Key Derivation using Pseudorandom Functions (SP 800-108) Validation System (KBKDFVS)

March 22, 2012

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1 Introduction

This document, *Key Derivation Using Pseudorandom Functions (SP 800-108KDF) Validation System (KBKDFVS)*, specifies the procedures involved in validating implementations of the three key derivation functions found in SP800-108. The three key derivation functions include KDF in Counter Mode, KDF in Feedback Mode, and KDF in Double-Pipeline Iteration Mode. Each KDF in SP 800-108 uses a key to generate a key. Therefore the abbreviation used for these KDFs is KBKDF (Key Based Key Derivation Function). The testing encompasses IUTs that implement SP 800-108, *Recommendation for Key Derivation Using Pseudorandom Functions* [1]. The KBKDFVS is designed to perform automated testing on Implementations Under Test (IUTs).

This document defines the purpose, the design philosophy, and the high-level description of the validation process for each key derivation function. It includes specifications for tests that make up the KBKDFVS. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of SP800-108 are presented. The requirements described include a specification of the data communicated between the IUT and the KBKDFVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the KBKDFVS.

A set of KDF test vectors is available on the http://csrc.nist.gov/groups/STM/cavp/index.html website for testing purposes.

2 Scope

This document specifies the tests required to validate implementations of SP 800-108 for conformance to the key derivation functions (KDF). When applied to an Implementation Under Test (IUT), the KBKDFVS provides testing to determine the correctness of the implementation of the KDF specifications. As detailed in the Recommendation, three KDFs are described in SP800-108: KDF in Counter Mode, KDF in Feedback Mode and KDF in Double-Pipeline Iteration Mode. A separate validation test suite has been designed for each KDF. The validation testing verifies that an IUT has implemented the components of the KDF according to the specifications in the Recommendation. The requirements of NIST SP 800-108 addressable at the algorithm level and indicated by **shall** statements that are tested by the validation suite are listed in Appendix B.

3 Conformance

The successful completion of the tests contained within the KBKDFVS is required to claim conformance to SP800-108. Testing for the cryptographic module in which a KDF(s) is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*. [2]

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING
CST laboratory	Cryptographic Security Testing laboratory that operates the KBKDFVS
Key Derivation Function	A function for generating keying material

4.2 Abbreviations

ABBREVIATION	MEANING
CMAC	Block Cipher-based MAC Algorithm
CMACVS	CMAC Validation System
FIPS	Federal Information Processing Standard
h	An integer whose value is the length of the output of the PRF in bits
HMAC	Keyed-Hash Message Authentication Code
HMACVS	HMAC Validation System
IUT	Implementation Under Test
K_I	A key derivation key. For a key derivation, K_I is used (along with other data) to derive keying material K_O
K_O	Keying material that is derived from a key derivation key K_I and other data
KBKDF	Key Based Key Derivation Function – KDFs that use a key in the computation
KDF	Key Derivation Function
L	An integer specifying the length of the derived keying material K_O in bits, which is represented as a binary string when it is an input to a key derivation function

MAC	Message Authentication Code
PRF	Pseudorandom Function
r	An integer, smaller or equal to 32, whose value is the length of the binary representation of the counter *i* when *i* is an input in counter mode or (optionally) in feedback mode and double-pipeline iteration mode of each iteration of the PRF

5 Design Philosophy of SP800-108 KDF Validation System

The KBKDFVS is designed to test conformance to the specifications for each of the KDFs specified in NIST SP 800-108 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 KBKDFVS has the following design philosophy:

1. The KBKDFVS is designed to allow the testing of an IUT at locations remote to the KBKDFVS. The KBKDFVS and the IUT communicate data via *REQUEST* and *RESPONSE* files. The KBKDFVS also generates *SAMPLE* files to provide the IUT with an example of the format required by the *RESPONSE* file.

2. The testing performed within the KBKDFVS 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 Recommendation.

6 The Key Derivation Using Pseudorandom Functions (SP800-108KDF) Validation System (KBKDFVS) Test

The KBKDFVS tests the implementation for its conformance to SP 800-108.

When applied to an IUT, the KBKDFVS provides testing to determine the correctness of the implementation of the KDF specifications. A separate validation test suite has been designed for each KDF. The validation test suite for each KDF verifies that an IUT has implemented the components of the KDF according to the specifications in the Recommendation.

