Privacy Enhancement for Internet Electronic Mail:
Part III: Algorithms, Modes, and Identifiers

D. Balenson

February 1993

Status of this Memo

This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document provides definitions, formats, references, and citations for cryptographic algorithms, usage modes, and associated identifiers and parameters used in support of Privacy Enhanced Mail (PEM) in the Internet community. It is intended to become one member of the set of related PEM RFCs. This document is organized into four primary sections, dealing with message encryption algorithms, message integrity check algorithms, symmetric key management algorithms, and asymmetric key management algorithms (including both asymmetric encryption and asymmetric signature algorithms).

Some parts of this material are cited by other documents and it is anticipated that some of the material herein may be changed, added, or replaced without affecting the citing documents. Therefore, algorithm-specific material has been placed into this separate document.

Use of other algorithms and/or modes will require case-by-case study to determine applicability and constraints. The use of additional algorithms may be documented first in Prototype or Experimental RFCs. As experience is gained, these protocols may be considered for incorporation into the standard. Additional algorithms and modes approved for use in PEM in this context will be specified in successors to this document.

Acknowledgments

This specification was initially developed by the Internet Research Task Force's Privacy and Security Research Group (IRTF PSRG) and subsequently refined based on discussion in the Internet Engineering Task Force's Privacy Enhanced Mail Working Group (IETF PEM WG). John Linn contributed significantly to the predecessor of this document (RFC 1115). I would like to thank the members of the PSRG and PEM WG, as well as all participants in discussions on the "pem-dev@tis.com" mailing list, for their contributions to this document.

1. Message Encryption Algorithms

This section identifies the alternative message encryption algorithms and modes that shall be used to encrypt message text and, when asymmetric key management is employed in an ENCRYPTED PEM message, for encryption of message signatures. Character string identifiers are assigned and any parameters required by the message encryption algorithm are defined for incorporation in an encapsulated "DEK- Info:" header field.

Only one alternative is currently defined in this category.

1.1 DES in CBC Mode (DES-CBC)

Message text and, if required, message signatures are encrypted using the Data Encryption Standard (DES) algorithm in the Cipher Block Chaining (CBC) mode of operation. The DES algorithm is defined in FIPS PUB 46-1 [1], and is equivalent to the Data Encryption Algorithm (DEA) provided in ANSI X3.92-1981 [2]. The CBC mode of operation ofDES is defined in FIPS PUB 81 [3], and is equivalent to those provided in ANSI X3.106 [4] and in ISO IS 8372 [5]. The character string "DES-CBC" within an encapsulated PEM header field indicates the use of this algorithm/mode combination.

The input to the DES CBC encryption process shall be padded to a multiple of 8 octets, in the following manner. Let n be the length in octets of the input. Pad the input by appending 8-(n mod 8) octets to the end of the message, each having the value 8-(n mod 8), the number of octets being added. In hexadecimal, the possible paddings are: 01, 0202, 030303, 04040404, 0505050505, 060606060606, 07070707070707, and 0808080808080808. All input is padded with 1 to 8 octets to produce a multiple of 8 octets in length. The padding can be removed unambiguously after decryption.

The DES CBC encryption process requires a 64-bit cryptographic key. A new, pseudorandom key shall be generated for each ENCRYPTED PEM message. Of the 64 bits, 56 are used directly by the DES CBC process, and 8 are odd parity bits, with one parity bit occupying the right-most bit of each octet. When symmetric key management is employed, the setting and checking of odd parity bits is encouraged, since these bits could detect an error in the decryption of a DES key encrypted under a symmetric key management algorithm (e.g., DES ECB). When asymmetric key management is employed, the setting of odd parity bits is encouraged, but the checking of odd parity bits is discouraged, in order to facilitate interoperability, and since an error in the decryption of a DES key can be detected by other means (e.g., an incorrect PKCS #1 encryption-block format). In all cases, the encrypted form of a DES key shall carry all 64 bits of the key, including the 8 parity bits, though those bits may have no meaning.

