# The Secure Hash Algorithm Validation System (SHAVS)

March 1, 2004

Lawrence E. Bassham III

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

## TABLE OF CONTENTS

## 1 Introduction

This document, *The Secure Hash Algorithm Validation System (SHAVS)* specifies the procedures involved in validating implementations of the Secure Hash Algorithms in FIPS 180-2, *Secure Hash Standard (SHA-2)* [1]. The SHAVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the SHAVS. It includes the specifications for the three categories of tests that make up the SHAVS, i.e., the Short Messages Test, Long Messages Test, and Pseudorandomly Generated Messages Test. The requirements and administrative procedures specific to those seeking formal validation of an implementation of the Secure Hash Algorithm(s) are presented. The requirements described include the specific protocols for communication between the IUT and the SHAVS, the types of tests that the IUT must pass for formal validation, and general instructions for accessing and interfacing with the SHAVS. Several appendices with sample values for each of the tests are also provided. Additionally, an appendix with the format of the various input and output files of the tool is also provided.

## 2 Scope

This document specifies the tests required to validate IUTs for conformance to the Secure Hash Algorithm(s) as specified in [1]. When applied to IUTs that implement SHA, the SHAVS provides testing to determine the correctness of the algorithm implementation. In addition to determining conformance, the SHAVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the SHA implementation.

The SHAVS is composed of three types of validation tests, the Short Message Test, the Long Message Test, and the Pseudorandomly Generated Messages test. Additionally, the first two tests have an option to test bit-oriented or byte-oriented implementations. Byte-oriented implementations only hash messages with lengths divisible by 8, or integral bytes worth of data. Bit-oriented implementations can handle any length message (up to the limitation of the particular algorithm). While the specification for SHA specifies that messages up to at least $2^{64} - 1$ bits are possible, these tests only test messages up to a limited size of approximately 100,000 bits. This is adequate for detecting algorithmic and implementation errors.

## 3 Conformance

The successful completion of the tests contained within the SHAVS is required to claim conformance to FIPS 180-2. Testing for the cryptographic module in which the various algorithms specified in FIPS 180-2 is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules* [2].

# 4 Definitions, Symbols, and Abbreviations

## 4.1 Definitions

| **DEFINITION** | **MEANING** |
|---|---|
| CMT laboratory | Cryptographic Module Testing laboratory that operates the DSAVS |
| Secure Hash Algorithm(s) | The algorithm(s) specified in FIPS 180-2, *Secure Hash Standard (SHS)*. |

## 4.2 Symbols

| **SYMBOL** | **MEANING** |
|---|---|
| $L$ | The length of the message digest in bytes |
| Len | The length of a message in bits |
| $l_{min}$ | Minimum message length for Selected Long Message Test |
| $l_{max}$ | Maximum message length for Selected Long Message Test |
| $m$ | Number of bits in the message block |
| MD | A Message Digest |
| Msg | A message |
| $n$ | Number of bits in the message digest |

## 4.3 Abbreviations

| **ABBREVIATION** | **MEANING** |
|---|---|
| DEA | Data Encryption Algorithm |
| DSA | Digital Signature Algorithm specified in FIPS 186-2 |
| DSAVS | Digital Signature Algorithm Validation System |
| IUT | Implementation Under Test |

## 5 Design Philosophy Of The Secure Hash Algorithm Validation System

The SHAVS is designed to test conformance to SHA rather than provide a measure of a product's security. The validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation should not be interpreted as an evaluation or endorsement of overall product security.

The SHAVS has the following design philosophy:

1. The SHAVS is designed to allow the testing of an IUT at locations remote to the SHAVS. The SHAVS and the IUT communicate data via *REQUEST* and *RESPONSE* files.

2. The testing performed within the SHAVS utilizes statistical sampling (i.e., only a small number of the possible cases are tested); hence, the successful validation of a device does not imply 100% conformance with the standard.

## 6 SHAVS Tests

The SHAVS for the Secure Hash Algorithm(s) consists of three types of tests: The Short Messages Test, The Long Messages Test, and The Pseudorandomly Generated Messages Test. The SHAVS provides conformance testing for the algorithm, as well as testing for apparent implementation errors. The IUT may be implemented in software, firmware, hardware, or any combination thereof.

An IUT may implement SHA in either of two modes. The first mode is the byte-oriented mode. In this mode the IUT only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the IUT hashes messages of arbitrary length. Selecting the proper mode for an implementation will determine how the SHAVS tests will be performed. Both modes can be selected for the Short Messages Test and the Selected Long Messages Test. The Pseudorandomly Generated Messages Test is always run in byte-oriented mode.

### 6.1 Configuration Information

To initiate the validation process of the SHAVS, a vendor submits an application to an accredited laboratory requesting the validation of their implementation of SHA. The vendor's implementation is referred to as the Implementation Under Test (IUT). The request for validation includes background information describing the IUT along with information needed by the SHAVS to perform the specific tests. More specifically, the request for validation should include:

1. Vendor Name;
2. Product Name;
3. Product Version;
4. Implementation in software, firmware, or hardware;
5. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;
6. Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences); and

Whether the IUT handles bit-oriented messages or only byte-oriented messages.

## 6.2 The Short Messages Test

An implementation of SHA must be able to correctly generate message digests for messages of arbitrary length. The SHAVS tests this property by supplying the IUT with a number of short messages. The Short Messages Test has two versions, one for bit-oriented implementations and another for byte-oriented implementations.

### 6.2.1 The Short Messages Test for Bit-Oriented Implementations

This test generates a number of short messages equal to the number of bits in the hash block plus one. For example, SHA-1 defines a block length of $m$=512 bits. Therefore, for testing SHA-1, 513 unpredictable messages will be generated with lengths from 0 to 512 bits.

The SHAVS:

A. Generates $m + 1$ messages of length 0 to $m$.
B. Creates a *REQUEST* file (Filename: <Alg>ShortMsg.req) containing:
    1. The Product Information (vendor, product name, version); and
    2. The sequence of $m + 1$ messages to be hashed.

    Note: The CMT laboratory sends the *REQUEST* file to the IUT.
C. Creates a *FAX* file (Filename: <Alg>ShortMsg.fax) containing:
    1. The Product Name; and
    2. The $m + 1$ datasets containing:

a. The messages, and

b. The message digest of the message.

Note: The CMT laboratory retains the *FAX* file.

The IUT:

A. Generates message digests using the messages supplied by the SHAVS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: <Alg>ShortMsg.rsp) containing:

1. The Product Name; and
2. The $m$ + 1 datasets containing:
    a. The messages, and
    b. The message digest of the messages.

Note: The IUT sends the *RESPONSE* file to the SHAVS.

The SHAVS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHAVS verifies that the IUT generated the correct message digest.

B. If all message digests generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

### 6.2.2 The Short Messages Test for Byte-Oriented Implementations

This test generates a number of short messages equal to the number of bytes in the hash block plus one. For example, SHA-1 defines a block length of $m$=512 bits resulting in a block size of 64 bytes. Therefore, for testing SHA-1, 65 unpredictable messages will be generated with lengths of 0, 8, 16, …, 512 bits.

The SHAVS:

A. Generates $m$/8 + 1 messages of length 0, 8, 16, …, $m$.

B. Creates a *REQUEST* file (Filename: <Alg>ShortMsg.req) containing:

1. The Product Name; and
2. The sequence of $m$/8 + 1 messages to be hashed.

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

C. Creates a *FAX* file (Filename: <Alg>ShortMsg.fax) containing:

1. The Product Name; and

2. The *m*/8 + 1 datasets containing:
    a. The message, and
    b. The message digest of the message.

Note: The CMT laboratory retains the *FAX* file.

The IUT:

A. Generates message digests using the messages supplied by the SHAVS in the *REQUEST* file.
B. Creates a *RESPONSE* file (Filename: <Alg>ShortMsg.rsp) containing:
    1. The Product Name; and
    2. The *m*/8 + 1 datasets containing:
        a. The message, and
        b. The message digest of the message.

    Note: The IUT sends the *RESPONSE* file to the SHAVS.

The SHAVS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHAVS verifies that the IUT generated the correct message digest.
B. If all message digests generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

## 6.3 The Selected Long Messages Test

An implementation of SHA must be able to correctly generate message digests for messages that span multiple message blocks. The SHAVS tests this property by supplying selected unpredictable long messages to the IUT. The IUT then generates message digests for each message. The Selected Long Messages Test has two versions, one for bit-oriented implementations and another for byte-oriented implementations.

### 6.3.1 The Selected Long Messages Test for Bit-Oriented Implementations

This test generates a number of long messages equal to the number of bits in the hash block, *m*. These message range in size from $m+99 \le len \le m*100$. For example, SHA-1 defines a block length of $m=512$ bits. Therefore, for testing SHA-1, 512 unpredictable long messages will be generated with lengths (in bits) of:

$512 + 99*i,\ 1 \le i \le 512.$

The SHAVS:

A. Generates *m* messages of the length specified above.

B. Creates a *REQUEST* file (Filename: <Alg>LongMsg.req) containing:

   1. The Product Name; and
   2. The sequence of *m* messages to be hashed.

   Note: The CMT laboratory sends the *REQUEST* file to the IUT.

C. Creates a *FAX* file (Filename: <Alg>LongMsg.fax) containing:

   1. The Product Name; and
   2. The *m* datasets containing:
      a. The message, and
      b. The message digest of the message.

