Network Working Group J. Linn
Request for Comments: 1115 DEC
IAB Privacy Task Force
August 1989

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

STATUS OF THIS MEMO

This RFC suggests a draft standard elective protocol for the Internet community, and requests discussion and suggestions for improvement. This RFC provides definitions, references, and citations for algorithms, usage modes, and associated identifiers used in RFC-1113 and RFC-1114 in support of privacy-enhanced electronic mail. Distribution of this memo is unlimited.

ACKNOWLEDGMENT

This RFC is the outgrowth of a series of IAB Privacy Task Force meetings and of internal working papers distributed for those meetings. I would like to thank the following Privacy Task Force members and meeting guests for their comments and contributions at the meetings which led to the preparation of this RFC: David Balenson, Curt Barker, Jim Bidzos, Matt Bishop, Morrie Gasser, Russ Housley, Steve Kent (chairman), Dan Nessett, Mike Padlipsky, Rob Shirey, and Steve Wilbur.

Table of Contents

1. Executive Summary 2
2. Symmetric Encryption Algorithms and Modes 2
2.1. DES Modes 2
2.1.1. DES in ECB mode (DES-ECB) 2
2.1.2. DES in EDE mode (DES-EDE) 2
2.1.3. DES in CBC mode (DES-CBC) 3
3. Asymmetric Encryption Algorithms and Modes 3
3.1. RSA 3
4. Integrity Check Algorithms 3
4.1. Message Authentication Code (MAC) 4
4.2. RSA-MD2 Message Digest Algorithm 4
4.2.1. Discussion 4
4.2.2. Reference Implementation 5
NOTES 7

1. Executive Summary

This RFC provides definitions, references, and citations for algorithms, usage modes, and associated identifiers used in RFC-1113 and RFC-1114 in support of privacy-enhanced electronic mail in the Internet community. As some parts of this material are cited by both RFC-1113 and RFC-1114, and as it is anticipated that some of the definitions herein may be changed, added, or replaced without affecting the citing RFCs, algorithm-specific material has been placed into this separate RFC. The text is organized into three primary sections; dealing with symmetric encryption algorithms, asymmetric encryption algorithms, and integrity check algorithms.

2. Symmetric Encryption Algorithms and Modes

This section identifies alternative symmetric encryption algorithms and modes which may be used to encrypt DEKs, MICs, and message text, and assigns them character string identifiers to be incorporated in encapsulated header fields to indicate the choice of algorithm employed. (Note: all alternatives presently defined in this category correspond to different usage modes of the DEA-1 (DES) algorithm, rather than to other algorithms per se.)

2.1. DES Modes

The Block Cipher Algorithm DEA-1, defined in ANSI X3.92-1981 [3] may be used for message text, DEKs, and MICs. The DEA-1 is equivalent to the Data Encryption Standard (DES), as defined in FIPS PUB 46 [4]. The ECB and CBC modes of operation of DEA-1 are defined in ISO IS 8372 [5].

2.1.1. DES in ECB mode (DES-ECB)

The string "DES-ECB" indicates use of the DES algorithm in Electronic Codebook (ECB) mode. This algorithm/mode combination is used for DEK and MIC encryption.

2.1.2. DES in EDE mode (DES-EDE)

The string "DES-EDE" indicates use of the DES algorithm in Encrypt-Decrypt-Encrypt (EDE) mode as defined by ANSI X9.17 [2] for key encryption and decryption with pairs of 64-bit keys. This algorithm/mode combination is used for DEK and MIC encryption.

2.1.3. DES in CBC mode (DES-CBC)

The string "DES-CBC" indicates use of the DES algorithm in Cipher Block Chaining (CBC) mode. This algorithm/mode combination is used for message text encryption only. The CBC mode definition in IS 8372 is equivalent to that provided in FIPS PUB 81 [6] and in ANSI X3.106-1983 [7].

3. Asymmetric Encryption Algorithms and Modes

This section identifies alternative asymmetric encryption algorithms and modes which may be used to encrypt DEKs and MICs, and assigns them character string identifiers to be incorporated in encapsulated header fields to indicate the choice of algorithm employed. (Note: only one alternative is presently defined in this category.)

3.1. RSA

The string "RSA" indicates use of the RSA public-key encryption algorithm, as described in [8]. This algorithm is used for DEK and MIC encryption, in the following fashion: the product n of a individual's selected primes p and q is used as the modulus for the RSA encryption algorithm, comprising, for our purposes, the individual's public key. A recipient's public key is used in conjunction with an associated public exponent (either 3 or 1+2**16) as identified in the recipient's certificate.