The DES CBC encryption process also requires a 64-bit Initialization Vector (IV). A new, pseudorandom IV shall be generated for each ENCRYPTED PEM message. Section 4.3.1 of [7] provides rationale for this requirement, even given the fact that individual DES keys are generated for individual messages. The IV is transmitted with the message within an encapsulated PEM header field.

When this algorithm/mode combination is used for message text encryption, the "DEK-Info:" header field carries exactly two arguments. The first argument identifies the DES CBC algorithm/mode using the character string defined above. The second argument contains the IV, represented as a contiguous string of 16 ASCII hexadecimal digits.

When symmetric key management is employed with this algorithm/mode combination, a symmetrically encrypted DES key will be represented in the third argument of a "Key-Info:" header field as a contiguous string of 16 ASCII hexadecimal digits (corresponding to a 64-bit key).

To avoid any potential ambiguity regarding the ordering of the octets of a DES key that is input as a data value to another encryption process (e.g., RSAEncryption), the following holds true. The first (or left-most displayed, if one thinks in terms of a key's "print" representation) (For purposes of discussion in this document, data values are normalized in terms of their "print" representation. For a octet stream, the "first" octet would appear as the one on the "left", and the "last" octet would appear on the "right".) octet of the key (i.e., bits 1-8 per FIPS PUB 46-1), when considered as a data value, has numerical weight 2**56. The last (or right-most displayed) octet (i.e., bits 57-64 per FIPS PUB 46-1) has numerical weight 2**0.

2. Message Integrity Check Algorithms

This section identifies the alternative algorithms that shall be used to compute Message Integrity Check (MIC) values for PEM messages. Character string identifiers and ASN.1 object identifiers are assigned for incorporation in encapsulated "MIC-Info:" and "Key- Info:" header fields to indicate the choice of MIC algorithm employed.

A compliant PEM implementation shall be able to process all of the alternative MIC algorithms defined here on incoming messages. It is a sender option as to which alternative is employed on an outbound message.

2.1 RSA-MD2 Message Digest Algorithm

The RSA-MD2 message digest is computed using the algorithm defined in RFC 1319 [9]. (An error has been identified in RFC 1319. The statement in the text of Section 3.2 which reads "Set C[j] to S[c xor L]" should read "Set C[j] to S[c xor L] xor C[j]". Note that the C source code in the appendix of RFC 1319 is correct.) The character string "RSA-MD2" within an encapsulated PEM header field indicates the use of this algorithm. Also, as defined in RFC 1319, the ASN.1 object identifier

```
md2 OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549)
        digestAlgorithm(2) 2
 }
```

identifies this algorithm. When this object identifier is used with the ASN.1 type AlgorithmIdentifier, the parameters component of that type is the ASN.1 type NULL.

The RSA-MD2 message digest algorithm accepts as input a message of any length and produces as output a 16-octet quantity. When symmetric key management is employed, an RSA-MD2 MIC is encrypted by splitting the MIC into two 8-octet halves, independently encrypting each half, and concatenating the results.

When symmetric key management is employed with this MIC algorithm, the symmetrically encrypted MD2 message digest is represented in a the fourth argument of a "Key-Info:" header field as a contiguous string of 32 ASCII hexadecimal digits (corresponding to a 128-bit MD2 message digest).

To avoid any potential ambiguity regarding the ordering of the octets of an MD2 message digest that is input as a data value to another encryption process (e.g., RSAEncryption), the following holds true. The first (or left-most displayed, if one thinks in terms of a digest's "print" representation) octet of the digest (i.e., digest[0] as specified in RFC 1319), when considered as an RSA data value, has numerical weight 2**120. The last (or right-most displayed) octet (i.e., digest[15] as specified in RFC 1319) has numerical weight 2**0.

2.2 RSA-MD5 Message Digest Algorithm

The RSA-MD5 message digest is computed using the algorithm defined in RFC 1321 [10]. The character string "RSA-MD5" within an encapsulated PEM header field indicates the use of this algorithm. Also, as defined in RFC 1321, the object identifier

```
md5 OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549)
        digestAlgorithm(2) 5
 }
```

identifies this algorithm. When this object identifier is used with the ASN.1 type AlgorithmIdentifier, the parameters component of that type is the ASN.1 type NULL.