   Note: The CMT laboratory retains the *FAX* file at the SHAVS.

The IUT:

A. Generates message digests using the messages supplied by the SHAVS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: <Alg>LongMsg.rsp) containing:

   1. The Product Name; and
   2. The *m* datasets containing:
      a. The message, and
      b. The message digest of the message.

      Note: The IUT sends the *RESPONSE* file to the SHAVS.

The SHAVS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHAVS verifies that the IUT generated the correct message digest.

B. If all message digests generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

### 6.3.2 The Selected Long Messages Test for Byte-Oriented Implementations

This test generates a number of long messages equal to the number of bytes in the hash block, $m/8$. These message range in size from $m+99 \leq len \leq m*100$. For example, SHA-1 defines a block length of $m/8 = 64$ bytes. Therefore, for testing SHA-1, 64 unpredictable long messages will be generated with lengths (in bits) of:

$512 + 8*99*i,\ 1 \leq i \leq 64.$

The SHAVS:

A. Generates *m*/8 messages of the length specified above.

B. Creates a *REQUEST* file (Filename: LongMsg.req) containing:

    1. The Product Information (vendor, product name, version); and

    2. The sequence of *m*/8 messages to be hashed.

    Note: The CMT laboratory sends the *REQUEST* file to the IUT.

C. Creates a *FAX* file (Filename: LongMsg.fax) containing:

    1. The Product Information (vendor, product name, version); and

    2. The *m*/8 datasets containing:

        a. The message, and

        b. The message digest of the message.

    Note: The CMT laboratory retains the *FAX* file.

The IUT:

A. Generates message digests using the message supplied by the SHAVS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: LongMsg.rsp) containing:

    1. The Product Information (vendor, product name, version); and

    2. The *m*/8 datasets containing:

        a. The message, and

        b. The message digest of the message.

    Note: The IUT sends the *RESPONSE* file to the SHAVS.

The SHAVS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHAVS verifies that the IUT generated the correct message digest.

    B. If all message digests generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

## 6.4 The Pseudorandomly Generated Messages Test

The SHAVS tests the correctness of message digests generated from pseudorandomly generated messages by supplying a seed, *Seed*, of length *n* bits. This seed is used by a pseudorandom function to generate 100,000 message digests. 100 of the 100,000 message digests, once every 1,000 hashes, are recorded as checkpoints to the operation of the generator. The IUT uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The SHAVS compares each

```
INPUT: Seed  - A random seed n bits long
{
     for (j=0; j<100; j++) {
          $MD_0$ = $MD_1$ = $MD_2$ = Seed;
          for (i=3; i<1003; i++) {
               $M_i$ = $MD_{i-3}$ || $MD_{i-2}$ || $MD_{i-1}$;
               $MD_i$ = SHA($M_i$);
               }
          $MD_j$ = Seed = $MD_i$;
          OUTPUT: $MD_j$
          }
     }
```

of the recorded 100 message digests with those generated by the IUT.

The procedure used to generate the 100 checkpoint messages digest is as follows:

1) 100,000 pseudorandom messages are generated by using previous message digests as the input to the hash algorithm; and,

2) After every 1,000 hashes a sample is taken and is provided as a checkpoint. These checkpoints are denoted $MD_j$ in Figure 1.

**Figure 1: Code for Generating Pseudorandom Messages**

The SHAVS:

A. Generates a seed, *Seed*, of length *n* bits.
B. Creates a *REQUEST* file (Filename: Monte.req) containing:
    1. The Product Information (vendor, product name, version); and
    2. The *Seed*.

    Note: The CMT laboratory sends the *REQUEST* file to the IUT.
C. Creates a *FAX* file (Filename: Monte.fax) containing:
    1. The Product Information (vendor, product name, version);
    2. The *Seed*; and
    3. The 100 message digests, $MD_j$, from Figure 1.

    Note: The CMT laboratory retains the *FAX* file at the SHAVS.

The IUT:

A. Generates the 100 message digest using the *Seed* supplied by the SHAVS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: Monte.rsp) containing:

   1. The Product Information (vendor, product name, version);

   2. The *Seed*; and

   3. The 100 message digest $MD_j$ from Figure 1.

   Note: The IUT sends the *RESPONSE* file to the SHAVS.

The SHAVS:

A. Compares the *RESPONSE* file with the *FAX* file. The SHAVS verifies that the IUT generated the correct message digests.

B. If all message digests generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

# Appendix A Samples Values for SHA-1 Tests

Each example contains the length of the sample message that is to be hashed, the hex representation of the message, *M*, and the message digest, *MD*, of the message using SHA-1. For bit-oriented messages, the message is in the most significant digits of the hex representation. In Example 1 below, the binary representation of the 5-bit message `98` is `10011`.

## A.1 Examples Of The Short Messages Test for SHA-1

Example 1:

|  |  |
|---|---|
| Length: | `5` |
| M: | `98` |
| MD: | `29826b00 3b906e66 0eff4027 ce98af35 31ac75ba` |

Example 2:

|  |  |
|---|---|
| Length: | `8` |
| M: | `5e` |
| MD: | `5e6f80a3 4a9798ca fc6a5db9 6cc57ba4 c4db59c2` |

Example 3:

|  |  |
|---|---|
| Length: | `123` |
| M: | `49b2aec2 594bbe3a 3b117542 d94ac880` |
| MD: | `6239781e 03729919 c01955b3 ffa8acb6 0b988340` |

Example 4:

|  |  |
|---|---|
| Length: | `128` |
| M: | `9a7dfdf1 ecead06e d646aa55 fe757146` |
| MD: | `82abff66 05dbe1c1 7def12a3 94fa22a8 2b544a35` |

## A.2 Examples Of The Selected Long Messages Test for SHA-1

Example 1:

| | |
|---|---|
| Length: | 611 |
| M: | 65f93299 5ba4ce2c b1b4a2e7 1ae70220 aacec896 2dd4499c |
| | bd7c887a 94eaaa10 1ea5aabc 529b4e7e 43665a5a f2cd03fe |
| | 678ea6a5 005bba3b 082204c2 8b9109f4 69dac92a aab3aa7c |
| | 11a1b32a e0 |
| MD: | 8c5b2a5d dae5a97f c7f9d856 61c672ad bf7933d4 |

Example 2:

| | |
|---|---|
| Length: | 1304 |
| M: | f78f9214 1bcd170a e89b4fba 15a1d59f 3fd84d22 3c9251bd |
| | acbbae61 d05ed115 a06a7ce1 17b7beea d24421de d9c32592 |
| | bd57edea e39c39fa 1fe8946a 84d0cf1f 7beead17 13e2e095 |
| | 9897347f 67c80b04 00c20981 5d6b10a6 83836fd5 562a56ca |
| | b1a28e81 b6576654 631cf165 66b86e3b 33a108b0 5307c00a |
| | ff14a768 ed735060 6a0f85e6 a91d396f 5b5cbe57 7f9b3880 |
| | 7c7d523d 6d792f6e bc24a4ec f2b3a427 cdbbfb |
| MD: | cb0082c8 f197d260 991ba6a4 60e76e20 2bad27b3 |

## A.3 Examples of The Pseudorandomly Generated Messages Test for SHA-1

A sample seed for this routine is the following string:

| | |
|---|---|
| Seed: | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |

The first few messages, $M_i$, and hashes, $Md_i$, contained in the inner loop of the routine in Figure 1 are:

| | |
|---|---|
| $M_{i=3}$: | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| $MD_{i=3}$: | 5c1f6ab1 dd1a1b92 313ef55b d94e5d90 eee5fd42 |
| $M_{i=4}$: | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| | 5c1f6ab1 dd1a1b92 313ef55b d94e5d90 eee5fd42 |
| $MD_{i=4}$: | 3dc3b220 871bb348 8a66c39d ea577c2d 1ce40168 |
| $M_{i=5}$: | d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d |
| | 5c1f6ab1 dd1a1b92 313ef55b d94e5d90 eee5fd42 |
| | 3dc3b220 871bb348 8a66c39d ea577c2d 1ce40168 |
| $MD_{i=5}$: | dd4b2577 7737c82c f228ac14 cf04b905 b081ac72 |
| $M_{i=6}$: | 5c1f6ab1 dd1a1b92 313ef55b d94e5d90 eee5fd42 |
| | 3dc3b220 871bb348 8a66c39d ea577c2d 1ce40168 |
| | dd4b2577 7737c82c f228ac14 cf04b905 b081ac72 |
| $MD_{i=6}$: | 129aef9d 88d256d5 a667637e a215d7d8 4d710156 |

The first few message digests, $MD_j$, that are the output of the routine found in Figure 1 are:

| | |
|---|---|
| $MD_{j=0}$: | e2168368 19477c7f 78e0d843 fe4ff1b6 d6c14cd4 |
| $MD_{j=1}$: | a2dbc7a5 b1c6c0a8 bcb7aaa4 1252a6a7 d0690dbc |
| $MD_{j=2}$: | db1f9050 bb863dfe f4ce3718 6044e2ee b17ee013 |
| $MD_{j=3}$: | 127fdedf 43d372a5 1d5747c4 8fbffe38 ef6cdf7b |

## Appendix B Sample Values for SHA-224 Tests

Each example contains the length of the sample message that is to be hashed, the hex representation of the message, *M*, and the message digest, *MD*, of the message using SHA-224. For bit-oriented messages, the message is in the most significant digits of the hex representation. In Example 1 below, the binary representation of the 5-bit message `68` is `01101`.

