Stream: Internet Engineering Task Force (IETF)
RFC: 9629
Updates: 5652
Category: Standards Track
Published: August 2024
ISSN: 2070-1721
Authors: R. Housley, *Vigil Security*; J. Gray, *Entrust*; 大久保 智史 (T. Okubo), *Penguin Securities Pte. Ltd.*

RFC 9629
Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS)

Abstract

The Cryptographic Message Syntax (CMS) supports key transport and key agreement algorithms. In recent years, cryptographers have been specifying Key Encapsulation Mechanism (KEM) algorithms, including quantum-secure KEM algorithms. This document defines conventions for the use of KEM algorithms by the originator and recipients to encrypt and decrypt CMS content. This document updates RFC 5652.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in 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/rfc9629.

Copyright Notice

Table of Contents

1. Introduction

This document updates "Cryptographic Message Syntax (CMS)" [RFC5652].

The CMS enveloped-data content type [RFC5652] and the CMS authenticated-enveloped-data content type [RFC5083] support both key transport and key agreement algorithms to establish the key used to encrypt and decrypt the content. In recent years, cryptographers have been

specifying Key Encapsulation Mechanism (KEM) algorithms, including quantum-secure KEM algorithms. This document defines conventions for the use of KEM algorithms for the CMS enveloped-data content type and the CMS authenticated-enveloped-data content type.

A KEM algorithm is a one-pass (store-and-forward) mechanism for transporting random keying material to a recipient using the recipient's public key. This means that the originator and the recipients do not need to be online at the same time. The recipient's private key is needed to recover the random keying material, which is then treated as a pairwise shared secret (ss) between the originator and recipient.

The KEMRecipientInfo structure defined in this document uses the pairwise shared secret as an input to a key derivation function (KDF) to produce a pairwise key-encryption key (KEK). Then, the pairwise KEK is used to encrypt a content-encryption key (CEK) or a content-authenticated-encryption key (CAEK) for that recipient. All of the recipients receive the same CEK or CAEK.

In this environment, security depends on three things. First, the KEM algorithm must be secure against adaptive chosen ciphertext attacks. Second, the key-encryption algorithm must provide confidentiality and integrity protection. Third, the choices of the KDF and the key-encryption algorithm need to provide the same level of security as the KEM algorithm.

A KEM algorithm provides three functions:

KeyGen() -> (pk, sk):
: Generate the public key (pk) and a private key (sk).

Encapsulate(pk) -> (ct, ss):
: Given the recipient's public key (pk), produce a ciphertext (ct) to be passed to the recipient and shared secret (ss) for the originator.

Decapsulate(sk, ct) -> ss:
: Given the private key (sk) and the ciphertext (ct), produce the shared secret (ss) for the recipient.

To support a particular KEM algorithm, the CMS originator **MUST** implement the KEM Encapsulate() function.

To support a particular KEM algorithm, the CMS recipient **MUST** implement the KEM KeyGen() function and the KEM Decapsulate() function. The recipient's public key is usually carried in a certificate [RFC5280].

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 [X.680], which uses the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X.690].

1.3. CMS Version Numbers

As described in Section 1.3 of [RFC5652], 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 originator.

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. KEM Processing Overview

KEM algorithms can be used with three CMS content types: the enveloped-data content type [RFC5652], the authenticated-data content type [RFC5652], or the authenticated-enveloped-data content type [RFC5083]. For simplicity, the terminology associated with the enveloped-data content type will be used in this overview.

The originator randomly generates the CEK (or the CAEK), and then all recipients obtain that key as an encrypted object within the KEMRecipientInfo encryptedKey field explained in Section 3. All recipients use the originator-generated symmetric key to decrypt the CMS message.

A KEM algorithm and a key derivation function are used to securely establish a pairwise symmetric KEK, which is used to encrypt the originator-generated CEK (or the CAEK).

In advance, each recipient uses the KEM KeyGen() function to create a key pair. The recipient will often obtain a certificate [RFC5280] that includes the newly generated public key. Whether the public key is certified or not, the newly generated public key is made available to potential originators.

The originator establishes the CEK (or the CAEK) using these steps:

1. The CEK (or the CAEK) is generated at random.
2. For each recipient:
   - The recipient's public key is used with the KEM Encapsulate() function to obtain a pairwise shared secret (ss) and the ciphertext for the recipient.
   - The key derivation function is used to derive a pairwise symmetric KEK, from the pairwise ss and other data that is optionally sent in the ukm field.