The RSA-MD5 message digest algorithm accepts as input a message of any length and produces as output a 16-octet quantity. When symmetric key management is employed, an RSA-MD5 MIC is encrypted by splitting the MIC into two 8-octet halves, independently encrypting each half, and concatenating the results.

When symmetric key management is employed with this MIC algorithm, the symmetrically encrypted MD5 message digest is represented in the fourth argument of a "Key-Info:" header field as a contiguous string of 32 ASCII hexadecimal digits (corresponding to a 128-bit MD5 message digest).

To avoid any potential ambiguity regarding the ordering of the octets of a MD5 message digest that is input as an RSA data value to the RSA encryption process, the following holds true. The first (or left- most displayed, if one thinks in terms of a digest's "print" representation) octet of the digest (i.e., the low-order octet of A as specified in RFC 1321), when considered as an RSA data value, has numerical weight 2**120. The last (or right-most displayed) octet (i.e., the high-order octet of D as specified in RFC 1321) has numerical weight 2**0.

3. Symmetric Key Management Algorithms

This section identifies the alternative algorithms and modes that shall be used when symmetric key management is employed, to encrypt data encryption keys (DEKs) and message integrity check (MIC) values. Character string identifiers are assigned for incorporation in encapsulated "Key-Info:" header fields to indicate the choice of algorithm employed.

All alternatives presently defined in this category correspond to different usage modes of the DES algorithm, rather than to other algorithms.

When symmetric key management is employed, the symmetrically encrypted DEK and MIC, carried in the third and fourth arguments of a "Key-Info:" header field, respectively, are each represented as a string of contiguous ASCII hexadecimal digits. The manner in which to use the following symmetric encryption algorithms and the length of the symmetrically encrypted DEK and MIC may vary depending on the length of the underlying DEK and MIC. Section 1, Message Encryption Algorithms, and Section 2, Message Integrity Check Algorithms, provide information on the proper manner in which a DEK and MIC, respectively, are symmetrically encrypted when the size of the DEK or MIC is not equal to the symmetric encryption algorithm's input block size. These sections also provide information on the proper format and length of the symmetrically encrypted DEK and MIC, respectively.

3.1 DES in ECB Mode (DES-ECB)

The DES algorithm in Electronic Codebook (ECB) mode [1][3] is used for DEK and MIC encryption when symmetric key management is employed. The character string "DES-ECB" within an encapsulated PEM header field indicates use of this algorithm/mode combination.

A compliant PEM implementation supporting symmetric key management shall support this algorithm/mode combination.

3.2 DES in EDE Mode (DES-EDE)

The DES algorithm in Encrypt-Decrypt-Encrypt (EDE) multiple encryption mode, as defined by ANSI X9.17 [6] for encryption and decryption with pairs of 64-bit keys, may be used for DEK and MIC encryption when symmetric key management is employed. The character string "DES-EDE" within an encapsulated a PEM header field indicates use of this algorithm/mode combination.

A compliant PEM implementation supporting symmetric key management may optionally support this algorithm/mode combination.

4. Asymmetric Key Management Algorithms

This section identifies the alternative asymmetric keys and the alternative asymmetric key management algorithms with which those keys shall be used, namely the asymmetric encryption algorithms with which DEKs and MICs are encrypted, and the asymmetric signature algorithms with which certificates and certificate revocation lists (CRLs) are signed.

4.1 Asymmetric Keys

This section describes the asymmetric keys that shall be used with the asymmetric encryption algorithms and the signature algorithms described later. ASN.1 object identifiers are identified for incorporation in a public-key certificate to identify the algorithm(s) with which the accompanying public key is to be employed.

4.1.1 RSA Keys

An RSA asymmetric key pair is comprised of matching public and private keys.