### B.1 Examples Of The Short Messages Test for SHA-224

Example 1:

| | |
|---|---|
| Length: | `5` |
| M: | `68` |
| MD: | `e3b04855 2c3c387b cab37f6e b06bb79b`<br>`96a4aee5 ff27f515 31a9551c` |

Example 2:

| | |
|---|---|
| Length: | `8` |
| M: | `07` |
| MD: | `00ecd5f1 38422b8a d74c9799 fd826c53`<br>`1bad2fca bc7450be e2aa8c2a` |

Example 3:

| | |
|---|---|
| Length: | `123` |
| M: | `f07006f2 5a0bea68 cd76a295 87c28da0` |
| MD: | `1b01db6c b4a9e43d ed1516be b3db0b87`<br>`b6d1ea43 187462c6 08137150` |

Example 4:

| | |
|---|---|
| Length: | `128` |
| M: | `18804005 dd4fbd15 56299d6f 9d93df62` |
| MD: | `df90d78a a78821c9 9b40ba4c 966921ac`<br>`cd8ffb1e 98ac388e 56191db1` |

## B.2 Examples Of The Selected Long Messages Test for SHA-224

Example 1:

```
Length:     611

M:          a2be6e46 32810902 94d9ce94 82656942 3a3a305e
            d5e2116c d4a4c987 fc065700 6491b149 ccd4b511
            30ac62b1 9dc248c7 44543d20 cd3952dc ed1f06cc
            3b18b91f 3f55633e cc3085f4 907060d2 e0

MD:         54bea6ea b8195a2e b0a7906a 4b4a8766

            66300eef bd1f3b84 74f9cd57
```

Example 2:

```
Length:     1304

M:          55b21007 9c61b53a dd520622 d1ac97d5 cdbe8cb3
            3aa0ae34 4517bee4 d7ba09ab c8533c52 50887a43
            bebbac90 6c2e1837 f26b36a5 9ae3be78 14d50689
            6b718b2a 383ecdac 16b96125 553f416f f32c6674
            c74599a9 005386d9 ce111224 5f48ee47 0d396c1e
            d63b9267 0ca56ec8 4deea814 b6135eca 54392bde
            db9489bc 9b875a8b af0dc1ae 78573691 4ab7daa2
            64bc079d 269f2c0d 7eddd810 a426145a 0776f67c 878273

MD:         0b31894e c8937ad9 b91bdfbc ba294d9a

            defaa18e 09305e9f 20d5c3a4
```

## B.3 Examples of The Pseudorandomly Generated Messages Test for SHA-224

A sample seed for this routine is the following string:

Seed:
```
d0569cb3 665a8a43 eb6ea23d 75a3c4d2
054a0d7d 66a9ca99 c9ceb027
```

The first few messages, $M_i$, and hashes, $Md_i$, contained in the inner loop of the routine in Figure 1 are:

$M_{i=3}$:
```
d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d
66a9ca99 c9ceb027 d0569cb3 665a8a43 eb6ea23d
75a3c4d2 054a0d7d 66a9ca99 c9ceb027 d0569cb3
665a8a43 eb6ea23d 75a3c4d2 054a0d7d 66a9ca99
c9ceb027
```

$MD_{i=3}$:
```
38cea69d 7be80c14 ab3becb1 e02e7b67
ced60a05 1c4a0100 e3e613b4
```

$M_{i=4}$:
```
d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d
66a9ca99 c9ceb027 d0569cb3 665a8a43 eb6ea23d
75a3c4d2 054a0d7d 66a9ca99 c9ceb027 38cea69d
7be80c14 ab3becb1 e02e7b67 ced60a05 1c4a0100
e3e613b4
```

$MD_{i=4}$:
```
ee210ffc 8db91566 6c2a4fcb 37d9e5c5
8993c01b dccf706a 4077f955
```

$M_{i=5}$:
```
d0569cb3 665a8a43 eb6ea23d 75a3c4d2 054a0d7d
66a9ca99 c9ceb027 38cea69d 7be80c14 ab3becb1
e02e7b67 ced60a05 1c4a0100 e3e613b4 ee210ffc
8db91566 6c2a4fcb 37d9e5c5 8993c01b dccf706a
4077f955
```

$MD_{i=5}$:
```
915eb09c 83e58cb8 ed11d8f4 bb361798
42146256 08de7de8 dcab4030
```

$M_{i=6}$:
```
38cea69d 7be80c14 ab3becb1 e02e7b67 ced60a05
1c4a0100 e3e613b4 ee210ffc 8db91566 6c2a4fcb
37d9e5c5 8993c01b dccf706a 4077f955 915eb09c
83e58cb8 ed11d8f4 bb361798 42146256 08de7de8
dcab4030
```

$MD_{i=6}$:
```
25867f03 8cfcf093 4c575ba5 67bb23a4
40e0fab9 84afbe23 a5e26b3b
```

The first few message digests, $MD_j$, that are the output of the routine found in Figure 1 are:

$MD_{j=0}$:
```
100966a5 b4fde0b4 2e2a6c59 53d4d7f4
1ba7cf79 fd2df431 416734be
```

$MD_{j=1}$:
```
1dca396b 0c417715 defaae96 41e10a2e
99d55abc b8a00061 eb3be8bd
```

$MD_{j=2}$:
```
1864e627 bdb23199 73cd5ed7 d68da71d
8bf0f983 d8d9ab32 c34adb34
```

$MD_{j=3}$:
```
a2406481 fc1bcaf2 4dd08e67 52e84470
9563fb91 6227fed5 98eb621f
```

## Appendix C Sample Values for SHA-256 Tests

Each example contains the length of the sample message that is to be hashed, the hex representation of the message, *M*, and the message digest, *MD*, of the message using SHA-256. For bit-oriented messages, the message is in the most significant digits of the hex representation. In Example 1 below, the binary representation of the 5-bit message `68` is `01101`.

### C.1 Examples Of The Short Messages Test for SHA-256

Example 1:

```
Length:  5
M:       68
MD:      d6d3e02a 31a84a8c aa9718ed 6c2057be
         09db45e7 823eb507 9ce7a573 a3760f95
```

Example 2:

```
Length:  8
M:       19
MD:      68aa2e2e e5dff96e 3355e6c7 ee373e3d
         6a4e17f7 5f9518d8 43709c0c 9bc3e3d4
```

Example 3:

```
Length:  123
M:       be2746c6 db52765f db2f8870 0f9a7360
MD:      77ec1dc8 9c821ff2 a1279089 fa091b35
         b8cd960b caf7de01 c6a76807 56beb972
```

Example 4:

```
Length:  128
M:       e3d72570 dcdd787c e3887ab2 cd684652
MD:      175ee69b 02ba9b58 e2b0a5fd 13819cea
         573f3940 a94f8251 28cf4209 beabb4e8
```

## C.2 Examples Of The Selected Long Messages Test for SHA-256

Example 1:

Length: `611`

M:
```
3e740371 c810c2b9 9fc04e80 4907ef7c f26be28b
57cb58a3 e2f3c007 166e49c1 2e9ba34c 01040691
29ea7615 64254570 3a2bd901 e16eb0e0 5deba014
ebff6406 a07d5436 4eff742d a779b0b3 a0
```

MD:
```
3e9ad646 8bbbad2a c3c2cdc2 92e018ba

5fd70b96 0cf16797 77fce708 fdb066e9
```

Example 2:

Length: `1304`

M:
```
8326754e 2277372f 4fc12b20 527afef0 4d8a0569
71b11ad5 7123a7c1 37760000 d7bef6f3 c1f7a908
3aa39d81 0db31077 7dab8b1e 7f02b84a 26c77332
5f8b2374 de7a4b5a 58cb5c5c f35bcee6 fb946e5b
d694fa59 3a8beb3f 9d6592ec edaa66ca 82a29d0c
51bcf933 6230e5d7 84e4c0a4 3f8d79a3 0a165cba
be452b77 4b9c7109 a97d138f 12922896 6f6c0adc
106aad5a 9fdd3082 5769b2c6 71af6759 df28eb39

3d54d6
```

MD:
```
97dbca7d f46d62c8 a422c941 dd7e835b

8ad33617 63f7e9b2 d95f4f0d a6e1ccbc
```

## C.3 Examples of The Pseudorandomly Generated Messages Test for SHA-256

A sample seed for this routine is the following string:

Seed:
```
f41ece26 13e45739 15696b5a dcd51ca3
28be3bf5 66a9ca99 c9ceb027 9c1cb0a7
```

The first few messages, $M_i$, and hashes, $Md_i$, contained in the inner loop of the routine in Figure 1 are:

$M_{i=3}$:
```
f41ece26 13e45739 15696b5a dcd51ca3 28be3bf5
66a9ca99 c9ceb027 9c1cb0a7 f41ece26 13e45739
15696b5a dcd51ca3 28be3bf5 66a9ca99 c9ceb027
9c1cb0a7 f41ece26 13e45739 15696b5a dcd51ca3
28be3bf5 66a9ca99 c9ceb027 9c1cb0a7
```

$MD_{i=3}$:
```
fddf1b37 dd34b3b2 01d43c57 bcde1158
38f0df70 1da93c3b f2c9c868 96e7e6c7
```

