The Keyed-Hash Message Authentication Code Validation System (HMACVS)

December 3, 2004

Lawrence E. Bassham III

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1 INTRODUCTION **1**

2 SCOPE **1**

3 CONFORMANCE **1**

4 DEFINITIONS AND ABBREVIATIONS **1**

4.1 DEFINITIONS 1
4.2 ABBREVIATIONS 2

5 DESIGN PHILOSOPHY OF THE KEYED-HASH MESSAGE AUTHENTICATION CODE VALIDATION SYSTEM **2**

6 HMACVS TEST **3**

6.1 CONFIGURATION INFORMATION 3
6.2 THE RANDOM MESSAGE TEST 4

APPENDIX A REFERENCES **6**

APPENDIX B EXAMPLES OF *REQUEST*, *FAX*, *RESPONSE*, AND *SAMPLE* FILES **7**

B.1 EXAMPLE OF THE *REQUEST* FILE 7
B.2 EXAMPLE OF THE *FAX* FILE 8
B.3 EXAMPLE OF THE *RESPONSE* FILE 9
B.4 EXAMPLE OF THE *SAMPLE* FILE 11

1 Introduction

This document, *The Keyed-Hash Message Authentication Code Validation System (HMACVS)* specifies the procedures involved in validating implementations of the Keyed-Hash Message Authentication Code (HMAC) as specified and approved in FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)* [1]. The HMACVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the HMACVS.

This document defines the purpose, the design philosophy, and the high-level description of the validation process for HMAC. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of an HMAC are presented. The requirements described include a specification of the data communicated between the IUT and the HMACVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the HMACVS. Additionally, an appendix is also provided containing samples of input and output files for the HMACVS.

2 Scope

This document specifies the tests required to validate IUTs for conformance to HMAC specified in [1]. When applied to an IUT, the HMACVS provides testing to determine the correctness of the implementation of HMAC. The HMACVS is composed of a single test that determines the conformance to the cryptographic specification. In addition to performing the test specified in HMACVS, the IUT must undergo testing of the underlying hash algorithm(s) used in the HMAC implementation via the SHAVS.

3 Conformance

The successful completion of the tests contained within the HMACVS and the SHAVS is required to be validated as conforming to the HMAC standard. Testing for the cryptographic module in which the HMAC is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*.[4]

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING

CMT laboratory	Cryptographic Module Testing laboratory that operates the HMACVS
Keyed-Hash Message Authentication Code	The algorithm specified in FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)*
Secure Hash Algorithm	The algorithm specified in FIPS 180-2, *Secure Hash Standard (SHS)*

4.2 Abbreviations

ABBREVIATION	MEANING
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code specified in FIPS 198
IUT	Implementation Under Test
SHA	Secure Hash Algorithm(s) specified in FIPS 180-2
SHAVS	Secure Hash Algorithm Validation System

5 Design Philosophy Of The Keyed-Hash Message Authentication Code Validation System

The HMACVS is designed to test conformance to the HMAC specification 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 HMACVS has the following design philosophy:

1. The HMACVS is designed to allow the testing of an IUT at locations remote to the HMACVS. The HMACVS and the IUT communicate data via *REQUEST* and *RESPONSE* files. The HMACVS also generates *SAMPLE* files to provide the IUT with a sample of what the *RESPONSE* file should look like.

2. The testing performed within the HMACVS 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 HMACVS Test

The HMACVS tests the implementation of HMAC for its conformance to the HMAC standard. The testing for HMAC consists of one test called the Random Message Test. The Random Message Test provides 15 sets of messages and keys for each hash algorithm/key size/MAC size combination supported by the IUT.

