Network Working Group
Request for Comments: 2286
Category: Informational

J. Kapp
Reaper Technologies

February 1998

Test Cases for HMAC-RIPEMD160 and HMAC-RIPEMD128

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document provides two sets of test cases for HMAC-RIPEMD160 and HMAC-RIPEMD128, respectively. HMAC-RIPEMD160 and HMAC-RIPEMD128 are two constructs of the HMAC [HMAC] message authentication function using the RIPEMD-160 and RIPEMD-128 [RIPE] hash functions. The test cases and results provided in this document are meant to be used as a conformance test for HMAC-RIPEMD160 and HMAC-RIPEMD128 implementations.

1. Introduction

The general method for constructing a HMAC message authentication function using a particular hash function is described in section 2 of [HMAC].

In sections 2 and 3 test cases for HMAC-RIPEMD160 and HMAC-RIPEMD128, respectively are provided. Each case includes the key, the data, and the result. The values of keys and data are either hexadecimal numbers (prefixed by "0x") or ASCII character strings in double quotes. If a value is an ASCII character string, then the HMAC computation for the corresponding test case DOES NOT include the trailing null character ('\0') in the string.

The C source code of the functions used to generate HMAC-RIPEMD160 and HMAC-RIPEMD128 results is listed in the Appendix. Please Note that the functions provided are implemented in such a way as to be simple and easy to understand as a result they are not optimized in any way. The C source code for computing HMAC-MD5 can be found in [MD5].

2. Test Cases for HMAC-RIPEMD160

```
test_case =         1
key =               0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
key_len =           20
data =              "Hi There"
data_len =          8
digest =            0x24cb4bd67d20fc1a5d2ed7732dcc39377f0a5668

test_case =         2
key =               "Jefe"
key_len =           4
data =              "what do ya want for nothing?"
data_len =          28
digest =            0xdda6c0213a485a9e24f4742064a7f033b43c4069

test_case =         3
key =               0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
key_len =           20
data =              0xdd repeated 50 times
data_len =          50
digest =            0xb0b105360de759960ab4f35298e116e295d8e7c1

test_case =         4
key =               0x0102030405060708090a0b0c0d0e0f10111213141516171819
key_len =           25
data =              0xcd repeated 50 times
data_len =          50
digest =            0xd5ca862f4d21d5e610e18b4cf1beb97a4365ecf4

test_case =         5
key =               0x0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c
key_len =           20
data =              "Test With Truncation"
data_len =          20
digest =            0x7619693978f91d90539ae786500ff3d8e0518e39
digest-96 =         0x7619693978f91d90539ae786

test_case =         6
key =               0xaa repeated 80 times
key_len =           80
data =              "Test Using Larger Than Block-Size Key - Hash Key
                    First"
data_len =          54
digest =            0x6466ca07ac5eac29e1bd523e5ada7605b791fd8b

test_case =         7
key =               0xaa repeated 80 times
```

```
key_len =           80
data =              "Test Using Larger Than Block-Size Key and Larger
                    Than One Block-Size Data"
data_len =          73
digest =            0x69ea60798d71616cce5fd0871e23754cd75d5a0a
```

3. Test Cases for HMAC-RIPEMD128

```
test_case =         1
key =               0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
key_len =           16
data =              "Hi There"
data_len =          8
digest =            0xfbf61f9492aa4bbf81c172e84e0734db

test_case =         2
key =               "Jefe"
key_len =           4
data =              "what do ya want for nothing?"
data_len =          28
digest =            0x875f828862b6b334b427c55f9f7ff09b

test_case =         3
key =               0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
key_len =           16
data =              0xdd repeated 50 times
data_len =          50
digest =            0x09f0b2846d2f543da363cbec8d62a38d

test_case =         4
key =               0x0102030405060708090a0b0c0d0e0f10111213141516171819
key_len =           25
data =              0xcd repeated 50 times
data_len =          50
digest =            0xbdbbd7cf03e44b5aa60af815be4d2294

test_case =         5
key =               0x0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c0c
key_len =           16
data =              "Test With Truncation"
data_len =          20
digest =            0xe79808f24b25fd031c155f0d551d9a3a
digest-96 =         0xe79808f24b25fd031c155f0d

test_case =         6
key =               0xaa repeated 80 times
key_len =           80
data =              "Test Using Larger Than Block-Size Key - Hash Key
```

```
                        First"
data_len =              54
digest =                0xdc732928de98104a1f59d373c150acbb

test_case =             7
key =                   0xaa repeated 80 times
key_len =               80
data =                  "Test Using Larger Than Block-Size Key and Larger
                        Than One Block-Size Data"
data_len =              73
digest =                0x5c6bec96793e16d40690c237635f30c5
```

4. Security Considerations

This document raises no security issues. Discussion on the strength of the HMAC construction can be found in [HMAC].