An RSA public key consists of an encryption exponent e and an arithmetic modulus n, which are both public quantities typically carried in a public-key certificate. For the value of e, Annex C to X.509 suggests the use of Fermat's Number F4 (65537 decimal, or 1+2**16) as a value "common to the whole environment in order to reduce transmission capacity and complexity of transformation", i.e., the value can be transmitted as 3 octets and at most seventeen (17) multiplications are required to effect exponentiation. As an alternative, the number three (3) can be employed as the value for e, requiring even less octets for transmission and yielding even faster exponentiation. For purposes of PEM, the value of e shall be either F4 or the number three (3). The use of the number three (3) for the value of e is encouraged, to permit rapid certificate validation.

An RSA private key consists of a decryption exponent d, which should be kept secret, and the arithmetic modulus n. Other values may be stored with a private key to facilitate efficient private key operations (see PKCS #1 [11]).

For purposes of PEM, the modulus n may vary in size from 508 to 1024 bits.

Two ASN.1 object identifiers have been defined to identify RSA public keys. In Annex H of X.509 [8], the object identifier

```
rsa OBJECT IDENTIFIER ::= {
        joint-iso-ccitt(2) ds(5) algorithm(8)
        encryptionAlgorithm(1) 1
  }
```

is defined to identify an RSA public key. A single parameter, KeySize, the length of the public key modulus in bits, is defined for use in conjunction with this object identifier. When this object identifier is used with the ASN.1 type AlgorithmIdentifier, the parameters component of that type is the number of bits in the modulus, ASN.1 encoded as an INTEGER.

Alternatively, in PKCS #1 [11], the ASN.1 object identifier

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

is defined to identify both an RSA public key and the RSAEncryption process. There are no parameters defined in conjunction with this object identifier, hence, when it is used with the ASN.1 type AlgorithmIdentifier, the parameters component of that type is the ASN.1 type NULL.

A compliant PEM implementation may optionally generate an RSA public-key certificate that identifies the enclosed RSA public key (within the SubjectPublicKeyInformation component) with either the "rsa" or the "rsaEncryption" object identifier. Use of the "rsa" object identifier is encouraged, since it is, in some sense, more generic in its identification of a key, without indicating how the key will be used. However, to facilitate interoperability, a compliant PEM implementation shall accept RSA public-key certificates that identify the enclosed RSA public key with either the "rsa" or the "rsaEncryption" object identifier. In all cases, an RSA public key identified in an RSA public-key certificate with either the "rsa" or "rsaEncryption" object identifier, shall be used according to the procedures defined below for asymmetric encryption algorithms and asymmetric signature algorithms.

4.2 Asymmetric Encryption Algorithms

This section identifies the alternative algorithms that shall be used when asymmetric key management is employed, to encrypt DEKs and MICs. Character string identifiers are assigned for incorporation in "MIC- Info:" and "Key-Info:" header fields to indicate the choice of algorithm employed.

Only one alternative is presently defined in this category.

4.2.1 RSAEncryption

The RSAEncryption public-key encryption algorithm, defined in PKCS #1 [11], is used for DEK and MIC encryption when asymmetric key management is employed. The character string "RSA" within a "MIC- Info:" or "Key-Info:" header field indicates the use of this algorithm.

All PEM implementations supporting asymmetric key management shall support this algorithm.

As described in PKCS #1, all quantities input as data values to the RSAEncryption process shall be properly justified and padded to the length of the modulus prior to the encryption process. In general, an RSAEncryption input value is formed by concatenating a leading NULL octet, a block type BT, a padding string PS, a NULL octet, and the data quantity D, that is,

RSA input value = 0x00 —— BT —— PS —— 0x00 —— D.

To prepare a DEK for RSAEncryption, the PKCS #1 "block type 02" encryption-block formatting scheme is employed. The block type BT is a single octet containing the value 0x02 and the padding string PS is one or more octets (enough octets to make the length of the complete RSA input value equal to the length of the modulus) each containing a pseudorandomly generated, non-zero value. For multiple recipient messages, a different, pseudorandom padding string should be used for each recipient. The data quantity D is the DEK itself, which is right-justified within the RSA input such that the last (or rightmost displayed, if one thinks in terms of the "print" representation) octet of the DEK is aligned with the right-most, or least- significant, octet of the RSA input. Proceeding to the left, each of the remaining octets of the DEK, up through the first (or left-most displayed) octet, are each aligned in the next more significant octet of the RSA input.