6.1 Configuration Information

To initiate the validation process of the HMACVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of HMAC. 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 HMACVS to perform the specific tests. More specifically, the request for validation includes:

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
7. Configuration information for the HMAC tests, including:
    a) Which underlying hash algorithms are supported (SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512);
    b) For each of the underlying hash algorithms specified above, supported key sizes, *K*, related to the block size, *B*, of the underlying hash algorithm. That is, does the IUT support key sizes of $K<B$, $K=B$, or $K>B$.
        - If the IUT supports key sizes of $K<B$: provide up to 2 distinct key sizes less than the block size that the IUT supports
        - If the IUT supports key sizes of $K>B$: provide up to 2 distinct key sizes greater than the block size that the IUT supports; and,

c) For each of the underlying hash algorithms supported, the length(s), *t*, in bytes, of the MAC the IUT is able to produce. Choices for the MAC lengths are:

- SHA-1: 10, 12, 16, 20
- SHA-224: 14, 16, 20, 24, 28
- SHA-256: 16, 24, 32
- SHA-384: 24, 32, 40, 48
- SHA-512: 32, 40, 48, 56, 64

6.2 The Random Message Test

The Random Message Test provides a series of random message and keys to the IUT. The IUT generates an HMAC for the messages using the keys provided. The HMACVS verifies the correctness of the HMACs produced by the IUT.

The HMACVS:

A. Creates a *REQUEST* file (Filename: HMAC.req) containing:

1. The Product Name;
2. The algorithm being tested; and
3. The messages and keys used as input to the HMAC algorithm.

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

B. Creates a *FAX* file (Filename: HMAC.fax) containing:

1. The information from the *REQUEST* file; and
2. The MAC generated by the HMAC algorithm.

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

The IUT:

A. Generates the requested MACs from the messages and keys specified in the *REQUEST* file.

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

1. The information from the *REQUEST* file; and
2. The MAC generated by the HMAC algorithm.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the HMACVS.

The HMACVS:

A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.

B. If all values match, records PASS for this test; otherwise, records FAIL.

Appendix A References

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

[2] *Public Key Cryptography for Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI X9.62-1988, January 1999.

[3] *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, ANSI X9.31-1988, September 1998.

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

Appendix B Examples of *REQUEST*, *FAX*, *RESPONSE*, and *SAMPLE* Files

The following are partial examples of *REQUEST*, *FAX*, *RESPONSE*, and *SAMPLE* files for the HMACVS.

B.1 Example of the *REQUEST* File

```
# CAVS 3.0
# Configuration information for "My Product"
# Hash sizes tested: 20 32 48 64
# Generated on Mon Sep 22 08:12:55 2003

[L=20]

Count = 0
Klen = 20
Tlen = 10
Key = 360546a692594fbbfc5ad3ddca918504e5da200b
Msg =
b15199bc12c0d5a6b3fff00f8c78b9580a34276904fceef6f0be5401573241a186bc96018795c0
ab4f268d0739b43e1eb342dda437b3d1367e30afe0f0c953e25e5f30b4f3c1e44768150f8e787a
663d94eb03d1d766ffd03c4e43520f0f9bc0a1000ddc408d680166951fac3111583c13f741f6cb
5ea34c94e2b95f9c4a42c2

Count = 1
Klen = 20
Tlen = 10
Key = 53b405624eb086c4c0d6e2853591a6982488b1e9
Msg =
2950f7de0995b924cdc20312e41d55337badeae643ee9cb474c902e112c2db416f74af4df13ab3
761530eebab45a9a748522cd78e408585fd195ef203c908be180750d0c2d2324a40ad570da5a53
e8db5f6a79ed1d99efba600d19eb00d028f9643d594045b7d456b49ad39a5f720610106b376c76
092a6c2819c76be7e87a34

Count = 2
Klen = 20
Tlen = 10
Key = 6f50a2778cc22a889ff49c5fd36e2deca6c5ef8d
Msg =
977dcc0d2fc504730c438a208d6b553b0cbd316c633c68eeae229be653d2f1f3400e7406746ab7
74c100bc49d06616e065dd4bb0e4cb7c69156c484024c4ed86ecd8b080741d8073ba31d6feb0c9
dfac85ebddca255dcb76035d3ff91692955721434a2438c7a698803c9ee6367cf8caf34db0612e
d89b3e025d4aba3124318b

...

Count = 178
Klen = 80
Tlen = 20
```

```
Key =
ef17028dbcae8e39023622381606cb30969534f19ce077fd380c33481e4ddf3c9b065d00ce32d2
fa1402b693c4c7273a1d1383ab7c1b3571d260cfa9568f5a0a77a5e922155675e2f0b0e45d61fc
6bcd
Msg =
7863eaf2b2d76a1987b9dfab417ef8692b9579d846498e453de85e079b25e4043cbbfa770153b7
f0c8a8fb9c9ce948c6285fd49e724fdc39c6223adaefc33a70ae49aa5f64d6308312bee17283ec
0e02c882610ce5c0296fca27f35785de5abfb554203442277ddc4b45b6b86051f226f2484aaa07
641fed3240b006c33eebf9

Count = 179
Klen = 80
Tlen = 20
Key =
650c17a3e6949522bd5aa513cbc9d25faa202e6493c852e77ee5a0508a2bcb16a0f29a78538de8
6e0bd8f0df6311c714cfe92a6d82da7a528fba056d4e91871601a8f10fb7e26872714f91e784ff
0c1a
Msg =
8fb978750339a43153ff72a59597cf5e3984c2979984a0a2d0167f89d03aecf4af78e14d4223f1
23eddfc6ccb961cd4fa77630814547056d7edbc5f9b6ddffc717a04d5a2560887a99cb38d96cbf
4ef632561642a920678292af1d74bf59c8b56e39a6e2d577cffdfec9b013d53aba1c802b5b23ee
972ea7b39b6f401335b186
```

