The Secure Hash Algorithm 3 Validation System (SHA3VS)

April 7, 2016
Original: January 29, 2016

Sharon S. Keller

Lawrence E. Bassham III

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

Update Log

- 04-07-16-Section 6 – Added detail that the dash(-) in the algorithm name is changed to an underline(_) in the filenames.

- Also realigned figures with text.

- 04-06-16- Section 6 – Added explanation of naming convention of filenames:

    - The algorithm names are used in the <Alg> portion of the request (req), fax, and response (rsp) file names generated for the validation tests. For example, <Alg>ShortMsg.req for SHA3-224 would be SHA3-224ShortMsg.req. For SHAKE128, the filename would be SHAKE128ShortMsg.req.

- Also removed inconsistencies in filename format throughout document.

- 03-09-16 – Section 6.3.1.2 – Corrected the maximum length of the messages used in the Short Message Test for Byte-Oriented Input Messages. This is the statement that was modified with the correction highlighted:

    - Therefore, for testing SHAKE128, (168*2)+1) = 337 unpredictable messages will be generated with lengths from 0 to ~~337~~ 336 bytes (omitting 0 if zero length is not supported) and SHAKE256, (136*2)+1) = 273 unpredictable messages will be generated with lengths from 0 to ~~273~~ 272 bytes (omitting 0 if zero length is not supported).

1 Introduction

This document, *The Secure Hash Algorithm-3 Validation System (SHA3VS)* specifies the procedures involved in validating algorithm implementations for the conformance to FIPS 202 *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* [1].

FIPS 202 specifies a new family of functions called SHA-3. The SHA-3 family consists of four cryptographic hash functions and two closely related Extendable-Output Functions (XOFs).

The four SHA-3 hash functions are named SHA3-224, SHA3-256, SHA3-384, and SHA3-512; in each case, the numeric suffix after the dash indicates the fixed length of the digest, e.g., SHA3-256 produces 256-bit message digests.

The two SHA-3 XOFs are named SHAKE128 and SHAKE256. The suffixes "128" and "256" indicate the security strengths that these two functions can **generally** support. (Security considerations of XOFs are discussed in FIPS 202, Section A.2.) The XOFs output a variable length output.

This document provides the basic design and configuration of the SHA3VS. It includes the specifications for the validation test suite for the SHA-3 family of functions providing automated testing on Implementations Under Test (IUTs).

2 Scope

This document specifies the validation tests required to validate IUTs for conformance to the FIPS 202. When applied to IUTs that implement SHA-3 hash functions and/or SHA-3 XOFs, the SHA3VS provides testing to determine the correctness of the algorithm implementation. In addition to determining conformance, the SHA3VS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the function(s) being tested in the implementation.

The requirements and administrative procedures specific to those seeking formal validation of an implementation of the SHA-3 hash functions and/or SHA-3 XOFs are presented. The requirements described include the specific protocols for communication between the IUT and the SHA3VS, the types of tests that the IUT must pass for formal validation, and general instructions for accessing and interfacing with the SHA3VS.

While the specification for the SHA-3 hash functions and the SHA-3 XOFs allow for unlimited input message lengths, these tests only test messages up to a limited size of approximately 100,000 bits. Likewise, the SHA-3 XOFs allow for unlimited output data lengths. But for testing purposes, the range of 16 bits up to 2^{16} bits is tested. These is adequate for detecting algorithmic and implementation errors.

3 Conformance

The successful completion of the applicable tests contained within the SHA3VS is required to claim conformance to the FIPS 202 SHA-3 hash and/or XOF functions. Testing for the cryptographic module in which the various algorithms specified in FIPS 202 are implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules* [2].

4 Definitions, Symbols, and Abbreviations

4.1 Definitions

DEFINITION	**MEANING**
CAVS	Cryptographic Algorithm Validation System
CST laboratory	Cryptographic Security Testing laboratory that operates the SHA3VS
IUT	Implementation Under Test
KECCAK	The family of all sponge functions with a KECCAK-*f* permutation as the underlying function and multi-rate padding as the padding rule, KECCAK was originally specified in [8] (see 202).
SHA-3	The Secure Hash Algorithm-3 specified in FIPS 202 [1]
SHA3VS	The Secure Hash Algorithm-3 Validation System
SHAKE	Secure Hash Algorithm KECCAK.