$M_{i=4}$:
```
f41ece26 13e45739 15696b5a dcd51ca3 28be3bf5
66a9ca99 c9ceb027 9c1cb0a7 f41ece26 13e45739
15696b5a dcd51ca3 28be3bf5 66a9ca99 c9ceb027
9c1cb0a7 fddf1b37 dd34b3b2 01d43c57 bcde1158
38f0df70 1da93c3b f2c9c868 96e7e6c7
```

$MD_{i=4}$:
```
3b9e2613 dc71d499 25cc3258 a3a4201a
ea4336c2 a648ca8d ffb45bbd ad4835e8
```

$M_{i=5}$:
```
f41ece26 13e45739 15696b5a dcd51ca3 28be3bf5
66a9ca99 c9ceb027 9c1cb0a7 fddf1b37 dd34b3b2
01d43c57 bcde1158 38f0df70 1da93c3b f2c9c868
96e7e6c7 3b9e2613 dc71d499 25cc3258 a3a4201a
ea4336c2 a648ca8d ffb45bbd ad4835e8
```

$MD_{i=5}$:
```
9fbac41c 7453a2c8 8fd3fed1 f685ef27
587bebcc 573209bc c1b9f9ee cf03c1fd
```

$M_{i=6}$:
```
fddf1b37 dd34b3b2 01d43c57 bcde1158 38f0df70
1da93c3b f2c9c868 96e7e6c7 3b9e2613 dc71d499
25cc3258 a3a4201a ea4336c2 a648ca8d ffb45bbd
ad4835e8 9fbac41c 7453a2c8 8fd3fed1 f685ef27
587bebcc 573209bc c1b9f9ee cf03c1fd
```

$MD_{i=6}$:
```
b125c98b 1a9d25f3 37b5a788 15b6b7a7
f091d328 80e8681b dec8584b 92aa3bf8
```

The first few message digests, $MD_j$, that are the output of the routine found in Figure 1 are:

| | |
|---|---|
| $MD_{j=0}$: | `83d28614 d49c3adc 1d6fc05d b5f48037` |
| | `c056f8d2 a4ce44ec 6457dea5 dd797cd1` |
| $MD_{j=1}$: | `99dbe312 7ef2e93d d9322d6a 07909eb3` |
| | `3b63995e 529b3f95 4b858162 1bb74d39` |
| $MD_{j=2}$: | `8d4be295 bb64661c a3c7efd1 29a2f725` |
| | `b33072db dde32385 b9a87b9a f88ea76f` |
| $MD_{j=3}$: | `40af5d3f 9716b040 df9408e3 1536b70f` |
| | `f906ec51 b00447ca 97d7dd97 c12411f4` |

# Appendix D Sample Values for SHA-384 Tests

Each example contains the length of the sample message that is to be hashed, the hex representation of the message, *M*, and the message digest, *MD*, of the message using SHA-384. For bit-oriented messages, the message is in the most significant digits of the hex representation. In Example 1 below, the binary representation of the 5-bit message `10` is `10000`.

## D.1 Examples Of The Short Messages Test for SHA-384

Example 1:

| | |
|---|---|
| Length: | `5` |
| M: | `10` |
| MD: | `8d17be79 e32b6718 e07d8a60 3eb84ba0 478f7fcf d1bb9399 5f7d1149 e09143ac 1ffcfc56 820e469f 3878d957 a15a3fe4` |

Example 2:

| | |
|---|---|
| Length: | `8` |
| M: | `b9` |
| MD: | `bc8089a1 9007c0b1 4195f4ec c74094fe c64f01f9 0929282c 2fb39288 1578208a d466828b 1c6c283d 2722cf0a d1ab6938` |

Example 3:

| | |
|---|---|
| Length: | `123` |
| M: | `8bc500c7 7ceed987 9da98910 7ce0aaa0` |
| MD: | `d8c43b38 e12e7c42 a7c9b810 299fd6a7 70bef309 20f17532 a898de62 c7a07e42 93449c0b 5fa70109 f0783211 cfc4bce3` |

Example 4:

| | |
|---|---|
| Length: | `128` |
| M: | `a41c4977 79c0375f f10a7f4e 08591739` |
| MD: | `c9a68443 a0058122 56b8ec76 b00516f0 dbb74fab 26d66591 3f194b6f fb0e91ea 9967566b 58109cbc 675cc208 e4c823f7` |

## D.2 Examples Of The Selected Long Messages Test for SHA-384

Example 1:

Length: 1123

M:

```
68f50179 2dea9796 767022d9 3da71679 309920fa
1012aea3 57b2b133 1d40a1d0 3c41c240 b3c9a75b
4892f4c0 724b68c8 75321ab8 cfe5023b d375bc0f
94bd89fe 04f29710 5d7b82ff c0021aeb 1ccb674f
5244ea34 97de26a4 191c5f62 e5e9a2d8 082f0551
f4a53068 26e91cc0 06ce1bf6 0ff719d4 2fa521c8
71cd2394 d96ef446 8f21966b 41f2ba80 c26e83a9

e0
```

MD:

```
5860e8de 91c21578 bb4174d2 27898a98 e0b45c4c
760f0095 49495614 daedc077 5d92d11d 9f8ce9b0
64eeac8d afc3a297
```

Example 2:

Length: 1816

M:

```
399669e2 8f6b9c6d bcbb6912 ec10ffcf 74790349
b7dc8fbe 4a8e7b3b 5621db0f 3e7dc87f 823264bb
e40d1811 c9ea2061 e1c84ad1 0a23fac1 727e7202
fc3f5042 e6bf58cb a8a2746e 1f64f9b9 ea352c71
1507053c f4e5339d 52865f25 cc22b5e8 7784a12f
c961d66c b6e89573 199a2ce6 565cbdf1 3dca4038
32cfcb0e 8b7211e8 3af32a11 ac17929f f1c073a5
1cc027aa edeff85a ad7c2b7c 5a803e24 04d96d2a
77357bda 1a6daeed 17151cb9 bc5125a4 22e941de
0ca0fc50 11c23ecf fefdd096 76711cf3 db0a3440
720e1615 c1f22fbc 3c721de5 21e1b99b a1bd5577
40864214 7ed096
```

MD:

```
4f440db1 e6edd289 9fa335f0 9515aa02 5ee177a7
9f4b4aaf 38e42b5c 4de660f5 de8fb2a5 b2fbd2a3
cbffd20c ff1288c0
```

## D.3 Examples of The Pseudorandomly Generated Messages Test for SHA-384

A sample seed for this routine is the following string:

```
Seed:     8240bc51 e4ec7ef7 6d18e352 04a19f51 a5213a73
          a81d6f94 4680d307 5948b7e4 63804ea3 d26e13ea
          820d65a4 84be7453
```

The first few messages, $M_i$, and hashes, $Md_i$, contained in the inner loop of the routine in Figure 1 are:

$M_{i=3}$:

```
8240bc51 e4ec7ef7 6d18e352 04a19f51 a5213a73
a81d6f94 4680d307 5948b7e4 63804ea3 d26e13ea
820d65a4 84be7453 8240bc51 e4ec7ef7 6d18e352
04a19f51 a5213a73 a81d6f94 4680d307 5948b7e4
63804ea3 d26e13ea 820d65a4 84be7453 8240bc51
e4ec7ef7 6d18e352 04a19f51 a5213a73 a81d6f94
4680d307 5948b7e4 63804ea3 d26e13ea 820d65a4
84be7453
```

$MD_{i=3}$:

```
8ff53918 fb4da556 6f074f64 1f2ce018 a688fcbc
b2efdb13 ba6b3578 bcd5a489 8d0b2d88 702dbbd8
310bea5a 975adf20
```

$M_{i=4}$:

```
8240bc51 e4ec7ef7 6d18e352 04a19f51 a5213a73
a81d6f94 4680d307 5948b7e4 63804ea3 d26e13ea
820d65a4 84be7453 8240bc51 e4ec7ef7 6d18e352
04a19f51 a5213a73 a81d6f94 4680d307 5948b7e4
63804ea3 d26e13ea 820d65a4 84be7453 8ff53918
fb4da556 6f074f64 1f2ce018 a688fcbc b2efdb13
ba6b3578 bcd5a489 8d0b2d88 702dbbd8 310bea5a
975adf20
```

$MD_{i=4}$:

```
b5d6eb75 ae9bd90a 27e46f0d 732b53bc b74b8ff2
f5df4d53 4ea78e8d beb31273 78bfafb1 35d59ec5
3d1f15f7 c839cfaf
```

$M_{i=5}$:

```
8240bc51 e4ec7ef7 6d18e352 04a19f51 a5213a73
a81d6f94 4680d307 5948b7e4 63804ea3 d26e13ea
820d65a4 84be7453 8ff53918 fb4da556 6f074f64
1f2ce018 a688fcbc b2efdb13 ba6b3578 bcd5a489
8d0b2d88 702dbbd8 310bea5a 975adf20 b5d6eb75
ae9bd90a 27e46f0d 732b53bc b74b8ff2 f5df4d53
4ea78e8d beb31273 78bfafb1 35d59ec5 3d1f15f7
c839cfaf
```

$MD_{i=5}$:

```
7870a434 da5d941c 4cdde45a de8e4fd1 178ea3d3
6ecd1231 fe675eb9 a5e61c01 34e495ba 2a4c3b11
1a861abc 679f9a37
```