B.2 Example of the *FAX* File

```
#  CAVS 3.0
#  Configuration information for "My Product"
#  Hash sizes tested: 20 32 48 64
#  Generated on Mon Sep 22 08:12:55 2003

[L=20]

Count = 0
Klen = 20
Tlen = 10
Key = 360546a692594fbbfc5ad3ddca918504e5da200b
Msg =
b15199bc12c0d5a6b3fff00f8c78b9580a34276904fceef6f0be5401573241a186bc96018795c0
ab4f268d0739b43e1eb342dda437b3d1367e30afe0f0c953e25e5f30b4f3c1e44768150f8e787a
663d94eb03d1d766ffd03c4e43520f0f9bc0a1000ddc408d680166951fac3111583c13f741f6cb
5ea34c94e2b95f9c4a42c2
Mac = 6f660bdba303bacbaf1f

Count = 1
Klen = 20
Tlen = 10
Key = 53b405624eb086c4c0d6e2853591a6982488b1e9
Msg =
2950f7de0995b924cdc20312e41d55337badeae643ee9cb474c902e112c2db416f74af4df13ab3
761530eebab45a9a748522cd78e408585fd195ef203c908be180750d0c2d2324a40ad570da5a53
e8db5f6a79ed1d99efba600d19eb00d028f9643d594045b7d456b49ad39a5f720610106b376c76
092a6c2819c76be7e87a34
```

```
Mac = 6aa8d611150333bdc4d6

Count = 2
Klen = 20
Tlen = 10
Key = 6f50a2778cc22a889ff49c5fd36e2deca6c5ef8d
Msg =
977dcc0d2fc504730c438a208d6b553b0cbd316c633c68eeae229be653d2f1f3400e7406746ab7
74c100bc49d06616e065dd4bb0e4cb7c69156c484024c4ed86ecd8b080741d8073ba31d6feb0c9
dfac85ebddca255dcb76035d3ff91692955721434a2438c7a698803c9ee6367cf8caf34db0612e
d89b3e025d4aba3124318b
Mac = 3ef5e685fa5eb3bb3bc3

...

Count = 178
Klen = 80
Tlen = 20
Key =
ef17028dbcae8e39023622381606cb30969534f19ce077fd380c33481e4ddf3c9b065d00ce32d2
fa1402b693c4c7273a1d1383ab7c1b3571d260cfa9568f5a0a77a5e922155675e2f0b0e45d61fc
6bcd
Msg =
7863eaf2b2d76a1987b9dfab417ef8692b9579d846498e453de85e079b25e4043cbbfa770153b7
f0c8a8fb9c9ce948c6285fd49e724fdc39c6223adaefc33a70ae49aa5f64d6308312bee17283ec
0e02c882610ce5c0296fca27f35785de5abfb554203442277ddc4b45b6b86051f226f2484aaa07
641fed3240b006c33eebf9
Mac = f3c77e284ebf0f0ab4d49cb73b58b1d40223c74f

Count = 179
Klen = 80
Tlen = 20
Key =
650c17a3e6949522bd5aa513cbc9d25faa202e6493c852e77ee5a0508a2bcb16a0f29a78538de8
6e0bd8f0df6311c714cfe92a6d82da7a528fba056d4e91871601a8f10fb7e26872714f91e784ff
0c1a
Msg =
8fb978750339a43153ff72a59597cf5e3984c2979984a0a2d0167f89d03aecf4af78e14d4223f1
23eddfc6ccb961cd4fa77630814547056d7edbc5f9b6ddffc717a04d5a2560887a99cb38d96cbf
4ef632561642a920678292af1d74bf59c8b56e39a6e2d577cffdfec9b013d53aba1c802b5b23ee
972ea7b39b6f401335b186
Mac = 9315c5b84ad71eb6294f1c0acf3673a9559cf791
```