When a MIC must be padded for RSA encryption, the MIC will be right-justified and padded on the left with zeroes. This is also appropriate for padding of DEKs on singly-addressed messages, and for padding of DEKs on multi-addressed messages if and only if an exponent of 3 is used for no more than one recipient. On multi-addressed messages in which an exponent of 3 is used for more than one recipient, it is recommended that a separate 64-bit pseudorandom quantity be generated for each recipient, in the same manner in which IVs are generated. (Reference [9] discusses the rationale for this recommendation.) At least one copy of the pseudorandom quantity should be included in the input to RSA encryption, placed to the left of the DEK.

4. Integrity Check Algorithms

This section identifies the alternative algorithms which may be used to compute Message Integrity Check (MIC) and Certificate Integrity Check (CIC) values, and assigns the algorithms character string identifiers for use in encapsulated header fields and within certificates to indicate the choice of algorithm employed.

MIC algorithms which utilize DEA-1 cryptography are computed using a key which is a variant of the DEK used for message text encryption. The variant is formed by modulo-2 addition of the hexadecimal quantity F0F0F0F0F0F0F0F0 to the encryption DEK.

For compatibility with this specification, a privacy-enhanced mail implementation must be able to process both MAC (Section 2.1) and RSA-MD2 (Section 2.2) MICs on incoming messages. It is a sender option whether MAC or RSA-MD2 is employed on an outbound message addressed to only one recipient. However, use of MAC is strongly discouraged for messages sent to more than a single recipient. The reason for this recommendation is that the use of MAC on multi-addressed mail fails to prevent other intended recipients from tampering with a message in a manner which preserves the message’s appearance as an authentic message from the sender. In other words, use of MAC on multi-addressed mail provides source authentication at the granularity of membership in the message’s authorized address list (plus the sender) rather than at a finer (and more desirable) granularity authenticating the individual sender.

4.1. Message Authentication Code (MAC)

A message authentication code (MAC), denoted by the string "MAC", is computed using the DEA-1 algorithm in the fashion defined in FIPS PUB 113 [1]. This algorithm is used only as a MIC algorithm, not as a CIC algorithm.

As noted above, use of the MAC is not recommended for multicast messages, as it does not preserve authentication and integrity among individual recipients, i.e., it is not cryptographically strong enough for this purpose. The message’s canonically encoded text is padded at the end, per FIPS PUB 113, with zero-valued octets as needed in order to form an integral number of 8-octet encryption quanta. These padding octets are inserted implicitly and are not transmitted with a message. The result of a MAC computation is a single 64-bit value.

4.2. RSA-MD2 Message Digest Algorithm

4.2.1. Discussion

The RSA-MD2 Message Digest Algorithm, denoted by the string "RSA-MD2", is computed using an algorithm defined in this section. It has been provided by Ron Rivest of RSA Data Security, Incorporated for use in support of privacy-enhanced electronic mail, free of licensing restrictions. This algorithm should be used as a MIC algorithm whenever a message is addressed to multiple recipients. It is also the only algorithm currently defined for use as CIC. While its continued use as the standard CIC algorithm is anticipated, RSA-MD2 may be supplanted by later recommendations for MIC algorithm selections.

The RSA-MD2 message digest algorithm accepts as input a message of any length and produces as output a 16-byte quantity. The attached reference implementation serves to define the algorithm; implementors may choose to develop optimizations suited to their operating environments.