$M_{i=6}$:

```
8ff53918 fb4da556 6f074f64 1f2ce018 a688fcbc
b2efdb13 ba6b3578 bcd5a489 8d0b2d88 702dbbd8
310bea5a 975adf20 b5d6eb75 ae9bd90a 27e46f0d
732b53bc b74b8ff2 f5df4d53 4ea78e8d beb31273
78bfafb1 35d59ec5 3d1f15f7 c839cfaf 7870a434
da5d941c 4cdde45a de8e4fd1 178ea3d3 6ecd1231
fe675eb9 a5e61c01 34e495ba 2a4c3b11 1a861abc
679f9a37
```

$MD_{i=6}$:

```
1ae75e92 d9a24acc fe1bc1b6 abe0b643 76efbb19
cc49fa08 fbd3bc17 724fb774 5045e826 daa4f0de
fca7adcf 165561a4
```

The first few message digests, $MD_j$, that are the output of the routine found in Figure 1 are:

$MD_{j=0}$:

```
ce44d7d6 3ae0c914 82998cf6 62a51ec8 0bf6fc68
661a3c57 f8756611 2bd635a7 43ea904d eb7d7a42
ac808cab e697f38f
```

$MD_{j=1}$:

```
f9c6d261 881fee41 acd39e67 aa8d0bad 507c7363
eb67e2b8 1f45759f 9c0fd7b5 03df1a0b 9e80bde7
bc333d75 b804197d
```

$MD_{j=2}$:

```
d96512d8 c9f4a7a4 967a366c 01c6fd97 384225b5
8343a882 64847c18 e4ef8ab7 aee4765f fbc3e30b
d485d363 8a01418f
```

$MD_{j=3}$:

```
0ca76bd0 813af150 9e170907 a9600593 8bc98562
8290b25f ef73cf6f ad68ddba 0ac8920c 94e05416
07b0915a 7b4457f7
```

# Appendix E  Sample Values for SHA-512 Tests

Each example contains the length of the sample message that is to be hashed, the hex representation of the message, *M*, and the message digest, *MD*, of the message using SHA-512. For bit-oriented messages, the message is in the most significant digits of the hex representation. In Example 1 below, the binary representation of the 5-bit message `C8` is `11001`.

## E.1 Examples Of The Short Messages Test for SHA-512

<u>Example 1:</u>

```
Length:   5

M:        b0

MD:       d4ee29a9 e9098544 6b913cf1 d1376c83 6f4be2c1
          cf3cada0 720a6bf4 857d886a 7ecb3c4e 4c0fa8c7
          f95214e4 1dc1b0d2 1b22a84c c03bf8ce 4845f34d
          d5bdbad4
```

<u>Example 2:</u>

```
Length:   8

M:        d0

MD:       99922029 38e882e7 3e20f6b6 9e68a0a7 14909042
          3d93c81b ab3f2167 8d4aceee e50e4e8c afada4c8
          5a54ea83 06826c4a d6e74cec e9631bfa 8a549b4a
          b3fbba15
```

<u>Example 3:</u>

```
Length:   123

M:        08ecb52e bae1f742 2db62bcd 54267080

MD:       ed8dc78e 8b01b697 50053dbb 7a0a9eda 0fb9e9d2
          92b1ed71 5e80a7fe 290a4e16 664fd913 e8585440
          0c5af05e 6dad316b 7359b43e 64f8bec3 c1f23711
          9986bbb6
```

<u>Example 4:</u>

```
Length:   128

M:        8d4e3c0e 38891914 91816e9d 98bff0a0

MD:       cb0b67a4 b8712cd7 3c9aabc0 b199e926 9b20844a
          fb75acbd d1c153c9 828924c3 ddedaafe 669c5fdd
```

```
0bc66f63 0f677398 8213eb1b 16f517ad 0de4b2f0
c95c90f8
```

## E.2 Examples Of The Selected Long Messages Test for SHA-512

Example 1:

Length: 1123

M:
```
3addec85 593216d1 619aa02d 9756970b fc70ace2
744f7c6b 27881510 28f7b6a2 550fd74a 7e6e69c2
c9b45fc4 54966dc3 1d2e10da 1f95ce02 beb4bf87
65574cbd 6e8337ef 420adc98 c15cb6d5 e4a0241b
a0046d25 0e510231 cac2046c 991606ab 4ee4145b
ee2ff4bb 123aab49 8d9d4479 4f99ccad 89a9a162
1259eda7 0a5b6dd4 bdd87778 c9043b93 84f54906 80
```

MD:
```
32ba76fc 30eaa020 8aeb50ff b5af1864 fdbf1790
2a4dc0a6 82c61fce a6d92b78 3267b210 80301837
f59de79c 6b337db2 526f8a0a 510e5e53 cafed435
5fe7c2f1
```

Example 2:

Length: 1816

M:
```
a55f20c4 11aad132 807a502d 65824e31 a2305432
aa3d06d3 e282a8d8 4e0de1de 6974bf49 5469fc7f
338f8054 d58c26c4 9360c3e8 7af56523 acf6d89d
03e56ff2 f868002b c3e431ed c44df2f0 223d4bb3
b243586e 1a7d9249 36694fcb baf88d95 19e4eb50
a644f8e4 f95eb0ea 95bc4465 c8821aac d2fe15ab
4981164b bb6dc32f 969087a1 45b0d9cc 9c67c22b
76329941 9cc4128b e9a077b3 ace63406 4e6d9928
3513dc06 e7515d0d 73132e9a 0dc6d3b1 f8b246f1
a98a3fc7 2941b1e3 bb2098e8 bf16f268 d64f0b0f
4707fe1e a1a1791b a2f3c0c7 58e5f551 863a96c9
49ad47d7 fb40d2
```

MD:
```
c665befb 36da189d 78822d10 528cbf3b 12b3eef7
26039909 c1a16a27 0d487193 77966b95 7a878e72
0584779a 62825c18 da26415e 49a7176a 894e7510
fd1451f5
```

## E.3 Examples of The Pseudorandomly Generated Messages Test for SHA-512

A sample seed for this routine is the following string:

```
Seed:     473ff1b9 b3ffdfa1 26699ac7 ef9e8e78 77730958
          24c64255 7c1399d9 8e422044 8dc35b99 bfdd4477
          9543924c 1ce93bc5 94153889 5db98826 1b00774b
          12272039
```

The first few messages, $M_i$, and hashes, $Md_i$, contained in the inner loop of the routine in Figure 1 are:

$M_{i=3}$:

```
473ff1b9 b3ffdfa1 26699ac7 ef9e8e78 77730958
24c64255 7c1399d9 8e422044 8dc35b99 bfdd4477
9543924c 1ce93bc5 94153889 5db98826 1b00774b
12272039 473ff1b9 b3ffdfa1 26699ac7 ef9e8e78
77730958 24c64255 7c1399d9 8e422044 8dc35b99
bfdd4477 9543924c 1ce93bc5 94153889 5db98826

1b00774b 12272039 473ff1b9 b3ffdfa1 26699ac7
ef9e8e78 77730958 24c64255 7c1399d9 8e422044
8dc35b99 bfdd4477 9543924c 1ce93bc5 94153889
5db98826 1b00774b 12272039
```

$MD_{i=3}$:

```
ab96e447 e4a23028 35b06c3e c33da499 72bfef0b
8678068b c5585945 2292eaaf cc0be757 2c2d14a5
2591fa92 357e5897 0f39ebfc d25877ae 1ba5159e
b251d029
```

$M_{i=4}$:

```
473ff1b9 b3ffdfa1 26699ac7 ef9e8e78 77730958
24c64255 7c1399d9 8e422044 8dc35b99 bfdd4477
9543924c 1ce93bc5 94153889 5db98826 1b00774b
12272039 473ff1b9 b3ffdfa1 26699ac7 ef9e8e78
77730958 24c64255 7c1399d9 8e422044 8dc35b99
bfdd4477 9543924c 1ce93bc5 94153889 5db98826
1b00774b 12272039 ab96e447 e4a23028 35b06c3e
c33da499 72bfef0b 8678068b c5585945 2292eaaf
cc0be757 2c2d14a5 2591fa92 357e5897 0f39ebfc
d25877ae 1ba5159e b251d029
```

$MD_{i=4}$:

```
94b957d0 fa320e59 38fa342a 753b16ae 74a6f3f1
e27e2260 40ab7ce1 1a3ea232 413539d5 dc08a6d0
74673f63 975628d1 30da35fd fce322a8 fe3f63de
44532669
```

$M_{i=5}$:

```
473ff1b9 b3ffdfa1 26699ac7 ef9e8e78 77730958
24c64255 7c1399d9 8e422044 8dc35b99 bfdd4477
9543924c 1ce93bc5 94153889 5db98826 1b00774b
```

```
12272039 ab96e447 e4a23028 35b06c3e c33da499
72bfef0b 8678068b c5585945 2292eaaf cc0be757
2c2d14a5 2591fa92 357e5897 0f39ebfc d25877ae
1ba5159e b251d029 94b957d0 fa320e59 38fa342a
753b16ae 74a6f3f1 e27e2260 40ab7ce1 1a3ea232
413539d5 dc08a6d0 74673f63 975628d1 30da35fd
fce322a8 fe3f63de 44532669
```

$MD_{i=5}$:

```
8d6751b5 554672e2 3219132d ad6bb041 b650bea9
35012c41 f488e847 20a1310e ad9133a1 6109627c
fe9895f2 4d5fee88 4aa754a3 4c6549e8 d17bd2c6
83261b43
```