B.3 Example of the *RESPONSE* File

```
#  CAVS 3.0
#  Configuration information for "My Product"
#  Hash sizes tested: 20 32 48 64
#  Generated on Mon Sep 22 08:12:55 2003

[L=20]
```

```
Count = 0
Klen = 20
Tlen = 10
Key = 360546a692594fbbfc5ad3ddca918504e5da200b
Msg =
b15199bc12c0d5a6b3fff00f8c78b9580a34276904fceef6f0be5401573241a186bc96018795c0
ab4f268d0739b43e1eb342dda437b3d1367e30afe0f0c953e25e5f30b4f3c1e44768150f8e787a
663d94eb03d1d766ffd03c4e43520f0f9bc0a1000ddc408d680166951fac3111583c13f741f6cb
5ea34c94e2b95f9c4a42c2
Mac = 6f660bdba303bacbaf1f

Count = 1
Klen = 20
Tlen = 10
Key = 53b405624eb086c4c0d6e2853591a6982488b1e9
Msg =
2950f7de0995b924cdc20312e41d55337badeae643ee9cb474c902e112c2db416f74af4df13ab3
761530eebab45a9a748522cd78e408585fd195ef203c908be180750d0c2d2324a40ad570da5a53
e8db5f6a79ed1d99efba600d19eb00d028f9643d594045b7d456b49ad39a5f720610106b376c76
092a6c2819c76be7e87a34
Mac = 6aa8d611150333bdc4d6

Count = 2
Klen = 20
Tlen = 10
Key = 6f50a2778cc22a889ff49c5fd36e2deca6c5ef8d
Msg =
977dcc0d2fc504730c438a208d6b553b0cbd316c633c68eeae229be653d2f1f3400e7406746ab7
74c100bc49d06616e065dd4bb0e4cb7c69156c484024c4ed86ecd8b080741d8073ba31d6feb0c9
dfac85ebddca255dcb76035d3ff91692955721434a2438c7a698803c9ee6367cf8caf34db0612e
d89b3e025d4aba3124318b
Mac = 3ef5e685fa5eb3bb3bc3

...

Count = 178
Klen = 80
Tlen = 20
Key =
ef17028dbcae8e39023622381606cb30969534f19ce077fd380c33481e4ddf3c9b065d00ce32d2
fa1402b693c4c7273a1d1383ab7c1b3571d260cfa9568f5a0a77a5e922155675e2f0b0e45d61fc
6bcd
Msg =
7863eaf2b2d76a1987b9dfab417ef8692b9579d846498e453de85e079b25e4043cbbfa770153b7
f0c8a8fb9c9ce948c6285fd49e724fdc39c6223adaefc33a70ae49aa5f64d6308312bee17283ec
0e02c882610ce5c0296fca27f35785de5abfb554203442277ddc4b45b6b86051f226f2484aaa07
641fed3240b006c33eebf9
Mac = f3c77e284ebf0f0ab4d49cb73b58b1d40223c74f

Count = 179
Klen = 80
Tlen = 20
```

```
Key =
650c17a3e6949522bd5aa513cbc9d25faa202e6493c852e77ee5a0508a2bcb16a0f29a78538de8
6e0bd8f0df6311c714cfe92a6d82da7a528fba056d4e91871601a8f10fb7e26872714f91e784ff
0c1a
Msg =
8fb978750339a43153ff72a59597cf5e3984c2979984a0a2d0167f89d03aecf4af78e14d4223f1
23eddfc6ccb961cd4fa77630814547056d7edbc5f9b6ddffc717a04d5a2560887a99cb38d96cbf
4ef632561642a920678292af1d74bf59c8b56e39a6e2d577cffdfec9b013d53aba1c802b5b23ee
972ea7b39b6f401335b186
Mac = 9315c5b84ad71eb6294f1c0acf3673a9559cf791
```