4.2 Symbols

SYMBOL	**MEANING**
b	The width of a KECCAK-*p* permutation in bits (block size)
c	The capacity of a sponge function
d	The length of a digest of a hash function or the requested length of the output of an XOF, in bits.
i	Increment amount
Len	Length of input message

MD	Message digest output from a SHA-3 function
minoutlen	The minimum output length tested for the IUT. For testing purposes, the minimum output length is greater than 16 bits. If the test is a byte-oriented implementation, the minimum output length is rounded up to the nearest multiple of 8 and is called *minoutbyte*.
maxoutlen	The maximum output length tested for the IUT. For testing purposes, the maximum output length that can be tested is 2^16 bits. If the test is a byte-oriented implementation, the maximum output length is rounded down to the nearest multiple of 8 and is called *maxoutbyte*.
Msg	The input message to a SHA-3 function
n	Length in bits of *Seed*
Output	Output data generated from SHA-3 XOF
Rightmost_Output_bits	Rightmost 16 bits of output to be used as the next output length
Outputlen	Length of *Output* from a SHA-3 XOF. Note the length will be the minimum of the *sec_level* and the *maxoutlen*, i.e., min (*sec_level*, *maxoutlen*).
Range	The number of output lengths allowed by a SHAKE implementation; (maximum output length (rounded down to the nearest multiple of 8) – minimum output length (rounded up to the nearest multiple of 8) + 1);
rate	The rate of a sponge function
sec_level	The security strength of the cryptographic primitive in bits.
Seed	Randomly generated value to be used as the initial Message

5 Design Philosophy of the Secure Hash Algorithm Validation System

The SHA3VS is designed to test conformance to the FIPS 202 SHA-3 and/or XOF functions 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 SHA3VS has the following design philosophy:

1. The SHA3VS is designed to allow the testing of an IUT at locations remote to the SHA3VS. The SHA3VS and the IUT communicate data via *REQUEST* and *RESPONSE* files.

2. The testing performed within the SHA3VS 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 SHA3VS Validation Tests

The SHA3VS specifies tests for the following algorithms:

- SHA-3 Hash Algorithms: SHA3-224, SHA3-256, SHA3-384, and SHA3-512;
- SHA-3 XOFs: SHAKE128 and SHAKE256.

The algorithm names are used in the <Alg> portion of the request (req), fax, and response (rsp) file names generated for the validation tests BUT REPLACE THE DASH (-) WITH AN UNDERSCORE (_). For example, <Alg>ShortMsg.req for SHA3-224 would be SHA3_224ShortMsg.req. For SHAKE128, the filename would be SHAKE128ShortMsg.req.

For the SHA-3 hash functions, the test suite includes:

- The Short Messages Test;
- The Long Messages Test; and
- The Pseudorandomly Generated Messages (Monte Carlo) Test.

For the SHA-3 XOFs, the test suite includes:

- The Short Messages Test;
- The Long Messages Test;
- The Pseudorandomly Generated Messages (Monte Carlo) Test; and
- The Variable-Length Output Test.

The Short Messages Test, the Long Messages Test, and the Pseudorandomly Generated Messages (Monte Carlo) Test have been designed for both the SHA3 and the XOF functions to exercise the ability of an IUT to accept different length input messages. The Monte Carlo test pseudo exhaustively exercises the underlying function being tested to assure the inputs result in the correct outputs. For the above three tests, when applied to the SHA-3 hash functions, the

output of these validation tests is dictated by the hash size being tested. For the SHA-3 XOFs, the output for these three tests is set to a fixed value for testing purposes only.

The Variable Output Test has been added for the XOF functions to test the ability of an IUT to correctly compute and output variable length data. This test is not needed for the SHA-3 hash functions since the output size is dictated by the hash size.

6.1. Configuration Information

To initiate the validation process of the SHA3VS, a vendor submits an application to an accredited laboratory requesting the validation of their implementation of FIPS 202. 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 SHA3VS 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(s) 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);
7. If a SHA-3 hash function is being tested:
    a. SHA-3 functions supported: SHA3-224, SHA3-256, SHA3-384, SHA3-512
    b. Indication of whether the IUT handles byte-oriented input messages only. (Default is IUT handles bit-oriented input messages.)
    c. Indication of whether the IUT handles zero length input messages. (Default is IUT does handle zero length input messages.)
8. If a SHA-3 XOF is being tested:
    a. SHAKE algorithms supported : SHAKE128, SHAKE256

b. The minimum and maximum output lengths tested for the IUT. Note, for testing purposes, 16 bits <= valid output lengths <= 2^{16} bits.

c. Indication of whether the IUT handles byte-oriented input messages only. (Default is IUT handles bit-oriented input messages.)

d. Indication of whether the IUT handles zero length input messages. (Default is IUT does handle zero length input messages.)

e. Indication of whether the IUT supports output values of byte-oriented lengths only. (Default is IUT supports output values of bit-oriented lengths.)