6.1 Configuration Information

To initiate the validation process of the KBKDFVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of one or more of the key derivation functions detailed in SP 800-108. The vendor's implementation is referred to as the IUT. The request for validation includes background information describing the IUT, along with information needed by the KBKDFVS to perform the specific tests. More specifically, the request for validation includes:

1. Cryptographic algorithm implementation information

    a. Vendor Name;

    b. Implementation Name;

    c. Implementation Version;

    d. Indication if implementation is software, firmware, or hardware;

    e. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;

    f. Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences); and

2. Configuration information for the KBKDFVS tests.

    a. The KDF(s) implemented:

        i. KDF in Counter Mode

        ii. KDF in Feedback Mode

        iii. KDF in Double-Pipeline Iteration Mode

3. If KDF in Counter Mode is implemented:

    a. *r*- length of the binary representation of the counter *i*. Possible values are 8, 16, 24, or 32 bits.

    b. Pseudo-random functions supported by the IUT

        i. CMAC AES 128, 192, 256

        ii. CMAC TDES 2, 3

        iii. HMAC SHA1, SHA224, SHA256, SHA384, SHA512

    c. *L*-length of the derived keying material K_0 in bytes – Enter all that apply:

i. A minimum and maximum value for full block lengths supported

ii. A minimum and maximum value for partial block lengths supported

d. Method(s) supported to generate *K*: SP800-56A, SP800-56B, SP800-90, an Approved RNG, N/A – Out of the scope of the algorithm implementation

e. Order of the fixed input data: Does the IUT support the counter coming before and/or after the fixed input data?

4. If KDF in Feedback Mode is implemented:

a. Pseudo-random functions supported by the IUT

i. CMAC AES 128, 192, 256

ii. CMAC TDES 2, 3

iii. HMAC SHA1, SHA224, SHA256, SHA384, SHA512

b. *L*-length of the derived keying material K_0 in bytes – Enter all that apply:

i. A minimum and maximum value for full block lengths supported

ii. A minimum and maximum value for partial block lengths supported

c. Method(s) supported to generate *K*: SP800-56A, SP800-56B, SP800-90A, an Approved RNG, N/A – Out of the scope of the algorithm implementation

d. Does the IUT support the counter being used as an input? If yes,

i. *r*- length of the binary representation of the counter *i*. Possible values are 8, 16, 24, or 32 bits.

ii. Does the IUT support the counter coming before and/or after the iteration variable and/or after the fixed input data?

5. If KDF in Double-Pipeline Iteration Mode is implemented:

a. Pseudo-random functions supported by the IUT

i. CMAC AES 128, 192, 256

ii. CMAC TDES 2, 3

iii. HMAC SHA1, SHA224, SHA256, SHA384, SHA512

b. *L*-length of the derived keying material K_0 in bytes – Enter all that apply:

   i. A minimum and maximum value for full block lengths supported

   ii. A minimum and maximum value for partial block lengths supported

c. Method(s) supported to generate *K*: SP800-56A, SP800-56B, SP800-90A, an Approved RNG, N/A – Out of the scope of the algorithm implementation

d. Does the IUT support the counter being used as an input? If yes,

   i. *r*- length of the binary representation of the counter *i*. Possible values are 8, 16, 24, or 32 bits.

   ii. Does the IUT support the counter coming before and/or after the iteration variable and/or after the fixed input data?

6.2 The Validation Test

A separate file is generated for each supported key derivation type. For example, if an IUT supports the KDF using Counter Mode and the KDF using Feedback Mode, two files will be generated:

KDFCTR_gen.req and

KDFFeedback_gen.req.

6.2.1 The Validation Test for KDF in Counter Mode

Within each request file, there is a section for each PRF supported, i.e., PRF=CMAC AES128, PRF=HMAC SHA224. Within each PRF section, there is a section for each "Counter Location" supported. Within each "Counter Location" section, there is a section for each *r* length supported. Within each *r* length supported, the test provides 10 sets of data for each *L* length specified. Four *L* lengths are specified including a minimum and maximum length divisible by *h* and a minimum and maximum length not evenly divisible by *h*. Therefore a total of 40 sets of data per PRF/Counter Location/rlen is generated. The set of data includes a count (this is a count of the test values and is not the counter used in the data to be MACed), an *L* and a randomly generated K_I.

The IUT supplies the fixed input data string length (in bytes), the fixed input data string value and the K_O.

The values generated by the IUT are stored in the *RESPONSE* file in the format specified

in the *SAMPLE* file. There shall be a *RESPONSE* file for every *SAMPLE* file.

The KBKDFVS will verify the correctness of the K_O using the inputs from the IUT and the values supplied by the KBKDFVS. The KBKDFVS will generate the string to be MACed by using the FixedInputData supplied by the IUT, generating the binary representation of *i* using the rlen information located in the section, and then concatenating the information in the order specified by the Counter location. This string is then MACed with the PRF algorithm specified resulting in the K_O. The KBKDFVS compares the IUT's K_O value to the KBKDFVS K_O value to see if they are the same. If they are, then it can be determined that the KDF is implemented correctly according to the Recommendation. If the values do not match, the IUT has an error in it. During the validation of the IUT, if an error occurs, the count, the variable and the values that don't match are stored in the log file. The laboratory uses this information to assist the vendor in debugging their IUT.