B.4 Example of the *SAMPLE* File

```
#  CAVS 3.0
#  Configuration information for "My Product"
#  Hash sizes tested: 20 32 48 64
#  Generated on Mon Sep 22 08:12:55 2003

[L=20]

Count = 0
Klen = 20
Tlen = 10
Key = 360546a692594fbbfc5ad3ddca918504e5da200b
Msg =
b15199bc12c0d5a6b3fff00f8c78b9580a34276904fceef6f0be5401573241a186bc96018795c0
ab4f268d0739b43e1eb342dda437b3d1367e30afe0f0c953e25e5f30b4f3c1e44768150f8e787a
663d94eb03d1d766ffd03c4e43520f0f9bc0a1000ddc408d680166951fac3111583c13f741f6cb
5ea34c94e2b95f9c4a42c2
Mac = ?

Count = 1
Klen = 20
Tlen = 10
Key = 53b405624eb086c4c0d6e2853591a6982488b1e9
Msg =
2950f7de0995b924cdc20312e41d55337badeae643ee9cb474c902e112c2db416f74af4df13ab3
761530eebab45a9a748522cd78e408585fd195ef203c908be180750d0c2d2324a40ad570da5a53
e8db5f6a79ed1d99efba600d19eb00d028f9643d594045b7d456b49ad39a5f720610106b376c76
092a6c2819c76be7e87a34
Mac = ?

Count = 2
Klen = 20
Tlen = 10
Key = 6f50a2778cc22a889ff49c5fd36e2deca6c5ef8d
Msg =
977dcc0d2fc504730c438a208d6b553b0cbd316c633c68eeae229be653d2f1f3400e7406746ab7
74c100bc49d06616e065dd4bb0e4cb7c69156c484024c4ed86ecd8b080741d8073ba31d6feb0c9
dfac85ebddca255dcb76035d3ff91692955721434a2438c7a698803c9ee6367cf8caf34db0612e
d89b3e025d4aba3124318b
Mac = ?
```

```
...

Count = 178
Klen = 80
Tlen = 20
Key =
ef17028dbcae8e39023622381606cb30969534f19ce077fd380c33481e4ddf3c9b065d00ce32d2
fa1402b693c4c7273a1d1383ab7c1b3571d260cfa9568f5a0a77a5e922155675e2f0b0e45d61fc
6bcd
Msg =
7863eaf2b2d76a1987b9dfab417ef8692b9579d846498e453de85e079b25e4043cbbfa770153b7
f0c8a8fb9c9ce948c6285fd49e724fdc39c6223adaefc33a70ae49aa5f64d6308312bee17283ec
0e02c882610ce5c0296fca27f35785de5abfb554203442277ddc4b45b6b86051f226f2484aaa07
641fed3240b006c33eebf9
Mac = ?

Count = 179
Klen = 80
Tlen = 20
Key =
650c17a3e6949522bd5aa513cbc9d25faa202e6493c852e77ee5a0508a2bcb16a0f29a78538de8
6e0bd8f0df6311c714cfe92a6d82da7a528fba056d4e91871601a8f10fb7e26872714f91e784ff
0c1a
Msg =
8fb978750339a43153ff72a59597cf5e3984c2979984a0a2d0167f89d03aecf4af78e14d4223f1
23eddfc6ccb961cd4fa77630814547056d7edbc5f9b6ddffc717a04d5a2560887a99cb38d96cbf
4ef632561642a920678292af1d74bf59c8b56e39a6e2d577cffdfec9b013d53aba1c802b5b23ee
972ea7b39b6f401335b186
Mac = ?
```