- The KEK is used to encrypt the CEK for this recipient.

3. The CEK (or the CAEK) is used to encrypt the content for all recipients.

The recipient obtains the CEK (or the CAEK) using these steps:

1. The recipient's private key and the ciphertext are used with the KEM Decapsulate() function to obtain a pairwise ss.
2. The key derivation function is used to derive a pairwise symmetric KEK, from the pairwise ss and other data that is optionally sent in the ukm field.
3. The KEK is used to decrypt the CEK (or the CAEK).
4. The CEK (or the CAEK) is used to decrypt the content.

3. KEM Recipient Information

This document defines KEMRecipientInfo for use with KEM algorithms. As specified in Section 6.2.5 of [RFC5652], recipient information for additional key management techniques is represented in the OtherRecipientInfo type. Each key management technique is identified by a unique ASN.1 object identifier.

The object identifier associated with KEMRecipientInfo is:

```
id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) 13 }

id-ori-kem OBJECT IDENTIFIER ::= { id-ori 3 }
```

The KEMRecipientInfo type is:

```
KEMRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  rid RecipientIdentifier,
  kem KEMAlgorithmIdentifier,
  kemct OCTET STRING,
  kdf KeyDerivationAlgorithmIdentifier,
  kekLength INTEGER (1..65535),
  ukm [0] EXPLICIT UserKeyingMaterial OPTIONAL,
  wrap KeyEncryptionAlgorithmIdentifier,
  encryptedKey EncryptedKey }
```

The fields of the KEMRecipientInfo 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].

rid specifies the recipient's certificate or key that was used by the originator with the KEM Encapsulate() function. The RecipientIdentifier provides two alternatives for specifying the recipient's certificate [RFC5280], and thereby the recipient's public key. The recipient's certificate **MUST** contain a KEM public key. Therefore, a recipient X.509 version 3 certificate that contains a key usage extension **MUST** assert the keyEncipherment bit. The issuerAndSerialNumber alternative identifies the recipient's certificate by the issuer's distinguished name and the certificate serial number; the subjectKeyIdentifier alternative identifies the recipient's certificate by a key identifier. When an X.509 certificate is referenced, the key identifier matches the X.509 subjectKeyIdentifier extension value. When other certificate formats are referenced, the documents that specify the certificate format and their use with the CMS must include details on matching the key identifier to the appropriate certificate field. For recipient processing, implementations **MUST** support both of these alternatives for specifying the recipient's certificate. For originator processing, implementations **MUST** support at least one of these alternatives.

kem identifies the KEM algorithm, and any associated parameters, used by the originator. The KEMAlgorithmIdentifier is described in Section 4.

kemct is the ciphertext produced by the KEM Encapsulate() function for this recipient.

kdf identifies the key derivation function, and any associated parameters, used by the originator to generate the KEK. The KeyDerivationAlgorithmIdentifier is described in Section 10.1.6 of [RFC5652].

kekLength is the size of the KEK in octets. This value is one of the inputs to the key derivation function. The upper bound on the integer value is provided to make it clear to implementers that support for very large integer values is not needed. Implementations **MUST** confirm that the value provided is consistent with the key-encryption algorithm identified in the wrap field below.

ukm is optional user keying material. When the ukm value is provided, it is used as part of the info structure described in Section 5 to provide a context input to the key derivation function. For example, the ukm value could include a nonce, application-specific context information, or an identifier for the originator. A KEM algorithm may place requirements on the ukm value. Implementations that do not support the ukm field **SHOULD** gracefully discontinue processing when the ukm field is present. Note that this requirement expands the original purpose of the ukm described in Section 10.2.6 of [RFC5652]; it is not limited to being used with key agreement algorithms.

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

encryptedKey is the result of encrypting the CEK or the CAEK (the content-authenticated-encryption key, as discussed in [RFC5083]) with the KEK. EncryptedKey is an OCTET STRING.

4. KEM Algorithm Identifier

The KEMAlgorithmIdentifier type identifies a KEM algorithm used to establish a pairwise ss. The details of establishment depend on the KEM algorithm used. A key derivation function is used to transform the pairwise ss value into a KEK.

```
KEMAlgorithmIdentifier ::= AlgorithmIdentifier{ KEM-ALGORITHM, {...} }
```

5. Key Derivation

This section describes the conventions of using the KDF to compute the KEK for KEMRecipientInfo. For simplicity, the terminology used in the HKDF specification [RFC5869] is used here.