6.2 Validation Test Suite for SHA-3 Hash Algorithms

The validation test suite for the Secure Hash Algorithm-3 consists of three types of tests: the Short Messages Test, the Long Messages Test, and the Pseudorandomly Generated Messages (Monte Carlo) Test. The SHA3VS 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.

Below is a chart for the SHA-3 algorithm indicating the allowable hash sizes and their associated capacity and rates (input bits processed per invocation of the underlying function) in both bits and bytes. This information will be used in the SHA-3 function validation tests:

d – SHA-3 digest length	*c* - Capacity (*c=2d*)	*rate* - (1600-*c*) (bits/bytes)
SHA3-224	448	1152/144
SHA3-256	512	1088/136
SHA3-384	768	832/104
SHA3-512	1024	576/72

An IUT may implement SHA-3 in either of two modes. The first mode is the bit-oriented mode. In this mode the IUT hashes messages of arbitrary length. The second 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. Selecting the proper mode for an implementation will determine how the SHA3VS tests will be performed. The Short Messages Test and the Selected Long Messages Test can be run in either mode. The Pseudorandomly Generated Messages (Monte Carlo) Test is always run in byte-oriented mode.

6.2.1 The Short Messages Test

An implementation of SHA-3 must be able to correctly generate message digests for messages of arbitrary length. The SHA3VS 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.

6.2.1.1 The Short Messages Test for Bit-Oriented Implementations

This test generates *rate*+1 short messages. The number of bits in the input data will vary from 0 to *rate*. Therefore, for testing SHA3-256, 1089 unpredictable messages will be generated with lengths from 0 to 1088 bits (omitting 0 if zero length is not supported).

The SHA3VS:

A. Generates *rate* + 1 messages of length 0 to *rate*. If zero length is not supported by the IUT, the zero length message is omitted.

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

    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The *rate* + 1 datasets containing:
        a. The length of the message *(Len)*
        b. The message (*Msg*).

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

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

    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The *rate* + 1 datasets containing:
        a. The length of the message *(Len)*
        b. The message (*Msg*), and
        c. The message digest (*MD*).

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

The IUT:

A. Generates message digests using the messages supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The *rate* + 1 datasets containing:
    a. The length of the message *(Len)*
    b. The message (*Msg*), and
    c. The message digest (*MD*).

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS 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.

6.2.1.2 The Short Messages Test for Byte-Oriented Implementations

This test generates *rate* (in bytes) +1 short messages. The number of bytes in the input data will vary from 0 to *rate* (in bytes). Therefore, for testing SHA3-256, 137 ((136 bytes) + 1) unpredictable messages will be generated with lengths of 0, 8, 16… 1088 bits (omitting 0 if zero length is not supported).

The SHA3VS:

A. Generates *rate* (in bytes) + 1 messages of length 0, 8, 16… *rate* (in bytes). If zero length is not supported by the IUT, the zero length message is omitted.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The *rate* (in bytes) + 1 datasets containing:
    a. The length of the message *(Len)*
    b. The message (*Msg*).

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

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The *rate* (in bytes) + 1 datasets containing:
    a. The length of the message *(Len)*
    b. The message (*Msg*), and
    c. The message digest (*MD*) of the message.

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

The IUT:

A. Generates message digests using the messages supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The *rate* (in bytes) + 1 datasets containing:
    a. The length of the message *(Len)*
    b. The message (*Msg*), and
    c. The message digest (*MD*) of the message.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS 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 Selected Long Messages Test

An implementation of SHA-3 must be able to correctly generate message digests for messages that span multiple message blocks. The SHA3VS 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.

6.2.2.1 The Selected Long Messages Test for Bit-Oriented Implementations

This test generates 100 long messages ranging in size from *rate*+(*rate*+1) <= *Len* <= *rate*+(100*(*rate*+1)) incrementing by *rate*+1. For example, SHA3-256 has a rate of 1088 bits. Therefore, for testing SHA3-256, 100 unpredictable long messages will be generated with lengths (in bits) of 2177, 3266 … 109988 bits.