4.2.2. Reference Implementation

```
/* RSA-MD2 Message Digest algorithm in C  */
/*  by Ronald L. Rivest 10/1/88  */

#include <stdio.h>

/**********************************************************************/
/* Message digest routines:                                           */
/* To form the message digest for a message M                         */
/*    (1) Initialize a context buffer md using MDINIT                 */
/*    (2) Call MDUPDATE on md and each character of M in turn         */
/*    (3) Call MDFINAL on md                                          */
/* The message digest is now in md->D[0...15]                         */
/**********************************************************************/
/* An MDCTX structure is a context buffer for a message digest        */
/*  computation; it holds the current "state" of a message digest     */
/*  computation                                                       */
struct MDCTX
{
   unsigned char  D[48];   /* buffer for forming digest in */
                           /* At the end, D[0...15] form the message */
                           /*  digest */
   unsigned char  C[16];   /* checksum register */
   unsigned char  i;       /* number of bytes handled, modulo 16 */
   unsigned char  L;       /* last checksum char saved */
};
/* The table S given below is a permutation of 0...255 constructed    */
/*  from the digits of pi.  It is a ‘‘random’’ nonlinear byte         */
/*  substitution operation.                                           */
int S[256] = {
        41, 46, 67,201,162,216,124,  1, 61, 54, 84,161,236,240,  6, 19,
        98,167,  5,243,192,199,115,140,152,147, 43,217,188, 76,130,202,
        30,155, 87, 60,253,212,224, 22,103, 66,111, 24,138, 23,229, 18,
       190, 78,196,214,218,158,222, 73,160,251,245,142,187, 47,238,122,
       169,104,121,145, 21,178,  7, 63,148,194, 16,137, 11, 34, 95, 33,
       128,127, 93,154, 90,144, 50, 39, 53, 62,204,231,191,247,151,  3,
       255, 25, 48,179, 72,165,181,209,215, 94,146, 42,172, 86,170,198,
        79,184, 56,210,150,164,125,182,118,252,107,226,156,116,  4,241,
        69,157,112, 89,100,113,135, 32,134, 91,207,101,230, 45,168,  2,
        27, 96, 37,173,174,176,185,246, 28, 70, 97,105, 52, 64,126, 15,
        85, 71,163, 35,221, 81,175, 58,195, 92,249,206,186,197,234, 38,
        44, 83, 13,110,133, 40,132,  9,211,223,205,244, 65,129, 77, 82,
       106,220, 55,200,108,193,171,250, 36,225,123,  8, 12,189,177, 74,
       120,136,149,139,227, 99,232,109,233,203,213,254, 59,  0, 29, 57,
       242,239,183, 14,102, 88,208,228,166,119,114,248,235,117, 75, 10,
        49, 68, 80,180,143,237, 31, 26,219,153,141, 51,159, 17,131, 20,
};
/*The routine MDINIT initializes the message digest context buffer md.*/
/* All fields are set to zero.                                        */
void MDINIT(md)
  struct MDCTX *md;
  { int i;
    for (i=0;i<16;i++) md->D[i] = md->C[i] = 0;
    md->i = 0;
    md->L = 0;
  }
/* The routine MDUPDATE updates the message digest context buffer to  */
/*  account for the presence of the character c in the message whose  */
/*  digest is being computed.  This routine will be called for each   */
/*   message byte in turn.                                            */
void MDUPDATE(md,c)
  struct MDCTX *md;
  unsigned char c;
  { register unsigned char i,j,t,*p;
    /**** Put i in a local register for efficiency ****/
       i = md->i;
    /**** Add new character to buffer ****/
       md->D[16+i] = c;
       md->D[32+i] = c ^ md->D[i];
    /**** Update checksum register C and value L ****/
       md->L = (md->C[i] ^= S[0xFF & (c ^ md->L)]);
    /**** Increment md->i by one modulo 16 ****/
       i = md->i = (i + 1) & 15;
    /**** Transform D if i=0 ****/
       if (i == 0)
         { t = 0;
           for (j=0;j<18;j++)
             {/*The following is a more efficient version of the loop:*/
               /*  for (i=0;i<48;i++) t = md->D[i] = md->D[i] ^ S[t]; */
               p = md->D;
               for (i=0;i<8;i++)
                 { t = (*p++ ^= S[t]);
                   t = (*p++ ^= S[t]);
                   t = (*p++ ^= S[t]);
                   t = (*p++ ^= S[t]);
                   t = (*p++ ^= S[t]);
                   t = (*p++ ^= S[t]);
                 }
               /* End of more efficient loop implementation */
               t = t + j;
             }
         }
  }
/* The routine MDFINAL terminates the message digest computation and  */
/* ends with the desired message digest being in md->D[0...15].       */
void MDFINAL(md)
  struct MDCTX *md;
  { int i,padlen;
    /* pad out to multiple of 16 */
       padlen  = 16 - (md->i);
       for (i=0;i<padlen;i++) MDUPDATE(md,(unsigned char)padlen);
    /* extend with checksum */
    /* Note that although md->C is modified by MDUPDATE, character    */
    /* md->C[i] is modified after it has been passed to MDUPDATE, so  */
    /* the net effect is the same as if md->C were not being modified.*/
    for (i=0;i<16;i++) MDUPDATE(md,md->C[i]);
  }

/**********************************************************************/
/* End of message digest implementation                               */
/**********************************************************************/
```

NOTES:

[1] Federal Information Processing Standards Publication 113, Computer Data Authentication, May 1985.

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

[3] American National Standard Data Encryption Algorithm (ANSI X3.92-1981), American National Standards Institute, Approved 30 December 1980.

[4] Federal Information Processing Standards Publication 46, Data Encryption Standard, 15 January 1977.

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

[6] Federal Information Processing Standards Publication 81, DES Modes of Operation, 2 December 1980.

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

[8] CCITT, Recommendation X.509, "The Directory: Authentication Framework", Annex C.

[9] Moore, J., "Protocol Failures in Cryptosystems", Proceedings of the IEEE, Vol. 76, No. 5, Pg. 597, May 1988.

Author's Address

John Linn
Secure Systems
Digital Equipment Corporation
85 Swanson Road, BXB1-2/D04
Boxborough, MA 01719-1326

Phone: 508-264-5491

EMail: Linn@ultra.enet.dec.com