Many KDFs internally employ a one-way hash function. When this is the case, the hash function that is used is indirectly indicated by the KeyDerivationAlgorithmIdentifier. Other KDFs internally employ an encryption algorithm. When this is the case, the encryption that is used is indirectly indicated by the KeyDerivationAlgorithmIdentifier.

The KDF inputs are as follows:

IKM is the input keying material. It is a symmetric secret input to the KDF. The KDF may use a hash function or an encryption algorithm to generate a pseudorandom key. The algorithm used to derive the IKM is dependent on the algorithm identified in the KeyDerivationAlgorithmIdentifier.

L is the length of the output keying material in octets. L is identified in the kekLength of the KEMRecipientInfo. The value is dependent on the key-encryption algorithm used; the key-encryption algorithm is identified in the KeyEncryptionAlgorithmIdentifier.

info is contextual input to the KDF. The DER-encoded CMSORIforKEMOtherInfo structure is created from elements of the KEMRecipientInfo structure. CMSORIforKEMOtherInfo is defined as:

```
CMSORIforKEMOtherInfo ::= SEQUENCE {
  wrap KeyEncryptionAlgorithmIdentifier,
  kekLength INTEGER (1..65535),
  ukm [0] EXPLICIT UserKeyingMaterial OPTIONAL }
```

The CMSORIforKEMOtherInfo structure contains the following:

wrap identifies a key-encryption algorithm; the output of the key derivation function will be used as a key for this algorithm.

kekLength is the length of the KEK in octets; the output of the key derivation function will be exactly this size.

ukm is optional user keying material; see Section 3.

The KDF output is as follows:

OKM is the output keying material with the exact length of L octets. The OKM is the KEK that is used to encrypt the CEK or the CAEK.

An acceptable KDF **MUST** accept an IKM, L, and info as inputs. An acceptable KDF **MAY** also accept a salt input value, which is carried as a parameter to the KeyDerivationAlgorithmIdentifier if present. All of these inputs **MUST** influence the output of the KDF.

6. ASN.1 Modules

This section provides two ASN.1 modules [X.680]. The first ASN.1 module is an extension to the AlgorithmInformation-2009 module discussed in [RFC5912]; it defines the KEM-ALGORITHM CLASS. The second ASN.1 module defines the structures needed to use KEM algorithms with CMS [RFC5652].

The first ASN.1 module uses EXPLICIT tagging, and the second ASN.1 module uses IMPLICIT tagging.

Both ASN.1 modules follow the conventions established in [RFC5911], [RFC5912], and [RFC6268].

6.1. KEMAlgorithmInformation-2023 ASN.1 Module

```
<CODE BEGINS>
  KEMAlgorithmInformation-2023
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-kemAlgorithmInformation-2023(109) }

  DEFINITIONS EXPLICIT TAGS ::=
  BEGIN
  -- EXPORTS ALL;
  IMPORTS
    ParamOptions, PUBLIC-KEY, 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) } ;

  -- KEM-ALGORITHM
  --
  -- Describes the basic properties of a KEM algorithm
  --
  -- Suggested prefix for KEM algorithm objects is: kema-
  --
```

```
   --  &id - contains the OID identifying the KEM algorithm
   --  &Value - if present, contains a type definition for the kemct;
   --               if absent, implies that no ASN.1 encoding is
   --               performed on the kemct value
   --  &Params - if present, contains the type for the algorithm
   --               parameters; if absent, implies no parameters
   --  &paramPresence - parameter presence requirement
   --  &PublicKeySet - specifies which public keys are used with
   --               this algorithm
   --  &Ukm - if absent, type for user keying material
   --  &ukmPresence - specifies the requirements to define the UKM
   --               field
   --  &smimeCaps - contains the object describing how the S/MIME
   --               capabilities are presented.
   --
   -- Example:
   -- kema-kem-rsa KEM-ALGORITHM ::= {
   --    IDENTIFIER id-kem-rsa
   --    PARAMS TYPE RsaKemParameters ARE optional
   --    PUBLIC-KEYS { pk-rsa | pk-rsa-kem }
   --    UKM ARE optional
   --    SMIME-CAPS { TYPE GenericHybridParameters
   --        IDENTIFIED BY id-rsa-kem }
   -- }

   KEM-ALGORITHM ::= CLASS {
     &id              OBJECT IDENTIFIER UNIQUE,
     &Value           OPTIONAL,
     &Params          OPTIONAL,
     &paramPresence   ParamOptions DEFAULT absent,
     &PublicKeySet    PUBLIC-KEY OPTIONAL,
     &Ukm             OPTIONAL,
     &ukmPresence     ParamOptions DEFAULT absent,
     &smimeCaps       SMIME-CAPS OPTIONAL
   } WITH SYNTAX {
     IDENTIFIER &id
     [VALUE &Value]
     [PARAMS [TYPE &Params] ARE &paramPresence]
     [PUBLIC-KEYS &PublicKeySet]
     [UKM [TYPE &Ukm] ARE &ukmPresence]
     [SMIME-CAPS &smimeCaps]
   }

   END

<CODE ENDS>
```