The SHA3VS:

A. Generates 100 messages of the lengths specified above.
B. Creates a *REQUEST* file (Filename: <Alg>LongMsg.req) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The sequence of 100 datasets to be hashed containing:
        a. The length of the message *(Len)*
        b. The message (*Msg*).

    Note: The CST laboratory sends the *REQUEST* file to the IUT.
C. Creates a *FAX* file (Filename: <Alg>LongMsg.fax) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The 100 datasets containing:
        a. The length of the message *(Len)*
        b. The message (*Msg*)
        c. The message digest (*MD*) of the message.

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

The IUT:

A. Generates 100 message digests using the messages supplied by the SHA3VS in the *REQUEST* file.
B. Creates a *RESPONSE* file (Filename: <Alg>LongMsg.rsp) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. The 100 datasets containing:

a. The length of the message *(Len)*

b. The message (*Msg*)

c. The message digest (*MD*) of the message.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS 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.

6.2.2.2 The Selected Long Messages Test for Byte-Oriented Implementations

This test generates 100 long messages ranging in size from (*rate*+ (*rate*+8)) <= *len* <= (*rate*+100*(*rate*+8)) incrementing by *rate*+8. For example, SHA3-256 defines a block length of m=1088 bits. For example, SHA3-256 defines a rate of 1088 bits. Therefore, for testing SHAKE-256, unpredictable long messages will be generated with lengths (in bits) of 2184, 3280, 4376… 110688 bits.

The SHA3VS:

A. Generates 100 messages of the length specified above.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The sequence of 100 datasets to be hashed containing:
    a. The length of the message *(Len)*
    b. The message (*Msg*).

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

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The 100 datasets containing:
    a. The length of the message *(Len)*

b. The message (*Msg*)

c. The message digest (*MD*) of the message.

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

The IUT:

A. Generates message digests using the message supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The 100 datasets containing:

    a. The length of the message *(Len)*

    b. The message (*Msg*)

    c. The message digest (*MD*) of the message.

    Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS 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.

6.2.3 The Pseudorandomly Generated Messages (Monte Carlo) Test

The SHA3VS 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 SHA3VS compares each of the recorded 100 message digests with those generated by the IUT.

```
INPUT: A random Seed n bits long
{
     MD0 = Seed;
     for (j=0; j<100; j++) {
          for (i=1; i<1001; i++) {
               Msgi = MDi-1;
               MDi = SHA3(Msgi);
               }
          MD0 = MD1000;
          OUTPUT: MD0
          }
     }
```

Figure 1: Code for Generating Pseudorandom Messages

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

1) 100,000 pseudorandom messages (*Msg)* are generated by using previous message digests (*MD*) 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_0 in Figure 1.

The SHAVS:

A. Generates a seed, *Seed*, of length *n* bits.

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, length of hash, etc.);
   4. The *Seed*.

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

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, length of hash, etc.);
   2. The *Seed*; and
   3. For 100 rounds

      a. *COUNT*

b. The OUTPUT message digests (MD_0), from Figure 1.

Note: The CST laboratory retains the *FAX* file at the SHA3VS.

The IUT:

A. Generates the 100 message digests using the *Seed* supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The *Seed*; and
4. For 100 rounds
    a. *COUNT*
    b. The OUTPUT message digest (*MD*) from Figure 1.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. The SHA3VS 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.

6.3 Validation Test Suite for SHA-3 XOFs

The validation test suite for the Secure Hash Algorithm-3 Extendable-Output Functions, SHAKE128 and SHAKE 256, consists of four types of tests: The Short Messages Test, The Long Messages Test, The Pseudorandomly Generated Messages (Monte Carlo) Test, and The Variable Output Messages Test.

The SHA3VS 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.

Below is a chart for the SHAKE algorithm indicating the allowable security levels and their associated capacity and rates (input bits processed per invocation of the underlying function) in both bits and bytes. This information will be used in the SHAKE function validation tests:

SHAKE algorithm	***sec_level* - Security Level**	***c* - Capacity**	***rate* (bits/bytes)**

SHAKE128	128	256	1344/168
SHAKE256	256	512	1088/136

6.3.1 The Short Messages Test

An implementation of the SHAKE algorithm must be able to correctly generate output for messages of arbitrary length. This test verifies that the IUT can pad the input message correctly. The Short Messages Test tests this property by supplying the IUT with a number of short messages. The Short Messages Test has two versions, one for implementations that support bit-oriented input messages and another for implementations that support byte-oriented input messages.

