Network Working Group
Request for Comments: 3217
Category: Informational

R. Housley
RSA Laboratories

December 2001

Triple-DES and RC2 Key Wrapping

Status of this Memo

Copyright Notice

Abstract

This document specifies the algorithm for wrapping one Triple-DES key with another Triple-DES key and the algorithm for wrapping one RC2 key with another RC2 key. These key wrap algorithms were originally published in section 12.6 of RFC 2630. They are republished since these key wrap algorithms have been found to be useful in contexts beyond those supported by RFC 2630.

1 Introduction

Management of symmetric cryptographic keys often leads to situations where one symmetric key is used to encrypt (or wrap) another. Key wrap algorithms are commonly used in two situations. First, key agreement algorithms (such as Diffie-Hellman [DH-X9.42]) generate a pairwise key-encryption key, and a key wrap algorithm is used to encrypt the content-encryption key or a multicast key with the pairwise key-encryption key. Second, a key wrap algorithm is used to encrypt the content-encryption key, multicast key, or session key in a locally generated storage key-encryption key or a key-encryption key that was distributed out-of-band.

This document specifies the algorithm for wrapping one Triple-DES key with another Triple-DES key [3DES], and it specifies the algorithm for wrapping one RC2 key with another RC2 key [RC2]. Encryption of a Triple-DES key with another Triple-DES key uses the algorithm specified in section 3. Encryption of a RC2 key with another RC2 key uses the algorithm specified in section 4. Both of these algorithms rely on the key checksum algorithm specified in section 2. Triple-DES and RC2 content-encryption keys are encrypted in Cipher Block Chaining (CBC) mode [MODES].

In this document, the key words MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, and MAY are to be interpreted as described by Scott Bradner in [STDWORDS].

2 Key Checksum

The key checksum algorithm is used to provide a key integrity check value. The algorithm is:

1. Compute a 20 octet SHA-1 [SHA1] message digest on the key that is to be wrapped.
2. Use the most significant (first) eight octets of the message digest value as the checksum value.

3 Triple-DES Key Wrapping and Unwrapping

This section specifies the algorithms for wrapping and unwrapping one Triple-DES key with another Triple-DES key [3DES].

The same key wrap algorithm is used for both Two-key Triple-DES and Three-key Triple-DES keys. When a Two-key Triple-DES key is to be wrapped, a third DES key with the same value as the first DES key is created. Thus, all wrapped Triple-DES keys include three DES keys. However, a Two-key Triple-DES key MUST NOT be used to wrap a Three-key Triple-DES key that is comprised of three unique DES keys.

3.1 Triple-DES Key Wrap

The Triple-DES key wrap algorithm encrypts a Triple-DES key with a Triple-DES key-encryption key. The Triple-DES key wrap algorithm is:

1. Set odd parity for each of the DES key octets comprising the Three-Key Triple-DES key that is to be wrapped, call the result CEK.
2. Compute an 8 octet key checksum value on CEK as described above in Section 2, call the result ICV.
3. Let CEKICV = CEK || ICV.
4. Generate 8 octets at random, call the result IV.
5. Encrypt CEKICV in CBC mode using the key-encryption key. Use the random value generated in the previous step as the initialization vector (IV). Call the ciphertext TEMP1.
6. Let TEMP2 = IV || TEMP1.
7. Reverse the order of the octets in TEMP2. That is, the most significant (first) octet is swapped with the least significant (last) octet, and so on. Call the result TEMP3.
8. Encrypt TEMP3 in CBC mode using the key-encryption key. Use an initialization vector (IV) of 0x4adda22c79e82105. The ciphertext is 40 octets long.

Note: When the same Three-Key Triple-DES key is wrapped in different key-encryption keys, a fresh initialization vector (IV) must be generated for each invocation of the key wrap algorithm.