$M_{i=6}$:

```
ab96e447 e4a23028 35b06c3e c33da499 72bfef0b
8678068b c5585945 2292eaaf cc0be757 2c2d14a5
2591fa92 357e5897 0f39ebfc d25877ae 1ba5159e
b251d029 94b957d0 fa320e59 38fa342a 753b16ae
74a6f3f1 e27e2260 40ab7ce1 1a3ea232 413539d5
dc08a6d0 74673f63 975628d1 30da35fd fce322a8
fe3f63de 44532669 8d6751b5 554672e2 3219132d
ad6bb041 b650bea9 35012c41 f488e847 20a1310e
ad9133a1 6109627c fe9895f2 4d5fee88 4aa754a3
4c6549e8 d17bd2c6 83261b43
```

$MD_{i=6}$:

```
69a635e9 14950e8e e15c27db d833cb87 a4166951
5049938a 3acf7ef3 a8c6c5ec e13300b1 0bb02b8d
fcc7d0bf 1b504d3a 45cc5623 f07d938d 1aa9902e
beae48ec
```

The first few message digests, $MD_j$, that are the output of the routine found in Figure 1 are:

$MD_{j=0}$:

```
2fbb1e7e 00f746ba 514fbc8c 421f3679 2ec0e11f
f5efc378 e1ab0c07 9aa5f0f6 6a1e3edb aeb4f998
4be14437 123038a4 52004a55 768c1fd8 eed49e4a
21bedcd0
```

$MD_{j=1}$:

```
25cbe5a4 f2c7b1d7 ef070117 05d50c62 c5000594
243eafd1 241fc9f3 d22b5818 4ae2fee3 8e171cf8
129e2945 9c9bc2ef 461af570 8887315f 15419d8d
17fe7949
```

$MD_{j=2}$:

```
5b8b1f26 87555ce2 d7182b92 e5c3f6c3 6547da1c
13dbb9ea 4f73ea4c bbaf8941 1527906d 35b1b06c
1b6a8007 d05ec66d f0a40606 6829eab6 18bde397
6515aafc
```

$MD_{j=3}$:

```
46e36b00 7d19876c db0b29ad 074fe3c0 8cdd174d
42169d6a be5a1414 b6e79707 df58776a 98091cf4
31854147 bb6d3c66 d43bfbc1 08fd715b de6aa127
c2b0e79f
```

# Appendix F  Sample Data files

The following subsections contain sample input and output files associated with the SHAVS. For sake of simplicity, abbreviated samples of these files are only provided for the SHA-1 algorithm. The files for other algorithms would by identical, with the exception that input data length may vary and the message digest produced would vary.

## F.1 Sample *REQUEST* Files

### F.1.1 Sample SHA1ShortMsg.req

```
#  CAVS 2.2
#  "SHA-1 ShortMsg" information for "Demo Product"
#  SHA-1 tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Len = 0
Msg =

Len = 1
Msg = 00

Len = 2
Msg = 40

Len = 3
Msg = c0

Len = 4
Msg = 20

Len = 5
Msg = 98

Len = 6
Msg = cc

Len = 7
Msg = 38

Len = 8
Msg = 5e

Len = 9
Msg = a200

Len = 10
Msg = 33c0
.
```

```
.
.

Len = 510
Msg =
5cef7fe793576809d8c889840c79d1dbbc8e63f63cdf460cf5fa5ec7ff6b9ba3f33b7098fab687
f82a71b39a138c26389435e416f46d77e42d0fe4e68b0edf50

Len = 511
Msg =
306bd95ece33d3b53cd3797625d678a841a4e226d7fcd3aa809e552b2d4881696be4e7c6e60030
572c7e023bb48d242b1fd8e353bdfdb46053401adb1c81cab4

Len = 512
Msg =
9888d0a161f7648325af72f63295daab867eafbce4f2e99027305afed771e4b5fa71e8db2c2c01
f4f2e5f3064c6ddefa89a2b8465f36f9c93bb298f6d6ba3735
```

### F.1.2 Sample SHA1LongMsg.req

```
#  CAVS 2.2
#  "SHA-1 LongMsg" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Len = 611
Msg =
65f932995ba4ce2cb1b4a2e71ae70220aacec8962dd4499cbd7c887a94eaaa101ea5aabc529b4e
7e43665a5af2cd03fe678ea6a5005bba3b082204c28b9109f469dac92aaab3aa7c11a1b32ae0

Len = 710
Msg =
e41f78c38e15fb4b00e45df1edc40e3467cdcda351a4c0a0185ac4649e91024377e1c331587a85
86cc0a4dfe29e14004c3536d305f5dee0eeb8c2f216c1b8d27375b239f6458e08980badd6d82e9
ee9e007578c0a3b48288d8

Len = 809
Msg =
b9c05687487749a356666bb10aef23d117c5e8e9d6aef99098884e2b3a8ace55facdc285880b8a
ea61a68e9a58ec78ec7599a122039dc84ce5200a6929c40ecee02a81cc2cb7c28d9a98de3fb203
be74cdc09fc927e9c3c635ee660639dbf49813de79a45a00

Len = 908
Msg =
bee29cf48865b3068418e1e7e142dfbb40cf7a650c508ea80a42318c31932f04bb62b7cd7c0753
9973531d6d9f18314813ec5e054d2b2182c43bd302c01421d06effbd26349d534c422cbf046a06
7d26316a6088267f872072968e6db1b00112004016c557329b671a2b2d74ea13ab9ec1d0

Len = 1007
Msg =
1c9f11faf2a0c6f7e932e650f31191b3d9727b1980354a9c6a623f887fd319b40e2e36130f17fc
```

```
86aba377f4fe0f439d1bb2adab5030150aacf1977abfd1eae3a11987e836af0ebeaf89e63f5467
243a81c2af0eefffa17735e9f0a78f5e4d99dd484ca2efd0c649ffce7c587f83fc33168ac969e5
0cb6890e34933777e2

Len = 1106
Msg =
ab37a9811dcc690394b31135bf2deb09595f9e5d58af007d68712bea97c3d35a52b5d7ff90ae15
0c4d0b83763a087cf7b3e45759f1403ef181c93d6af4169ac4d9d3659be8204fad8034c0975446
23df61ad853723465e000816ae0e25304cab27d97bde8debbfed1577ef2074ae8ac84a024e8055
8807b3e5a1a65e90d99217260f434fe8d70cd4f41c00

Len = 1205
Msg =
2384405f3d67496fcfa8e0e5fdb052299bd043a3883ea1d93d5ad6d7a53642d43056e81223b467
0632072db95045e76ec6673b85826cba9fb2a01003921e7e4182948b155c46f68d0456ee6e4142
85d5cbae631aa564b7f9ac9974f47172b9f344c6902e9126dc4c701a8dcc77ae16a5d22d212996
2d812f53cb6871e938da339c7561266a5a038b9bdd719b9b918127056dde7c00cff8

Len = 1304
Msg =
f78f92141bcd170ae89b4fba15a1d59f3fd84d223c9251bdacbbae61d05ed115a06a7ce117b7be
ead24421ded9c32592bd57edeae39c39fa1fe8946a84d0cf1f7beead1713e2e0959897347f67c8
0b0400c209815d6b10a683836fd5562a56cab1a28e81b6576654631cf16566b86e3b33a108b053
07c00aff14a768ed7350606a0f85e6a91d396f5b5cbe577f9b38807c7d523d6d792f6ebc24a4ec
f2b3a427cdbbfb
.
.
.
Len = 51200
Msg = ...
```

The message in the 51200 entry is omitted here for brevity.

### F.1.3 Sample SHA1Monte.req

```
#  CAVS 2.2
#  "SHA-1 Monte" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Seed = d0569cb3665a8a43eb6ea23d75a3c4d2054a0d7d
```

## F.2 Sample *FAX* Files

### F.2.1 Sample SHA1ShortMsg.fax

```
#  CAVS 2.2
#  "SHA-1 ShortMsg" information for "Demo Product"
#  SHA-1 tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Len = 0
Msg =
MD = da39a3ee5e6b4b0d3255bfef95601890afd80709

Len = 1
Msg = 00
MD = bb6b3e18f0115b57925241676f5b1ae88747b08a

Len = 2
Msg = 40
MD = ec6b39952e1a3ec3ab3507185cf756181c84bbe2

Len = 3
Msg = c0
MD = 6f3b55b9054d756109c4c2e7162970783fd38683

Len = 4
Msg = 20
MD = 91c3d1038358bb4e453e2e67946ef40a32b8d102

Len = 5
Msg = 98
MD = 29826b003b906e660eff4027ce98af3531ac75ba

Len = 6
Msg = cc
MD = 1ac92d2abc0d6ea3c096e80e3e1252de01747bc0

Len = 7
Msg = 38
MD = 64191fbdd73e57f30aec609c6f3258524121604f

Len = 8
Msg = 5e
MD = 5e6f80a34a9798cafc6a5db96cc57ba4c4db59c2

Len = 9
Msg = a200
MD = 658506c6238be94b4a259921ea11a04c0064e5a4

Len = 10
Msg = 33c0
MD = b577bb4bb91ca83d464a1562c343533a7fe50672
.
.
.
Len = 511
Msg =
306bd95ece33d3b53cd3797625d678a841a4e226d7fcd3aa809e552b2d4881696be4e7c6e60030
572c7e023bb48d242b1fd8e353bdfdb46053401adb1c81cab4
```

```
MD = f5151d5f4bb68b7b878eb5cdb16e8d325dbb5cf2

Len = 512
Msg =
9888d0a161f7648325af72f63295daab867eafbce4f2e99027305afed771e4b5fa71e8db2c2c01
f4f2e5f3064c6ddefa89a2b8465f36f9c93bb298f6d6ba3735
MD = 004f60b04a2e51be10924ac96a69a9f20c47e6ce
```

