Workgroup: LAMPS Working Group
Internet-Draft: draft-ietf-lamps-cms-kemri-00
Published: 24 February 2023
Intended Status: Standards Track
Expires: 28 August 2023
Authors: R. Housley (Vigil Security), J. Gray (Entrust), T. Okubo (DigiCert)

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 CMS content.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME Working Group mailing list (spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/lamps-wg/cms-kemri.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Cryptographic Message Syntax (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 the content. In recent years, cryptographers have be 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. The recipient's private key is needed to recover the random keying material, which is then treated as a pairwise shared secret between the originator and recipient. 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 Encapsulate().

To support a particular KEM algorithm, the CMS recipient **MUST** implement KeyGen() and Decapsulate(). 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

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

This is a reasonable approach; it places error processing outside of the fast path. This approach is also forgiving when an incorrect version number is used by the sender.

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

2. 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. Thus, the content-encryption key is used to protect the in CMS content.

The originator randomly generates the content-encryption key, and then all recipients obtain that key. 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 key-encryption key, which is used to encrypt the originator-generated content-encryption key.

In advance, each recipient recipient uses KeyGen() to create a key pair, and then obtains a certificate [RFC5280] that includes the public key.

The originator establishes the content-encryption key using these steps:

1. The content-encryption key, called CEK, is generated at random.

2. For each recipient:

    *The recipient's public key is used with the Encapsulate() function to obtain a pairwise shared secret and the ciphertext for the recipient.

    *The key-derivation function is used to derive a pairwise key-encryption key, called KEK, from the pairwise shared secret and other data that is send in the clear.

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

The recipient obtains the content-encryption key using these steps:

1. The recipient's private key and the ciphertext are used with the Decapsulate() function to obtain a pairwise shared secret.

2. The key-derivation function is used to derive a pairwise key-encryption key, called KEK, from the pairwise shared secret and other data that is send in the clear.

3. The KEK is used to decrypt the content-encryption key, called CEK.

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 are represented in the OtherRecipientInfo type, and they are each 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 TBD1 }
```

The KEMRecipientInfo type is:

```
KEMRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- always set to 0
  rid RecipientIdentifier,
  kem KEMAlgorithmIdentifier,
  kemct OCTET STRING,
  kdf KeyDerivationAlgorithmIdentifier,
  kekLength INTEGER (1..MAX),
  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 to with the 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 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 Encapsulate() function for this recipient.

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

kekLength is the size of the key-encryption key in octets. This value is one of the inputs to the key-derivation function. Implementations **MUST** confirm that the value provided is consistent with the key-encryption algorithm identified in the wrap field below.

ukm is optional. When the ukm value is provided, it is used as an input to the key-derivation function as a context input. For example, user key material could include a nonce, an IV, or other data required by the key-derivation function. Implementations **MUST** accept a KEMRecipientInfo SEQUENCE that includes a ukm field. Note that this expands of 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 content-encryption key. The KeyEncryptionAlgorithmIdentifier is described in Section 10.1.3 of [RFC5652].

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

4. KEM Algorithm Identifier

The KEMAlgorithmIdentifier type identifies a KEM algorithm used to establish a pairwise shared secret. The details of establishment depend on the KEM algorithm used. A Key derivation algorithm is used to transform the pairwise shared secret value into a key-encryption key.

```
  KEMAlgorithmIdentifier ::= AlgorithmIdentifier
```

5. Key Derivation

This section describes the conventions of using the KDF to compute the key-encryption key 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:

IKM is the input key material. It is a symmetric secret input to the KDF which 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 which is identified in the keklength of the KEMRecipientInfo. The value is dependent on the key-encryption algorithm that is used which is identified in the KeyEncryptionAlgorithmIdentifier.

info is the context used as in additional input to the KDF; it is the DER-encoded CMSORIforKEMOtherInfo structure defined as:

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

The CMSORIforKEMOtherInfo structure contains:

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 key-encryption key in octets; the output of the key-derivation function will be exactly this size.

ukm is optional user keying material, which may be useful for some key-derivation functions. For example, user keying material could include a nonce, an IV, or additional key binding information.