3.2 Triple-DES Key Unwrap

The Triple-DES key unwrap algorithm decrypts a Triple-DES key using a Triple-DES key-encryption key. The Triple-DES key unwrap algorithm is:

1. If the wrapped key is not 40 octets, then error.
2. Decrypt the wrapped key in CBC mode using the key-encryption key. Use an initialization vector (IV) of 0x4adda22c79e82105. Call the output TEMP3.
3. Reverse the order of the octets in TEMP3. That is, the most significant (first) octet is swapped with the least significant (last) octet, and so on. Call the result TEMP2.
4. Decompose TEMP2 into IV and TEMP1. IV is the most significant (first) 8 octets, and TEMP1 is the least significant (last) 32 octets.
5. Decrypt TEMP1 in CBC mode using the key-encryption key. Use the IV value from the previous step as the initialization vector. Call the ciphertext CEKICV.
6. Decompose CEKICV into CEK and ICV. CEK is the most significant (first) 24 octets, and ICV is the least significant (last) 8 octets.
7. Compute an 8 octet key checksum value on CEK as described above in Section 2. If the computed key checksum value does not match the decrypted key checksum value, ICV, then error.
8. Check for odd parity each of the DES key octets comprising CEK. If parity is incorrect, then error.
9. Use CEK as a Triple-DES key.

3.3 Triple-DES Key Wrap Algorithm Identifier

Some security protocols employ ASN.1 [X.208-88, X.209-88], and these protocols employ algorithm identifiers to name cryptographic algorithms. To support these protocols, the Triple-DES key wrap algorithm has been assigned the following algorithm identifier:

```
id-alg-CMS3DESwrap OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 6 }
```

The AlgorithmIdentifier parameter field MUST be NULL.

3.4 Triple-DES Key Wrap Example

This section contains a Triple-DES Key Wrap example. Intermediate values corresponding to the named items in section 3.1 are given in hexadecimal.

```
CEK:     2923 bf85 e06d d6ae 5291 49f1 f1ba e9ea b3a7 da3d 860d 3e98
KEK:     255e 0d1c 07b6 46df b313 4cc8 43ba 8aa7 1f02 5b7c 0838 251f
ICV:     181b 7e96 86e0 4a4e
CEKICV:  2923 bf85 e06d d6ae 5291 49f1 f1ba e9ea b3a7 da3d 860d 3e98
         181b 7e96 86e0 4a4e
IV:      5dd4 cbfc 96f5 453b
TEMP1:   cfc1 a789 c675 dd2a b49a 3204 ef92 cc03 5c1f 973b 7a79 60f6
         a44d cc5f 729d 8449
TEMP2:   5dd4 cbfc 96f5 453b cfc1 a789 c675 dd2a b49a 3204 ef92 cc03
         5c1f 973b 7a79 60f6 a44d cc5f 729d 8449
TEMP3:   4984 9d72 5fcc 4da4 f660 797a 3b97 1f5c 03cc 92ef 0432 9ab4
         2add 75c6 89a7 c1cf 3b45 f596 fccb d45d
RESULT:  6901 0761 8ef0 92b3 b48c a179 6b23 4ae9 fa33 ebb4 1596 0403
         7db5 d6a8 4eb3 aac2 768c 6327 75a4 67d4
```

4 RC2 Key Wrapping and Unwrapping

This section specifies the algorithms for wrapping and unwrapping one RC2 key with another RC2 key [RC2].

RC2 supports variable length keys. RC2 128-bit keys MUST be used as key-encryption keys; however, the wrapped RC2 key MAY be of any size.