### F.2.2 Sample SHA1LongMsg.fax

```
#  CAVS 2.2
#  "SHA-1 LongMsg" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Len = 611
Msg =
65f932995ba4ce2cb1b4a2e71ae70220aacec8962dd4499cbd7c887a94eaaa101ea5aabc529b4e
7e43665a5af2cd03fe678ea6a5005bba3b082204c28b9109f469dac92aaab3aa7c11a1b32ae0
MD = 8c5b2a5ddae5a97fc7f9d85661c672adbf7933d4

Len = 710
Msg =
e41f78c38e15fb4b00e45df1edc40e3467cdcda351a4c0a0185ac4649e91024377e1c331587a85
86cc0a4dfe29e14004c3536d305f5dee0eeb8c2f216c1b8d27375b239f6458e08980badd6d82e9
ee9e007578c0a3b48288d8
MD = 4fddf903c1843162c19a6114eacf407ab9be9de5

Len = 809
Msg =
b9c05687487749a356666bb10aef23d117c5e8e9d6aef99098884e2b3a8ace55facdc285880b8a
ea61a68e9a58ec78ec7599a122039dc84ce5200a6929c40ecee02a81cc2cb7c28d9a98de3fb203
be74cdc09fc927e9c3c635ee660639dbf49813de79a45a00
MD = 1639095becdd1a6056e0ff2e68730fc4d77dde5a

Len = 908
Msg =
bee29cf48865b3068418e1e7e142dfbb40cf7a650c508ea80a42318c31932f04bb62b7cd7c0753
9973531d6d9f18314813ec5e054d2b2182c43bd302c01421d06effbd26349d534c422cbf046a06
7d26316a6088267f872072968e6db1b00112004016c557329b671a2b2d74ea13ab9ec1d0
MD = e0fd0edfc72a87de298ad669eca781e21c2f4b69

Len = 1007
Msg =
1c9f11faf2a0c6f7e932e650f31191b3d9727b1980354a9c6a623f887fd319b40e2e36130f17fc
86aba377f4fe0f439d1bb2adab5030150aacf1977abfd1eae3a11987e836af0ebeaf89e63f5467
243a81c2af0eefffa17735e9f0a78f5e4d99dd484ca2efd0c649ffce7c587f83fc33168ac969e5
0cb6890e34933777e2
MD = cc7ac764006c94bbd7e723b59e4faf11c400538f

Len = 1106
```

```
Msg =
ab37a9811dcc690394b31135bf2deb09595f9e5d58af007d68712bea97c3d35a52b5d7ff90ae15
0c4d0b83763a087cf7b3e45759f1403ef181c93d6af4169ac4d9d3659be8204fad8034c0975446
23df61ad853723465e000816ae0e25304cab27d97bde8debbfed1577ef2074ae8ac84a024e8055
8807b3e5a1a65e90d99217260f434fe8d70cd4f41c00
MD = 617b0883711762c604cfe17d65031925a6890cf4

Len = 1205
Msg =
2384405f3d67496fcfa8e0e5fdb052299bd043a3883ea1d93d5ad6d7a53642d43056e81223b467
0632072db95045e76ec6673b85826cba9fb2a01003921e7e4182948b155c46f68d0456ee6e4142
85d5cbae631aa564b7f9ac9974f47172b9f344c6902e9126dc4c701a8dcc77ae16a5d22d212996
2d812f53cb6871e938da339c7561266a5a038b9bdd719b9b918127056dde7c00cff8
MD = 26606fc972b80dea994126c925d18fef688f5569

Len = 1304
Msg =
f78f92141bcd170ae89b4fba15a1d59f3fd84d223c9251bdacbbae61d05ed115a06a7ce117b7be
ead24421ded9c32592bd57edeae39c39fa1fe8946a84d0cf1f7beead1713e2e0959897347f67c8
0b0400c209815d6b10a683836fd5562a56cab1a28e81b6576654631cf16566b86e3b33a108b053
07c00aff14a768ed7350606a0f85e6a91d396f5b5cbe577f9b38807c7d523d6d792f6ebc24a4ec
f2b3a427cdbbfb
MD = cb0082c8f197d260991ba6a460e76e202bad27b3
.
.
.
Len = 51200
Msg = …
MD = 21cb7972cd6758fc716743d01fab7c947e168e1f
```

The message in the 51200 entry is omitted here for brevity.

### F.2.3 Sample SHA1Monte.fax

```
#  CAVS 2.2
#  "SHA-1 Monte" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Seed = d0569cb3665a8a43eb6ea23d75a3c4d2054a0d7d

COUNT = 0
MD = e216836819477c7f78e0d843fe4ff1b6d6c14cd4

COUNT = 1
MD = a2dbc7a5b1c6c0a8bcb7aaa41252a6a7d0690dbc

COUNT = 2
MD = db1f9050bb863dfef4ce37186044e2eeb17ee013

COUNT = 3
```

```
MD = 127fdedf43d372a51d5747c48fbffe38ef6cdf7b
.
.
.
COUNT = 98
MD = 1fa936c81d44366c9592a618d140097d4d0555e4

COUNT = 99
MD = 29fc313684e1735f15dc0bc984064fb081dab588
```

## F.3 Sample *SAMPLE* Files

### F.3.1 Sample SHA1ShortMsg.sam

```
#  CAVS 2.2
#  "SHA-1 ShortMsg" information for "Demo Product"
#  SHA-1 tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

Len = 0
Msg =
MD = ?

Len = 1
Msg = 00
MD = ?

Len = 2
Msg = 40
MD = ?

Len = 3
Msg = c0
MD = ?

Len = 4
Msg = 20
MD = ?

Len = 5
Msg = 98
MD = ?

Len = 6
Msg = cc
MD = ?

Len = 7
Msg = 38
MD = ?

Len = 8
```

```
Msg = 5e
MD = ?

Len = 9
Msg = a200
MD = ?

Len = 10
Msg = 33c0
MD = ?
.
.
.

Len = 511
Msg =
306bd95ece33d3b53cd3797625d678a841a4e226d7fcd3aa809e552b2d4881696be4e7c6e60030
572c7e023bb48d242b1fd8e353bdfdb46053401adb1c81cab4
MD = ?

Len = 512
Msg =
9888d0a161f7648325af72f63295daab867eafbce4f2e99027305afed771e4b5fa71e8db2c2c01
f4f2e5f3064c6ddefa89a2b8465f36f9c93bb298f6d6ba3735
MD = ?
```

### F.3.2 Sample SHA1LongMsg.sam

```
#  CAVS 2.2
#  "SHA-1 LongMsg" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

Len = 611
Msg =
65f932995ba4ce2cb1b4a2e71ae70220aacec8962dd4499cbd7c887a94eaaa101ea5aabc529b4e
7e43665a5af2cd03fe678ea6a5005bba3b082204c28b9109f469dac92aaab3aa7c11a1b32ae0
MD = ?

Len = 710
Msg =
e41f78c38e15fb4b00e45df1edc40e3467cdcda351a4c0a0185ac4649e91024377e1c331587a85
86cc0a4dfe29e14004c3536d305f5dee0eeb8c2f216c1b8d27375b239f6458e08980badd6d82e9
ee9e007578c0a3b48288d8
MD = ?

Len = 809
Msg =
b9c05687487749a356666bb10aef23d117c5e8e9d6aef99098884e2b3a8ace55facdc285880b8a
ea61a68e9a58ec78ec7599a122039dc84ce5200a6929c40ecee02a81cc2cb7c28d9a98de3fb203
be74cdc09fc927e9c3c635ee660639dbf49813de79a45a00
```

```
MD = ?

Len = 908
Msg =
bee29cf48865b3068418e1e7e142dfbb40cf7a650c508ea80a42318c31932f04bb62b7cd7c0753
9973531d6d9f18314813ec5e054d2b2182c43bd302c01421d06effbd26349d534c422cbf046a06
7d26316a6088267f872072968e6db1b00112004016c557329b671a2b2d74ea13ab9ec1d0
MD = ?

Len = 1007
Msg =
1c9f11faf2a0c6f7e932e650f31191b3d9727b1980354a9c6a623f887fd319b40e2e36130f17fc
86aba377f4fe0f439d1bb2adab5030150aacf1977abfd1eae3a11987e836af0ebeaf89e63f5467
243a81c2af0eefffa17735e9f0a78f5e4d99dd484ca2efd0c649ffce7c587f83fc33168ac969e5
0cb6890e34933777e2
MD = ?

Len = 1106
Msg =
ab37a9811dcc690394b31135bf2deb09595f9e5d58af007d68712bea97c3d35a52b5d7ff90ae15
0c4d0b83763a087cf7b3e45759f1403ef181c93d6af4169ac4d9d3659be8204fad8034c0975446
23df61ad853723465e000816ae0e25304cab27d97bde8debbfed1577ef2074ae8ac84a024e8055
8807b3e5a1a65e90d99217260f434fe8d70cd4f41c00
MD = ?

Len = 1205
Msg =
2384405f3d67496fcfa8e0e5fdb052299bd043a3883ea1d93d5ad6d7a53642d43056e81223b467
0632072db95045e76ec6673b85826cba9fb2a01003921e7e4182948b155c46f68d0456ee6e4142
85d5cbae631aa564b7f9ac9974f47172b9f344c6902e9126dc4c701a8dcc77ae16a5d22d212996
2d812f53cb6871e938da339c7561266a5a038b9bdd719b9b918127056dde7c00cff8
MD = ?

Len = 1304
Msg =
f78f92141bcd170ae89b4fba15a1d59f3fd84d223c9251bdacbbae61d05ed115a06a7ce117b7be
ead24421ded9c32592bd57edeae39c39fa1fe8946a84d0cf1f7beead1713e2e0959897347f67c8
0b0400c209815d6b10a683836fd5562a56cab1a28e81b6576654631cf16566b86e3b33a108b053
07c00aff14a768ed7350606a0f85e6a91d396f5b5cbe577f9b38807c7d523d6d792f6ebc24a4ec
f2b3a427cdbbfb
MD = ?
.
.
.
Len = 51200
Msg = …
MD = ?
```

The message in the 51200 entry is omitted here for brevity.