References

[HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[OG] Oehler, M., and R. Glenn, "HMAC-MD5 IP Authentication with Replay Prevention", RFC 2085, February 1997

[CG] Chang, S., and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202, September 1997.

[RIPE] Dobbertin, H., Bosselaers A., and Preneel, B. "RIPEMD-160: A Strengthened Version of RIPEMD" April 1996

Author's Address

Justin S. Kapp
Reaper Technologies
The Post Office, Dunsop Bridge
Clitheroe, Lancashire.
BB7 3BB. United Kingdom

EMail: skapp@reapertech.com

Appendix

This code which implements HMAC-RIPEMD160 using an existing RIPEMD-160 library. It assumes that the RIPEMD-160 library has similar API's as those of the MD5 code described in RFC 1321. The code for HMAC-MD5, is similar, this HMAC-MD5 code is also listed in RFC 2104. To adapt this example to produce the HMAC-RIPEMD128 then replace each occurance of 'RMD160' with 'RMD128'.

```
#ifndef RMD160_DIGESTSIZE
#define RMD160_DIGESTSIZE  20
#endif

#ifndef RMD128_DIGESTSIZE
#define RMD128_DIGESTSIZE  16
#endif


/*  HMAC_RMD160 implements HMAC-RIPEMD160 */

void HMAC_RMD160(input, len, key, keylen, digest)
unsigned char *input;                          /* pointer to data stream */
int len;                                        /* length of data stream */
unsigned char *key;                     /* pointer to authentication key */
int keylen;                              /* length of authentication key */
unsigned char *digest;                           /* resulting MAC digest */
{
    RMD160_CTX context;
    unsigned char k_ipad[65];  /* inner padding - key XORd with ipad */
    unsigned char k_opad[65];  /* outer padding - key XORd with opad */
    unsigned char tk[RMD160_DIGESTSIZE];
    int i;

    /* if key is longer than 64 bytes reset it to key=SHA1(key) */
    if (keylen > 64) {
        RMD160_CTX      tctx;

        RMD160Init(&tctx);
        RMD160Update(&tctx, key, keylen);
        RMD160Final(tk, &tctx);

        key = tk;
        keylen = RMD160_DIGESTSIZE;
    }

        /* The HMAC_SHA1 transform looks like:

           RMD160(K XOR opad, RMD160(K XOR ipad, text))
```

```
           where K is an n byte key
           ipad is the byte 0x36 repeated 64 times
           opad is the byte 0x5c repeated 64 times
           and text is the data being protected */

        /* start out by storing key in pads */
    memset(k_ipad, 0x36, sizeof(k_ipad));
    memset(k_opad, 0x5c, sizeof(k_opad));

        /* XOR key with ipad and opad values */
    for (i=0; i<keylen; i++) {
        k_ipad[i] ^= key[i];
        k_opad[i] ^= key[i];
    }

        /* perform inner RIPEMD-160 */

    RMD160Init(&context);           /* init context for 1st pass */
    RMD160Update(&context, k_ipad, 64);  /* start with inner pad */
    RMD160Update(&context, input, len); /* then text of datagram */
    RMD160Final(digest, &context);         /* finish up 1st pass */

        /* perform outer RIPEMD-160 */
    RMD160Init(&context);           /* init context for 2nd pass */
    RMD160Update(&context, k_opad, 64);  /* start with outer pad */
    /* then results of 1st hash */
    RMD160Update(&context, digest, RMD160_DIGESTSIZE);
    RMD160Final(digest, &context);         /* finish up 2nd pass */

    memset(k_ipad, 0x00, sizeof(k_ipad));
    memset(k_opad, 0x00, sizeof(k_opad));
}
```

Full Copyright Statement