To prepare a MIC for RSAEncryption, the PKCS #1 "block type 01" encryption-block formatting scheme is employed. The block type BT is a single octet containing the value 0x01 and the padding string PS is one or more octets (enough octets to make the length of the complete RSA input value equal to the length of the modulus) each containing the value 0xFF. The data quantity D is comprised of the MIC and the MIC algorithm identifier which are ASN.1 encoded as the following sequence.

```
SEQUENCE {
        digestAlgorithm     AlgorithmIdentifier,
        digest              OCTET STRING
 }
```

The ASN.1 type *AlgorithmIdentifier* is defined in X.509 as follows.

```
AlgorithmIdentifier ::=     SEQUENCE {
        algorithm           OBJECT IDENTIFIER,
        parameters          ANY DEFINED BY algorithm OPTIONAL
 }
```

An RSA input block is encrypted using the RSA algorithm with the first (or left-most) octet taken as the most significant octet, and the last (or right-most) octet taken as the least significant octet. The resulting RSA output block is interpreted in a similar manner.

When RSAEncryption is used to encrypt a DEK, the second argument in a "MIC-Info:" header field, an asymmetrically encrypted DEK, is represented using the printable encoding technique defined in Section 4.3.2.4 of RFC 1421 [12].

When RSAEncryption is used to sign a MIC, the third argument in a "MIC-Info:" header field, an asymmetrically signed MIC, is represented using the printable encoding technique defined in Section 4.3.2.4 of RFC 1421.

4.3 Asymmetric Signature Algorithms

This section identifies the alternative algorithms which shall be used to asymmetrically sign certificates and certificate revocation lists (CRLs) in accordance with the SIGNED macro defined in Annex G of X.509. ASN.1 object identifiers are identified for incorporation in certificates and CRLs to indicate the choice of algorithm employed.

Only one alternative is presently defined in this category.

4.3.1 md2WithRSAEncryption

The md2WithRSAEncryption signature algorithm is used to sign certificates and CRLs. The algorithm is defined in PKCS #1 [11]. It combines the RSA-MD2 message digest algorithm described here in Section 2.2 with the RSAEncryption asymmetric encryption algorithm described here in Section 4.2.1. As defined in PKCS #1, the ASN.1 object identifier

```
md2WithRSAEncryption OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1)
        pkcs-1(1) 2
 }
```

identifies this algorithm. When this object identifier is used with the ASN.1 type AlgorithmIdentifier, the parameters component of that type is the ASN.1 type NULL.

There is some ambiguity in X.509 regarding the definition of the SIGNED macro and, in particular, the representation of a signature in a certificate or a CRL. The interpretation selected for PEM requires that the data to be signed (in our case, an MD2 message digest) is first ASN.1 encoded as an OCTET STRING and the result is encrypted (in our case, using RSAEncryption) to form the signed quantity, which is then ASN.1 encoded as a BIT STRING.

5. Descriptive Grammar

; Addendum to PEM BNF representation, using RFC 822 notation

; Provides specification for official PEM cryptographic algorithms, modes, identifiers and formats.

; Imports <hexchar> and <encbin> from RFC [1113ID]

<dekalgid> ::= "DES-CBC"
<ikalgid> ::= "DES-EDE" | "DES-ECB" | "RSA"
<sigalgid> ::= "RSA"
<micalgid> ::= "RSA-MD2" | "RSA-MD5"

<dekparameters> ::= <DESCBCparameters>
<DESCBCparameters> ::= <IV>
<IV> ::= <hexchar16>

<symencdek> ::= <DESECBencDESCBC> | <DESEDEencDESCBC>
<DESECBencDESCBC> ::= <hexchar16>
<DESEDEencDESCBC> ::= <hexchar16>

<symencmic> ::= <DESECBencRSAMD2> | <DESECBencRSAMD5>

<DESECBencRSAMD2> ::= 2*2<hexchar16>
<DESECBencRSAMD5> ::= 2*2<hexchar16>

<asymsignmic> ::= <RSAsignmic>
<RSAsignmic> ::= <encbin>

<asymencdek> ::= <RSAencdek>
<RSAencdek> ::= <encbin>

<hexchar16> ::= 16*16<hexchar>

References

[1] Federal Information Processing Standards Publication (FIPS PUB) 46-1, Data Encryption Standard, Reaffirmed 1988 January 22 (supersedes FIPS PUB 46, 1977 January 15).