6.3.1.1 The Short Messages Test for Bit-Oriented Input Message Length Implementations

This test generates (*rate*2)*+1 short messages. The number of bits in the input data will vary from 0 to *rate*2*. Therefore, for testing SHAKE28, (1344*2)+1 = 2689 unpredictable messages will be generated with lengths from 0 to 2689 bits (omitting 0 if zero length is not supported). For testing SHAKE256, (1088*2)+1 = 2177 unpredictable messages will be generated with lengths from 0 to 2177 bits (omitting 0 if zero length is not supported).

The SHA3VS for the XOF's Short Message Test:

A. Generates (*rate*2)* + 1 messages of length 0 to *rate*2*. If zero length is not supported by the IUT, the zero length message is omitted.

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, etc.);
   4. The output length (*Outputlen*) for the data to be generated. For testing purposes, the output length will be set to the minimum of the security level and max output length, i.e., min (*sec_level*, *maxoutlen*). For example, if SHAKE128 is being tested and the maxoutlen is 5000 bits, the output length for this test will be 128 bits.
   5. The (*rate * 2*) + 1 datasets containing:
      a. The length of the input message (*Len*)
      b. The input message (*Msg*).

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

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The output length (*Outputlen*) for the data to be generated.
5. The (*rate*2)* + 1 datasets containing:
    a. The length of the input message (*Len*)
    b. The input message (*Msg*), and
    c. The resulting *Output* of length *Outputlen*.

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

The IUT:

A. Generates output data using the messages supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The (*rate*2)* + 1 datasets containing:
    a. The length of the input message *(Len)*
    b. The input message (*Msg*), and
    c. The *Output* of the required length as specified in the request file.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output.

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

6.3.1.2 The Short Messages Test for Byte-Oriented Input Message Length Implementations

This test generates (*rate* (in bytes) **2)* +1 short messages. The number of bytes in the input data will vary from 0 to *rate*2* (in bytes). Therefore, for testing SHAKE128, (168*2)+1) = 337

unpredictable messages will be generated with lengths from 0 to 336 bytes (omitting 0 if zero length is not supported) and SHAKE256, (136*2)+1) = 273 unpredictable messages will be generated with lengths from 0 to 272 bytes (omitting 0 if zero length is not supported).

The SHA3VS for the XOF's Short Message Test:

A. Generates (*rate* (in bytes) * *2*) + 1 messages of length 0, 8, 16… *rate*2* (in bytes). If the IUT doesn't support zero length messages, the zero length is omitted.

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

    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The output length (*Outputlen*) for the data to be generated. For testing purposes, the output length will be set to the minimum of the security level and max output length, i.e., min (*sec_level*, *maxoutlen*). For example, if SHAKE128 is being tested and the maxoutlen is 5000 bits, the output length for this test will be 128 bits.
    5. The (*rate* (in bytes) * *2*) + 1 datasets containing:
        a. The length of the input message *(Len)*
        b. The input message (*Msg*).

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

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

    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.);
    4. The output length (*Outputlen*) for the data to be generated.
    5. The (*rate* (in bytes) * *2*) + 1 datasets containing:
        a. The length of the input message *(Len)*
        b. The input message (*Msg*), and
        c. The resulting *Output* of length *Outputlen*.

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

The IUT:

A. Generates output data using the messages supplied by the SHA3VS in the *REQUEST* file.

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

    1. The CAVS Version

2. The Product Information (product name, test specifications)
3. The (*rate* (in bytes) * *2*) + 1 datasets containing:
    a. The length of the input message *(Len)*
    b. The input message (*Msg*), and
    c. The *Output* of the required length as specified in the request file.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output.
B. If all output data generated by the IUT are correct, records PASS for this test; otherwise, records FAIL.

6.3.2 The Selected Long Messages Test

An implementation of the SHAKE algorithm must be able to correctly generate output for messages that span multiple message blocks. The SHA3VS tests this property by supplying selected unpredictable long messages to the IUT. The IUT then generates output data for each message. The Selected Long Messages Test has two versions, one for implementations that support bit-oriented input messages and another for implementations that support byte-oriented input messages.

6.3.2.1 The Selected Long Messages Test for Bit-Oriented Input Message Length Implementations

This test generates 100 long messages ranging in size from *rate*+(*rate*+1) <= *len* <= *rate*+(100*(*rate*+1)) incrementing by *rate*+1. For example, SHAKE256 has a rate of 1088 bits. Therefore, for testing SHAKE256, unpredictable long messages will be generated with lengths (in bits) of 2177, 3266 … 109988 bits.