### F.3.3 Sample SHA1Monte.sam

```
#  CAVS 2.2
#  "SHA-1 Monte" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations
#  Generated on Wed Mar 26 09:58:58 2003

[L = 20]

Seed = d0569cb3665a8a43eb6ea23d75a3c4d2054a0d7d
COUNT = 0
MD = e216836819477c7f78e0d843fe4ff1b6d6c14cd4

COUNT = 1
MD = a2dbc7a5b1c6c0a8bcb7aaa41252a6a7d0690dbc

COUNT = 2
MD = db1f9050bb863dfef4ce37186044e2eeb17ee013

COUNT = 3
MD = ?
.
.
.
COUNT = 98
MD = ?

COUNT = 99
MD = ?
```

## F.4 Sample *RESPONSE* Files

### F.4.1 Sample SHA1ShortMsg.rsp

```
#  CAVS 2.2
#  "SHA-1 ShortMsg" information for "Demo Product"
#  SHA-1 tests are configured for BIT oriented implementations

[L = 20]

Len = 0
Msg =
MD = da39a3ee5e6b4b0d3255bfef95601890afd80709

Len = 1
Msg = 00
MD = bb6b3e18f0115b57925241676f5b1ae88747b08a

Len = 2
Msg = 40
MD = ec6b39952e1a3ec3ab3507185cf756181c84bbe2

Len = 3
Msg = c0
MD = 6f3b55b9054d756109c4c2e7162970783fd38683
```

```
Len = 4
Msg = 20
MD = 91c3d1038358bb4e453e2e67946ef40a32b8d102

Len = 5
Msg = 98
MD = 29826b003b906e660eff4027ce98af3531ac75ba

Len = 6
Msg = cc
MD = 1ac92d2abc0d6ea3c096e80e3e1252de01747bc0

Len = 7
Msg = 38
MD = 64191fbdd73e57f30aec609c6f3258524121604f

Len = 8
Msg = 5e
MD = 5e6f80a34a9798cafc6a5db96cc57ba4c4db59c2

Len = 9
Msg = a200
MD = 658506c6238be94b4a259921ea11a04c0064e5a4

Len = 10
Msg = 33c0
MD = b577bb4bb91ca83d464a1562c343533a7fe50672
.
.
.
Len = 511
Msg =
306bd95ece33d3b53cd3797625d678a841a4e226d7fcd3aa809e552b2d4881696be4e7c6e60030
572c7e023bb48d242b1fd8e353bdfdb46053401adb1c81cab4
MD = f5151d5f4bb68b7b878eb5cdb16e8d325dbb5cf2

Len = 512
Msg =
9888d0a161f7648325af72f63295daab867eafbce4f2e99027305afed771e4b5fa71e8db2c2c01
f4f2e5f3064c6ddefa89a2b8465f36f9c93bb298f6d6ba3735
MD = 004f60b04a2e51be10924ac96a69a9f20c47e6ce
```

### F.2.2 Sample SHA1LongMsg.rsp

```
#  CAVS 2.2
#  "SHA-1 LongMsg" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations

[L = 20]

Len = 611
```

```
Msg =
65f932995ba4ce2cb1b4a2e71ae70220aacec8962dd4499cbd7c887a94eaaa101ea5aabc529b4e
7e43665a5af2cd03fe678ea6a5005bba3b082204c28b9109f469dac92aaab3aa7c11a1b32ae0
MD = 8c5b2a5ddae5a97fc7f9d85661c672adbf7933d4

Len = 710
Msg =
e41f78c38e15fb4b00e45df1edc40e3467cdcda351a4c0a0185ac4649e91024377e1c331587a85
86cc0a4dfe29e14004c3536d305f5dee0eeb8c2f216c1b8d27375b239f6458e08980badd6d82e9
ee9e007578c0a3b48288d8
MD = 4fddf903c1843162c19a6114eacf407ab9be9de5

Len = 809
Msg =
b9c05687487749a356666bb10aef23d117c5e8e9d6aef99098884e2b3a8ace55facdc285880b8a
ea61a68e9a58ec78ec7599a122039dc84ce5200a6929c40ecee02a81cc2cb7c28d9a98de3fb203
be74cdc09fc927e9c3c635ee660639dbf49813de79a45a00
MD = 1639095becdd1a6056e0ff2e68730fc4d77dde5a

Len = 908
Msg =
bee29cf48865b3068418e1e7e142dfbb40cf7a650c508ea80a42318c31932f04bb62b7cd7c0753
9973531d6d9f18314813ec5e054d2b2182c43bd302c01421d06effbd26349d534c422cbf046a06
7d26316a6088267f872072968e6db1b00112004016c557329b671a2b2d74ea13ab9ec1d0
MD = e0fd0edfc72a87de298ad669eca781e21c2f4b69

Len = 1007
Msg =
1c9f11faf2a0c6f7e932e650f31191b3d9727b1980354a9c6a623f887fd319b40e2e36130f17fc
86aba377f4fe0f439d1bb2adab5030150aacf1977abfd1eae3a11987e836af0ebeaf89e63f5467
243a81c2af0eefffa17735e9f0a78f5e4d99dd484ca2efd0c649ffce7c587f83fc33168ac969e5
0cb6890e34933777e2
MD = cc7ac764006c94bbd7e723b59e4faf11c400538f

Len = 1106
Msg =
ab37a9811dcc690394b31135bf2deb09595f9e5d58af007d68712bea97c3d35a52b5d7ff90ae15
0c4d0b83763a087cf7b3e45759f1403ef181c93d6af4169ac4d9d3659be8204fad8034c0975446
23df61ad853723465e000816ae0e25304cab27d97bde8debbfed1577ef2074ae8ac84a024e8055
8807b3e5a1a65e90d99217260f434fe8d70cd4f41c00
MD = 617b0883711762c604cfe17d65031925a6890cf4

Len = 1205
Msg =
2384405f3d67496fcfa8e0e5fdb052299bd043a3883ea1d93d5ad6d7a53642d43056e81223b467
0632072db95045e76ec6673b85826cba9fb2a01003921e7e4182948b155c46f68d0456ee6e4142
85d5cbae631aa564b7f9ac9974f47172b9f344c6902e9126dc4c701a8dcc77ae16a5d22d212996
2d812f53cb6871e938da339c7561266a5a038b9bdd719b9b918127056dde7c00cff8
MD = 26606fc972b80dea994126c925d18fef688f5569

Len = 1304
Msg =
f78f92141bcd170ae89b4fba15a1d59f3fd84d223c9251bdacbbae61d05ed115a06a7ce117b7be
ead24421ded9c32592bd57edeae39c39fa1fe8946a84d0cf1f7beead1713e2e0959897347f67c8
```

```
0b0400c209815d6b10a683836fd5562a56cab1a28e81b6576654631cf16566b86e3b33a108b053
07c00aff14a768ed7350606a0f85e6a91d396f5b5cbe577f9b38807c7d523d6d792f6ebc24a4ec
f2b3a427cdbbfb
MD = cb0082c8f197d260991ba6a460e76e202bad27b3
.
.
.
Len = 51200
Msg = …
MD = 21cb7972cd6758fc716743d01fab7c947e168e1f
```

The message in the 51200 entry is omitted here for brevity.

### F.2.3 Sample SHA1Monte.rsp

```
#  CAVS 2.2
#  "SHA-1 Monte" information for "Demo Product"
#  SHA tests are configured for BIT oriented implementations

[L = 20]

Seed = d0569cb3665a8a43eb6ea23d75a3c4d2054a0d7d

COUNT = 0
MD = e216836819477c7f78e0d843fe4ff1b6d6c14cd4

COUNT = 1
MD = a2dbc7a5b1c6c0a8bcb7aaa41252a6a7d0690dbc

COUNT = 2
MD = db1f9050bb863dfef4ce37186044e2eeb17ee013

COUNT = 3
MD = 127fdedf43d372a51d5747c48fbffe38ef6cdf7b
.
.
.
COUNT = 98
MD = 1fa936c81d44366c9592a618d140097d4d0555e4

COUNT = 99
MD = 29fc313684e1735f15dc0bc984064fb081dab588
```

## Appendix G References

[1] *Digital Signature Standard (DSS)*, FIPS Publication 186-2 (+Change Notice), National Institute of Standards and Technology, January 2000.

[2] *Security Requirements for Cryptographic Modules*, FIPS Publication 140-2, National Institute of Standards and Technology, May 2001.