 2020, <https://www.rfc-editor.org/info/rfc8702>.

[RFC9044] Housley, R., "Using the AES-GMAC Algorithm with the Cryptographic Message Syntax (CMS)", RFC 9044, DOI 10.17487/RFC9044, June 2021, <https://www.rfc-editor.org/info/rfc9044>.

[RFC9045] Housley, R., "Algorithm Requirements Update to the Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 9045, DOI 10.17487/RFC9045, June 2021, <https://www.rfc-editor.org/info/rfc9045>.

[RFC9480] Brockhaus, H., von Oheimb, D., and J. Gray, "Certificate Management Protocol (CMP) Updates", RFC 9480, DOI 10.17487/RFC9480, November 2023, <https://www.rfc-editor.org/info/rfc9480>.

[RFC9483] Brockhaus, H., von Oheimb, D., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", RFC 9483, DOI 10.17487/RFC9483, November 2023, <https://www.rfc-editor.org/info/rfc9483>.

10.2. Informative References

[ECRYPT.CSA.D5.4] University of Bristol, "Algorithms, Key Size and Protocols Report (2018)", March 2015, <https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf>.

[NIST.SP.800-57pt1r5] Barker, E. and NIST, "Recommendation for key management:part 1 - general", NIST Special Publications (General) 800-57pt1r5, DOI 10.6028/NIST.SP.800-57pt1r5, May 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>.

Acknowledgements

Thanks to Russ Housley for his work on [RFC9044] and [RFC9045] upon which this RFC relies heavily.

May thanks also to all reviewers like Serge Mister, Mark Ferreira, Yuefei Lu, Tomas Gustavsson, Lijun Liao, David von Oheimb, and Steffen Fries for their input and feedback to this document. Apologies to all not mentioned reviewers and supporters.

Authors' Addresses

Hendrik Brockhaus
Siemens AG
Werner-von-Siemens-Strasse 1
80333 Munich
Germany
Email: hendrik.brockhaus@siemens.com
URI: https://www.siemens.com

Hans Aschauer
Siemens AG
Werner-von-Siemens-Strasse 1
80333 Munich
Germany
Email: hans.aschauer@siemens.com
URI: https://www.siemens.com

Mike Ounsworth
Entrust
1187 Park Place
Minneapolis, MN 55379
United States of America
Email: mike.ounsworth@entrust.com
URI: https://www.entrust.com

John Gray
Entrust
1187 Park Place
Minneapolis, MN 55379
United States of America
Email: john.gray@entrust.com
URI: https://www.entrust.com