The SHA3VS:

A. Generates 100 messages of the lengths specified above.
B. Creates a *REQUEST* file (Filename: <Alg>LongMsg.req) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.)
    4. The output length (*Outputlen*) for the data to be generated. For testing purposes, the output length will be set to the minimum of the security level and max output length,

i.e., min (*sec_level*, *maxoutlen*). For example, if SHAKE128 is being tested and the maxoutlen is 5000 bits, the output length for this test will be 128 bits.

5. The sequence of 100 datasets containing:
    a. The length of the input message *(Len)*
    b. The input message (*Msg*).

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

C. Creates a *FAX* file (Filename: <Alg>LongMsg.fax) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.)
    4. The output length (*Outputlen*) for the data to be generated
    5. The 100 datasets containing:
        a. The length of the input message *(Len)*
        b. The input message (*Msg*)
        c. The resulting *Output* of length *Outputlen*.

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

The IUT:

A. Generates 100 output data using the messages supplied by the SHA3VS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: <Alg>LongMsg.rsp) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. The 100 datasets containing:
        a. The length of the input message *(Len)*
        b. The input message (*Msg*)
        c. The resulting *Output* of length *Outputlen*.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output data.

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

6.3.2.2 The Selected Long Messages Test for Byte-Oriented Input Message Length Implementations

This test generates 100 long messages ranging in size from (*rate* + (*rate* + 8)) <= *len* <= (*rate* + 100 * (*rate* + 8)) incrementing by (*rate* + 8). For example, SHAKE256 defines a rate of 1088 bits. Therefore, for testing SHAKE256, unpredictable long messages will be generated with lengths (in bits) of 2184, 3280, 4376… 110688 bits.

The SHA3VS:

A. Generates 100 messages of the lengths specified above.

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, etc.)
   4. The output length (*Outputlen*) for the data to be generated. For testing purposes, the output length will be set to the minimum of the security level and max output length, i.e., min (*sec_level*, *maxoutlen*). For example, if SHAKE128 is being tested and the maxoutlen is 5000 bits, the output length for this test will be 128 bits.
   5. The sequence of 100 datasets containing:
      a. The length of the input message *(Len)*
      b. The input message (*Msg*).

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

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, etc.)
   4. The output length (*Outputlen*) for the data to be generated
   5. The sequence of 100 datasets containing:
      a. The length of the input message *(Len)*
      b. The input message (*Msg*)
      c. The resulting *Output* of length *Outputlen*.

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

The IUT:

A. Generates output data using the messages supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. The sequence of 100 datasets containing:
    a. The length of the input message *(Len)*
    b. The input message (*Msg*)
    c. The *Output* of the required length as specified in the request file.

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output data.

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

6.3.3 The Pseudorandomly Generated Messages (Monte Carlo) Test

The SHA3VS tests the correctness of output data generated from pseudorandomly generated messages. The SHA3VS provides an initial message, *Seed*, which is processed by the SHAKE algorithm to generate output. Each Monte Carlo test generates 100,000 outputs. 100 of the 100,000 outputs, every 1,000th output, is recorded as checkpoints to the operation of the SHAKE generator. The IUT uses the same procedure to generate the same 100,000 outputs and 100 checkpoint values. The SHA3VS compares each of the recorded 100 outputs with those generated by the IUT.

```
INPUT: The initial Msg of 128 bits long
       Initial Outputlen = (floor(maxoutlen/8) )*8
     //makes maxoutlen a multiple of 8 and remains within the
     range specified.
     {
     Output0 = Msg;
     for (j=0; j<100; j++) {
          for (i=1; i<1001; i++) {
               Msgi = 128 leftmost bits of Outputi-1;
               Outputi = SHAKE(Msgi,Outputlen);
               If (i == 1000){
                    Outputlenj = Outputlen;
               }
          Rightmost_Output_bits = rightmost 16 bits of Outputi;

               Range = (maxoutbytes – minoutbytes + 1);
               Outputlen = minoutbytes + (Rightmost_Output_bits
          mod Range);
               }
          Outputj = Output1000;
          OUTPUT: Outputlenj, Outputj
          }
     }
```

Figure 2: Code for Generating Pseudorandom Messages

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

1) 100,000 pseudorandom messages are generated. Initially, the message is a 128-bit message, *Msg*, supplied by the SHA3VS. For subsequent input messages, the leftmost 128 bits of the previous output is used as input to the SHAKE

algorithm. If the previous output has less than 128 bits, then zeros are concatenated to the end of the bits of output.