[2] ANSI X3.92-1981, American National Standard Data Encryption Algorithm, USA National Standards Institute, Approved: Dec 1980.

[3] Federal Information Processing Standards Publication (FIPS PUB) 81, DES Modes of Operation, 1980 December 2.

[4] ANSI X3.106-1983, American National Standard for Information Systems - Data Encryption Algorithm - Modes of Operation, American National Standards Institute, Approved 16 May 1983.

[5] ISO 8372, Information Processing Systems: Data Encipherment: Modes of Operation of a 64-bit Block Cipher.

[6] ANSI X9.17-1985, American National Standard, Financial Institution Key Management (Wholesale), American Bankers Association, April 4, 1985, Section 7.2.

[7] Voydock, V. L. and Kent, S. T., "Security Mechanisms in High- Level Network Protocols", ACM Computing Surveys, Vol. 15, No. 2, June 1983, pp. 135-171.

[8] CCITT Recommendation X.509, "The Directory - Authentication Framework", November 1988, (Developed in collaboration, and technically aligned, with ISO 9594-8).

[9] Kaliski, B., "The MD2 Message-Digest Algorithm", RFC 1319, RSA Laboratories, April 1992.

[10] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[11] PKCS #1: RSA Encryption Standard, Version 1.4, RSA Data Security, Inc., June 3, 1991.

[12] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, DEC, February 1993.

[13] Kent, S., "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422, BBN, February 1993.

[14] Kaliski, B., "Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services", RFC 1424, RSA Laboratories, February 1993.

Patent Statement

This version of Privacy Enhanced Mail (PEM) relies on the use of patented public key encryption technology for authentication and encryption. The Internet Standards Process as defined in RFC 1310 requires a written statement from the Patent holder that a license will be made available to applicants under reasonable terms and conditions prior to approving a specification as a Proposed, Draft or Internet Standard.

The Massachusetts Institute of Technology and the Board of Trustees of the Leland Stanford Junior University have granted Public Key Partners (PKP) exclusive sub-licensing rights to the following patents issued in the United States, and all of their corresponding foreign patents:

Cryptographic Apparatus and Method
("Diffie-Hellman")................................... No. 4,200,770

Public Key Cryptographic Apparatus
and Method ("Hellman-Merkle").................... No. 4,218,582

Cryptographic Communications System and
Method ("RSA")....................................... No. 4,405,829

Exponential Cryptographic Apparatus
and Method ("Hellman-Pohlig").................... No. 4,424,414

These patents are stated by PKP to cover all known methods of practicing the art of Public Key encryption, including the variations collectively known as El Gamal.

Public Key Partners has provided written assurance to the Internet Society that parties will be able to obtain, under reasonable, nondiscriminatory terms, the right to use the technology covered by these patents. This assurance is documented in RFC 1170 titled "Public Key Standards and Licenses". A copy of the written assurance dated April 20, 1990, may be obtained from the Internet Assigned Number Authority (IANA).

The Internet Society, Internet Architecture Board, Internet Engineering Steering Group and the Corporation for National Research Initiatives take no position on the validity or scope of the patents and patent applications, nor on the appropriateness of the terms of the assurance. The Internet Society and other groups mentioned above have not made any determination as to any other intellectual property rights which may apply to the practice of this standard. Any further consideration of these matters is the user's own responsibility.

Security Considerations

This entire document is about security.

Author's Address

David Balenson
Trusted Information Systems
3060 Washington Road
Glenwood, Maryland 21738

Phone: 301-854-6889
EMail: balenson@tis.com