6.2. CMS-KEMRecipientInfo-2023 ASN.1 Module

```
<CODE BEGINS>
  CMS-KEMRecipientInfo-2023
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0)
      id-mod-cms-kemri-2023(77) }

  DEFINITIONS IMPLICIT TAGS ::=
```

```
BEGIN
-- EXPORTS ALL;
IMPORTS
  OTHER-RECIPIENT, CMSVersion, RecipientIdentifier,
  EncryptedKey, KeyDerivationAlgorithmIdentifier,
  KeyEncryptionAlgorithmIdentifier, UserKeyingMaterial
    FROM CryptographicMessageSyntax-2010  -- RFC 6268
      { iso(1) member-body(2) us(840) rsadsi(113549)
        pkcs(1) pkcs-9(9) smime(16) modules(0)
        id-mod-cms-2009(58) }
  KEM-ALGORITHM
    FROM KEMAlgorithmInformation-2023  -- RFC 9629
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-kemAlgorithmInformation-2023(109) }
  AlgorithmIdentifier{}
    FROM AlgorithmInformation-2009  -- RFC 5912
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) } ;

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

SupportedOtherRecipInfo OTHER-RECIPIENT ::= { ori-KEM, ... }

ori-KEM OTHER-RECIPIENT ::= {
  KEMRecipientInfo IDENTIFIED BY id-ori-kem }

id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) 13 }

id-ori-kem OBJECT IDENTIFIER ::= { id-ori 3 }

--
-- KEMRecipientInfo
--

KEMRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  rid RecipientIdentifier,
  kem KEMAlgorithmIdentifier,
  kemct OCTET STRING,
  kdf KeyDerivationAlgorithmIdentifier,
  kekLength INTEGER (1..65535),
  ukm [0] EXPLICIT UserKeyingMaterial OPTIONAL,
  wrap KeyEncryptionAlgorithmIdentifier,
  encryptedKey EncryptedKey }

KEMAlgSet KEM-ALGORITHM ::= { ... }

KEMAlgorithmIdentifier ::=
  AlgorithmIdentifier{ KEM-ALGORITHM, {KEMAlgSet} }

--
-- CMSORIforKEMOtherInfo
--
```

```
  CMSORIforKEMOtherInfo ::= SEQUENCE {
    wrap KeyEncryptionAlgorithmIdentifier,
    kekLength INTEGER (1..65535),
    ukm [0] EXPLICIT UserKeyingMaterial OPTIONAL }

  END

<CODE ENDS>
```

7. Security Considerations

The security considerations discussed in [RFC5652] are applicable to this document.

To be appropriate for use with this specification, the KEM algorithm **MUST** explicitly be designed to be secure when the public key is used many times. For example, a KEM algorithm with a single-use public key is not appropriate, because the public key is expected to be carried in a long-lived certificate [RFC5280] and used over and over. Thus, KEM algorithms that offer indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) security are appropriate. A common design pattern for obtaining IND-CCA2 security with public key reuse is to apply the Fujisaki-Okamoto (FO) transform [FO] or a variant of the FO transform [HHK].

The KDF **SHOULD** offer at least the security level of the KEM.

The choice of the key-encryption algorithm and the size of the KEK **SHOULD** be made based on the security level provided by the KEM. The key-encryption algorithm and the KEK **SHOULD** offer at least the security level of the KEM.

KEM algorithms do not provide data origin authentication; therefore, when a KEM algorithm is used with the authenticated-data content type, the contents are delivered with integrity from an unknown source.

Implementations **MUST** protect the KEM private key, the KEK, and the CEK (or the CAEK). Compromise of the KEM private key may result in the disclosure of all contents protected with that KEM private key. However, compromise of the KEK, the CEK, or the CAEK may result in disclosure of the encrypted content of a single message.