2) The *minoutlen* is the minimum length (in bits) tested for the output value. The *maxoutlen* is the maximum length in (bits) tested for the output value. The Monte Carlo test operates on byte-oriented data so these values are rounded to the nearest multiple of 8 bits while staying within the specified range. *minoutbytes* is the minoutlen rounded up to the nearest multiple of 8. *maxoutbytes* is the maxoutlen rounded down to the nearest multiple of 8. (Note by rounding the *minoutlen* up and the *maxoutlen* down, the values stay within the range to be tested as specified by the IUT.) (Note2: If *minoutlen* = *maxoutlen* and if the value is a multiple of 8, that value is used as the length for all rounds. If the value is NOT a multiple of 8, a message appears instructing the lab to contact the CAVP for guidance on this special case.)

3) Initially the output length is *maxoutbytes*. Subsequently, the rightmost 16 bits of the output is used in the calculation of the next iterations output length. Let *Rightmost_Output_bits* = rightmost 16 bits of output interpreted as an integer. Let *Range* (in bytes) = (*maxoutbytes – minoutbytes* + 1). The next output length (in bytes) = *minoutbytes* + (*Rightmost_Output_bits* mod *Range*).

   For example, let the *minoutbytes* be 2 bytes (16 bits), the *maxoutbytes* be 8192 bytes (2^16 = 65536 bits) and the 16 *Rightmost_Output_bits* = C596 (in hexadecimal).

   The initial output length is 8192 bytes (65536 bits). The *Rightmost_Output_bits* = int (C596) = 50582, *Range* = 8192 - 2 + 1 = 8191 bytes, and *Outputlen* = 2 + (50582 mod 8191) = 1438 bytes. This is the new output length in bytes; and

4) After every 1,000 outputs, a sample is taken and is provided as a checkpoint. These checkpoints are denoted $Output_0$ in Figure 2.

The SHAVS:

A. Generates an initial message, *Msg*, of length 128 bits.

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

   1. The CAVS Version
   2. The Product Information (product name, test specifications)
   3. Test Information (generation date, specification of data representation, etc.);
   4. The minimum output length, *minoutlen*, tested for the IUT. For testing purposes, the minimum output length is greater than 16 bits and is rounded up to a multiple of 8 bits in the test, *minoutbytes*.

5. The maximum output length, *maxoutlen*, tested for the IUT. For testing purposes, the maximum output length that can be tested is 2^16 bits and is rounded down to a multiple of 8 bits in the test, *maxoutbytes*.
6. Initial message *Msg* of length 128 bits.

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

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.);
4. The minimum output length rounded up to a multiple of 8 bits, (*minoutbytes *8)*, tested for the IUT. For testing purposes, the minimum output length is greater than 16 bits.
5. The maximum output length rounded down to a multiple of 8 bits, (*maxoutbytes *8)*, tested for the IUT. For testing purposes, the maximum output length that can be tested is 2^16 bits.
6. Initial message *Msg* of length 128 bits
7. 100 data sets containing:
    a. *COUNT*
    b. Length in bytes of the expected output, *Outputlen*
    c. The *Output* of the 1000^{th} round

Note: The CST laboratory retains the *FAX* file at the SHA3VS.

The IUT:

A. Generates the 100 *Outputs* using the initial *Msg* supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (vendor, product name, version);
3. The initial *Msg*; and
4. 100 data sets containing:
    a. *COUNT*
    b. Length in bytes of the expected output, *Outputlen*
    c. The *Output* of every 1000^{th} round

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. The SHA3VS verifies that the IUT generated the correct outputs.

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

6.3.4 The Variable Output Test

An implementation of the SHAKE algorithms must be able to correctly generate arbitrary length output for a given input. The SHA3VS tests this property by supplying the IUT with a number of input messages of fixed length and assuring that the correct output value of the specified length is calculated by the IUT. The Variable Output Test has two versions, one for bit-oriented output length implementations and another for byte-oriented output length implementations.

6.3.4.1 The Variable Output Test for Bit-Oriented Output Length Implementations

For testing purposes, the input message length will be set to the security level for the SHAKE algorithm, i.e., for SHAKE128, the input message length will be 128 bits; for SHAKE256, the input message length will be 256 bits. The IUT supplies the minimum and maximum output lengths to be tested. For each round, the IUT will input a 128 bit message into the SHAKE algorithm and output a value. Initially the length of the output value will be *minoutlen*.