4.1 RC2 Key Wrap

The RC2 key wrap algorithm encrypts a RC2 key with a RC2 key-encryption key. The RC2 key wrap algorithm is:

1. Let the RC2 key be called CEK, and let the length of CEK in octets be called LENGTH. LENGTH is a single octet.
2. Let LCEK = LENGTH || CEK.
3. Let LCEKPAD = LCEK || PAD. If the length of LCEK is a multiple of 8, the PAD has a length of zero. If the length of LCEK is not a multiple of 8, then PAD contains the fewest number of random octets to make the length of LCEKPAD a multiple of 8.
4. Compute an 8 octet key checksum value on LCEKPAD as described above in Section 2, call the result ICV.
5. Let LCEKPADICV = LCEKPAD || ICV.
6. Generate 8 octets at random, call the result IV.
7. Encrypt LCEKPADICV in CBC mode using the key-encryption key. Use the random value generated in the previous step as the initialization vector (IV). Call the ciphertext TEMP1.

8. Let TEMP2 = IV || TEMP1.
9. Reverse the order of the octets in TEMP2. That is, the most significant (first) octet is swapped with the least significant (last) octet, and so on. Call the result TEMP3.
10. Encrypt TEMP3 in CBC mode using the key-encryption key. Use an initialization vector (IV) of 0x4adda22c79e82105.

Note: When the same RC2 key is wrapped in different key-encryption keys, a fresh initialization vector (IV) must be generated for each invocation of the key wrap algorithm.

4.2 RC2 Key Unwrap

The RC2 key unwrap algorithm decrypts a RC2 key using a RC2 key-encryption key. The RC2 key unwrap algorithm is:

1. If the wrapped key is not a multiple of 8 octets, then error.
2. Decrypt the wrapped key in CBC mode using the key-encryption key. Use an initialization vector (IV) of 0x4adda22c79e82105. Call the output TEMP3.
3. Reverse the order of the octets in TEMP3. That is, the most significant (first) octet is swapped with the least significant (last) octet, and so on. Call the result TEMP2.
4. Decompose the TEMP2 into IV and TEMP1. IV is the most significant (first) 8 octets, and TEMP1 is the remaining octets.
5. Decrypt TEMP1 in CBC mode using the key-encryption key. Use the IV value from the previous step as the initialization vector. Call the plaintext LCEKPADICV.
6. Decompose the LCEKPADICV into LCEKPAD, and ICV. ICV is the least significant (last) octet 8 octets. LCEKPAD is the remaining octets.
7. Compute an 8 octet key checksum value on LCEKPAD as described above in Section 2. If the computed key checksum value does not match the decrypted key checksum value, ICV, then error.
8. Decompose the LCEKPAD into LENGTH, CEK, and PAD. LENGTH is the most significant (first) octet. CEK is the following LENGTH octets. PAD is the remaining octets, if any.
9. If the length of PAD is more than 7 octets, then error.
10. Use CEK as an RC2 key.

4.3 RC2 Key Wrap Algorithm Identifier

Some security protocols employ ASN.1 [X.208-88, X.209-88], and these protocols employ algorithm identifiers to name cryptographic algorithms. To support these protocols, the RC2 key wrap algorithm has been assigned the following algorithm identifier:

```
id-alg-CMSRC2wrap OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 7 }
```

The AlgorithmIdentifier parameter field MUST be RC2wrapParameter:

```
RC2wrapParameter ::= RC2ParameterVersion

RC2ParameterVersion ::= INTEGER
```

The RC2 effective-key-bits (key size) greater than 32 and less than 256 is encoded in the RC2ParameterVersion. For the effective-key-bits of 40, 64, and 128, the rc2ParameterVersion values are 160, 120, and 58 respectively. These values are not simply the RC2 key length. Note that the value 160 must be encoded as two octets (00 A0), because the one octet (A0) encoding represents a negative number.