The KEM produces the IKM input value for the KDF. This IKM value **MUST NOT** be reused for any other purpose. Likewise, any random value used by the KEM algorithm to produce the shared secret or its encapsulation **MUST NOT** be reused for any other purpose. That is, the originator **MUST** generate a fresh KEM shared secret for each recipient in the KEMRecipientInfo structure, including any random value used by the KEM algorithm to produce the KEM shared secret. In addition, the originator **MUST** discard the KEM shared secret, including any random value used by the KEM algorithm to produce the KEM shared secret, after constructing the entry in the KEMRecipientInfo structure for the corresponding recipient. Similarly, the recipient **MUST** discard the KEM shared secret, including any random value used by the KEM algorithm to produce the KEM shared secret, after constructing the KEK from the KEMRecipientInfo structure.

Implementations **MUST** randomly generate content-encryption keys, content-authenticated-encryption keys, and message-authentication keys. Also, the generation of KEM key pairs relies on random numbers. The use of inadequate pseudorandom number generators (PRNGs) to generate these 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.

If the cipher and key sizes used for the key-encryption algorithm and the content-encryption algorithm are different, the effective security is determined by the weaker of the two algorithms. If, for example, the content is encrypted with AES-CBC using a 128-bit CEK and the CEK is wrapped with AES-KEYWRAP using a 256-bit KEK, then at most 128 bits of protection is provided.

If the cipher and key sizes used for the key-encryption algorithm and the content-authenticated-encryption algorithm are different, the effective security is determined by the weaker of the two algorithms. If, for example, the content is encrypted with AES-GCM using a 128-bit CAEK and the CAEK is wrapped with AES-KEYWRAP using a 192-bit KEK, then at most 128 bits of protection is provided.

If the cipher and key sizes used for the key-encryption algorithm and the message-authentication algorithm are different, the effective security is determined by the weaker of the two algorithms. If, for example, the content is authenticated with HMAC-SHA256 using a 512-bit message-authentication key and the message-authentication key is wrapped with AES-KEYWRAP using a 256-bit KEK, then at most 256 bits of protection is provided.

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

8. IANA Considerations

For KEMRecipientInfo as defined in Section 3, IANA has assigned the following OID in the "SMI Security for S/MIME Other Recipient Info Identifiers (1.2.840.113549.1.9.16.13)" registry:

Decimal	Description	References
3	id-ori-kem	RFC 9629

Table 1

For the ASN.1 module defined in Section 6.1, IANA has assigned the following OID in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0):

Decimal	Description	References
109	id-mod-kemAlgorithmInformation-2023	RFC 9629

Table 2

For the ASN.1 module defined in Section 6.2, IANA has assigned the following OID in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

Decimal	Description	References
77	id-mod-cms-kemri-2023	RFC 9629

Table 3

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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[X.680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

[X.690] 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, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

9.2. Informative References

[FO] Fujisaki, E. and T. Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", Springer Science and Business Media LLC, Journal of Cryptology, vol. 26, no. 1, pp. 80-101, DOI 10.1007/s00145-011-9114-1, December 2011, <https://doi.org/10.1007/s00145-011-9114-1>.

[HHK] Hofheinz, D., Hövelmanns, K., and E. Kiltz, "A Modular Analysis of the Fujisaki-Okamoto Transformation", Springer International Publishing, Theory of Cryptography, TCC 2017, Lecture Notes in Computer Science, vol. 10677, pp. 341-371, Print ISBN 9783319704999, Online ISBN 9783319705002, DOI 10.1007/978-3-319-70500-2_12, November 2017, <https://doi.org/10.1007/978-3-319-70500-2_12>.

[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>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[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>.

Acknowledgements

Our thanks to Burt Kaliski for his guidance and design review.

Our thanks to Carl Wallace for his careful review of the ASN.1 modules.

Our thanks to Hendrik Brockhaus, Jonathan Hammell, Mike Jenkins, David von Oheimb, Francois Rousseau, and Linda Dunbar for their careful reviews and thoughtful comments.

Authors' Addresses

Russ Housley
Vigil Security, LLC
Herndon, VA
United States of America
Email: housley@vigilsec.com

John Gray
Entrust
Minneapolis, MN
United States of America
Email: john.gray@entrust.com

Tomofumi Okubo
Penguin Securities Pte. Ltd.
Singapore
Email: tomofumi.okubo+ietf@gmail.com

Additional contact information:

大久保　智史
Penguin Securities Pte. Ltd.
Singapore