The SHA3VS:

A. Creates a *REQUEST* file (Filename: <Alg>VariableOut.req) containing:

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.)
4. Indication of tested for output of byte-oriented or bit-oriented messages
5. The length of the input message *(Input Length)*
6. The minimum output length (*minoutlen)* tested
7. The maximum output length (*maxoutlen*) tested
8. A set of datasets containing:
    a. *COUNT*
    b. The length of the output (*Outputlen*). Values of *Outputlen* range from *minoutlen* to the *maxoutlen*.
    c. A random message *Msg*.

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

B. Creates a *FAX* file (Filename: <Alg>VariableOut.fax) containing:

1. The CAVS Version

2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.)
4. Indication of tested for output of byte-oriented or bit-oriented messages
5. The length of the input message *(Input Length)*
6. The minimum output length (*minoutlen)* tested
7. The maximum output length (*maxoutlen*) tested
8. A set of datasets containing:
    a. *COUNT*
    b. The length of the output (*Outputlen*)
    c. A random message *Msg*
    d. The resulting *Output* of length *Outputlen*

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

The IUT:

A. Generates *Output* using the input messages supplied by the SHA3VS in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: <Alg>VariableOut.rsp) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.)
    4. Indication of tested for output of byte-oriented or bit-oriented messages
    5. The length of the input message *(Input Length)*
    6. The minimum output length (*minoutlen)* tested
    7. The maximum output length (*maxoutlen*) tested
    8. A set of datasets containing:
        a. *COUNT*
        b. The length of the output (*Outputlen*)
        c. The input message (*Msg*)
        d. The resulting *Output*

    Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output.

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

6.3.4.2 The Variable Output Test for Byte-Oriented Output Length Implementations

For testing purposes, the input message length will be set to the security level for the SHAKE algorithm, i.e., for SHAKE128, the input message length will be 128 bits; for SHAKE256, the input message length will be 256 bits. The IUT supplies the minimum and maximum output lengths to be tested. For each round, the IUT will input a 128 bit message into the SHAKE algorithm and output a value. Initially the length of the output value will be *minoutlen*.

The SHA3VS:

A. Creates a *REQUEST* file (Filename: <Alg>VariableOut.req) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.)
    4. Indication of tested for output of byte-oriented or bit-oriented messages
    5. The length of the input message *(Input Length)*
    6. The minimum output length (*minoutlen)* tested
    7. The maximum output length (*maxoutlen*) tested
    8. A set of datasets containing:
        a. *COUNT*
        b. The length of the output (*Outputlen*). Values of *Outputlen* range from *minoutlen* to the *maxoutlen*.
        c. The input message (*Msg*)

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

B. Creates a *FAX* file (Filename: <Alg>VariableOut.fax) containing:
    1. The CAVS Version
    2. The Product Information (product name, test specifications)
    3. Test Information (generation date, specification of data representation, etc.)
    4. Indication of tested for output of byte-oriented or bit-oriented messages
    5. The length of the input message *(Input Length)*
    6. The minimum output length (*minoutlen)* tested
    7. The maximum output length (*maxoutlen*) tested
    8. A set of datasets containing:

a. *COUNT*

b. The length of the output (*Outputlen)*

c. The input message (*Msg*)

d. The resulting *Output* of length *Outputlen*

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

The IUT:

A. Generates *Output* using the input messages supplied by the SHA3VS in the *REQUEST* file.

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

1. The CAVS Version
2. The Product Information (product name, test specifications)
3. Test Information (generation date, specification of data representation, etc.)
4. Indication of tested for output of byte-oriented or bit-oriented messages
5. The length of the input message *(Input Length)*
6. The minimum output length (*minoutlen)* tested
7. The maximum output length (*maxoutlen*) tested
8. A set of datasets containing:
    a. *COUNT*
    b. The length of the output (*Outputlen*)
    c. The input message (*Msg*)
    d. The resulting *Output*

Note: The IUT sends the *RESPONSE* file to the SHA3VS.

The SHA3VS:

A. Compares the *RESPONSE* file with the *FAX* file. For each message, the SHA3VS verifies that the IUT generated the correct output.

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

Appendix A References

[1] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, FIPS Publication 202, National Institute of Standards and Technology, August 2015.

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