4.4 RC2 Key Wrap Example

This section contains a RC2 Key Wrap example. Intermediate values corresponding to the named items in section 4.1 are given in hexadecimal.

```
CEK:         b70a 25fb c9d8 6a86 050c e0d7 11ea d4d9
KEK:         fd04 fd08 0607 07fb 0003 feff fd02 fe05
LENGTH:      10
LCEK:        10b7 0a25 fbc9 d86a 8605 0ce0 d711 ead4 d9
PAD:         4845 cce7 fd12 50
LCEKPAD:     10b7 0a25 fbc9 d86a 8605 0ce0 d711 ead4
             d948 45cc e7fd 1250
ICV:         0a6f f19f db40 4988
LCEKPADICV: 10b7 0a25 fbc9 d86a 8605 0ce0 d711 ead4
             d948 45cc e7fd 1250 0a6f f19f db40 4988
IV:          c7d9 0059 b29e 97f7
TEMP1:       a01d a259 3793 1260 e48c 55f5 04ce 70b8
             ac8c d79e ffe8 9932 9fa9 8a07 a31f f7a7
TEMP2:       c7d9 0059 b29e 97f7 a01d a259 3793 1260
             e48c 55f5 04ce 70b8 ac8c d79e ffe8 9932
             9fa9 8a07 a31f f7a7
TEMP3:       a7f7 1fa3 078a a99f 3299 8eff 9ed7 8cac
             b870 ce04 f555 8ce4 6012 9337 59a2 1da0
             f797 9eb2 5900 d9c7
RESULT:      70e6 99fb 5701 f783 3330 fb71 e87c 85a4
             20bd c99a f05d 22af 5a0e 48d3 5f31 3898
             6cba afb4 b28d 4f35
```

5 References

[3DES] American National Standards Institute. ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation. 1998.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[DES] American National Standards Institute. ANSI X3.106, "American National Standard for Information Systems - Data Link Encryption". 1983.

[DH-X9.42] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[DSS] National Institute of Standards and Technology. FIPS Pub 186: Digital Signature Standard. 19 May 1994.

[MODES] National Institute of Standards and Technology. FIPS Pub 81: DES Modes of Operation. 2 December 1980.

[RANDOM] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RC2] Rivest, R., "A Description of the RC2 (r) Encryption Algorithm", RFC 2268, March 1998.

[SHA1] National Institute of Standards and Technology. FIPS Pub 180-1: Secure Hash Standard. 17 April 1995.

[STDWORDS] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[X.208-88] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.

[X.209-88] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

6 Security Considerations

Implementations must protect the key-encryption key. Compromise of the key-encryption key may result in the disclosure of all keys that have been wrapped with the key-encryption key, which may lead to the disclosure of all traffic protected with those wrapped key.

Implementations must randomly generate initialization vectors (IVs) and padding. The generation of quality random numbers is difficult. RFC 1750 [RANDOM] offers important guidance in this area, and Appendix 3 of FIPS Pub 186 [DSS] provides one quality PRNG technique.

If the key-encryption key and wrapped key are associated with different symmetric encryption algorithms, the effective security provided to data encrypted with the wrapped key is determined by the weaker of the two algorithms. If, for example, data is encrypted with 168-bit Triple-DES and that Triple-DES key is wrapped with a 40-bit RC2 key, then at most 40 bits of protection is provided. A trivial search to determine the value of the 40-bit RC2 key can recover Triple-DES key, and then the Triple-DES key can be used to decrypt the content. Therefore, implementers must ensure that key-encryption algorithms are as strong or stronger than content-encryption algorithms.

These key wrap algorithms specified in this document have been reviewed for use with Triple-DES and RC2, and they have not been reviewed for use with other encryption algorithms. Similarly, the key wrap algorithms make use of CBC mode [MODES], and they have not been reviewed for use with other cryptographic modes.

7 Acknowledgments

This document is the result of contributions from many professionals. I appreciate the hard work of all members of the IETF S/MIME Working Group. I extend a special thanks to Carl Ellison, Peter Gutmann, Bob Jueneman, Don Johnson, Burt Kaliski, John Pawling, and Jim Schaad for their support in defining these algorithms and generating this specification.

8 Author Address

Russell Housley
RSA Laboratories
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: rhousley@rsasecurity.com

9 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.