The KDF output is:

OKM is the output keying material with the exact length of L octets. The OKM is the key-encryption key that is used to encrypt the content-encryption key or the content-authenticated-encryption key.

An acceptable KDF **MUST** accept an IKM and L as inputs; an acceptable KDF **MAY** also accept salt and other inputs identified in the UserKeyingMaterial. All of these inputs **MUST** influence the output of the KDF. If the KDF requires a salt or other inputs, then those input **MUST** be provided as parameters of the KeyDerivationAlgorithmIdentifier.

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 in [RFC5912], and 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(TBD3) }

  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 prefixes 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 ASN.1 Module

RFC Editor: Please replace "[THISRFC]" with the the number assigned to this document.

```
<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(TBD2) }

  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN
  -- EXPORTS ALL;
  IMPORTS
    OTHER-RECIPIENT, CMSVersion, RecipientIdentifier,
    EncryptedKey, KeyDerivationAlgorithmIdentifier,
    KeyEncryptionAlgorithmIdentifier, UserKeyingMaterial
      FROM CryptographicMessageSyntax-2010  -- [RFC6268]
        { 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  -- [THISRFC]
        { iso(1) identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-kemAlgorithmInformation-2023(TBD3) }
    AlgorithmIdentifier{}
      FROM AlgorithmInformation-2009  -- [RFC5912]
        { iso(1) identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-algorithmInformation-02(58) } ;

  --
  -- 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 TBD1 }

  --
  -- KEMRecipientInfo
  --

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

  END

<CODE ENDS>
```

7. Security Considerations

The Security Considerations of [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 choice of the KDF **SHOULD** be made based on the security level provided by the KEM. The KDF **SHOULD** at least have the security level of the KEM.

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

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

Implementations **MUST** protect the KEM private key, the key-encryption key, the content-encryption key and the content-authenticated-encryption. Compromise of the KEM private key may result in the disclosure of all contents protected with that KEM private key. However, compromise of the key-encryption key, the content-encryption key, or the content-authenticated-encryption 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 to 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 key-encryption key 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 pseudo-random 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 and the content-encryption algorithms 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 content-encryption key, and the content-encryption key is wrapped with AES-KEYWRAP using a 256-bit key-encryption key, then at most 128 bits of protection is provided.

If the cipher and key sizes used for the key-encryption and the content-authenticated-encryption algorithms 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 content-authenticated-encryption key, and the content-authenticated-encryption key is wrapped with AES-KEYWRAP using a 192-bit key-encryption key, then at most 128 bits of protection is provided.

If the cipher and key sizes used for the key-encryption and the message-authentication algorithms 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 key-encryption key, 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 in Section 3, IANA is requested to assign an object identifier (OID) to replace TBD1. The OID for KEMRecipientInfo should be allocated in the "SMI Security for S/MIME Other Recipient Info Identifiers" registry (1.2.840.113549.1.9.16.13).

For the ASN.1 Module in Section 6.1, IANA is requested to assign an object identifier (OID) for the module identifier to replace TBD3. The OID for the module should be allocated in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).

For the ASN.1 Module in Section 6.2, IANA is requested to assign an object identifier (OID) for the module identifier to replace TBD2. The OID for the module should be allocated in the "SMI Security for S/MIME Module Identifier" registry (1.2.840.113549.1.9.16.0).

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.

References

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/rfc/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/rfc/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/rfc/rfc5280>.

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

[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/rfc/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.680>.

Informative References

[FO] Fujisaki, E. and T. Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", 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", Theory of Cryptography pp. 341-371, DOI 10.1007/978-3-319-70500-2_12, 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/rfc/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/rfc/rfc5869>.

[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/rfc/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/rfc/rfc5912>.

[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/rfc/rfc6268>.

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
DigiCert, Inc.
Fairfax, VA,
United States of America

Email: tomofumi.okubo+ietf@gmail.com