6.2.2 The Validation test for KDF in Feedback Mode

Within each request file, there is a section for each PRF supported, i.e., PRF=CMAC AES128, PRF=HMAC SHA224. The counter is optional in the data to be MACed. If the counter is not used, the PRF supported sections are the only sections in the files. If the counter is used, within each PRF section, there is a section for each "Counter Location" supported. Within each "Counter Location" section, there is a section for each r length supported.

For each combination of PRF (if no counter is used in the data to be MACed) or for each combination of PRF/Counter Location/rlen (if counter is used), the test provides 10 sets of data for each L length specified. Four L lengths are specified including a minimum and maximum length divisible by h and a minimum and maximum length not evenly divisible by h. Therefore a total of 40 sets of data per PRF/{Counter Location/rlen} is generated to the IUT. The set of data includes a count (this is a count of the test values and is not the counter used in the data to be MACed), an L, a randomly generated K_I, an IV length in bits, and an IV.

The IUT supplies the fixed input data string length (in bytes), the fixed input data string value and the K_O.

The values generated by the IUT are stored in the *RESPONSE* file in the format specified in the *SAMPLE* file. There shall be a *RESPONSE* file for every *SAMPLE* file.

The KBKDFVS will verify the correctness of the K_O using the inputs from the IUT and the values supplied by KBKDFVS. The KBKDFVS will generate the string to be MACed by using the FixedInputData supplied by the IUT, and the *IV*, for the first round, or the previous K_O value for subsequent rounds. If the counter is used in the data to be MACed, the KBKDFVS will generate the binary representation of *i* using the *rlen* information located in the section, and concatenating the information in the order specified. This string is then MACed with the PRF algorithm specified resulting in the K_O. The KBKDFVS compares the IUT's K_O value to the KBKDFVS K_O value to see if they are the same. If they are, then it can be determined that the KDF is implemented

correctly according to the Recommendation. If the values do not match, the IUT has an error in it. During the validation of the IUT, if an error occurs, the count and the values that don't match are stored in the log file. The laboratory uses this information to assist the vendor in debugging their IUT.

6.2.3 The Validation test for KDF in Double-Pipeline Iteration Mode

Within each request file, there is a section for each PRF supported, i.e., PRF=CMAC AES128, PRF=HMAC SHA224. The counter is optional in the data to be MACed. If the counter is not used, the PRF supported sections are the only sections in the files. If the counter is used, within each PRF section, there is a section for each "Counter Location" supported. Within each "Counter Location" section, there is a section for each *r* length supported.

For each combination of PRF (if no counter is used in the data to be MACed) or for each combination of PRF/Counter Location/rlen (if counter is used), the test provides 10 sets of data for each L length specified. Four L lengths are specified including a minimum and maximum length divisible by h and a minimum and maximum length not evenly divisible by h. Therefore a total of 40 sets of data per PRF/{Counter Location/rlen} is generated to the IUT. The set of data includes a count (this is a count of the test values and is not the counter used in the data to be MACed), an L, and a randomly generated K_I.

The IUT supplies the fixed input data string length (in bytes), the fixed input data string value and the K_O.

The values generated by the IUT are stored in the *RESPONSE* file in the format specified in the *SAMPLE* file. There shall be a *RESPONSE* file for every *SAMPLE* file.

The KBKDFVS will verify the correctness of the K_O using the inputs from the IUT and the values supplied by KBKDFVS. The KBKDFVS will generate the string to be MACed by using the FixedInputData supplied by the IUT. If the counter is used in the data to be MACed, the KBKDFVS will generate the binary representation of *i* using the *rlen* information located in the section, and then concatenate the information in the order specified. This string is then MACed with the PRF algorithm specified resulting in the K_O. The KBKDFVS compares the IUT's K_O value to the KBKDFVS K_O value to see if they are the same. If they are, then it can be determined that the KDF is implemented correctly according to the Recommendation. If the values do not match, the IUT has an error in it. During the validation of the IUT, if an error occurs, the count and the values that are in error are stored in the log file. The laboratory uses this information to assist the vendor in debugging their IUT.

Appendix A References

[1] *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, Special Publication 800-108, National Institute of Standards and Technology, October 2009.

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

Appendix B Tested Components of SP 800-108

The KBKDFVS validation testing for SP 800-108 tests all the algorithmic specifications, components, features, and functionalities specified in the Special Publication. In addition to these algorithmic conditions, the validation testing also addresses additional requirements identified as "shall" statements in the Special Publication that are applicable at the algorithm level.

B.1 SP800-108 Algorithmic Specifications

The following sections of SP 800-108 contain the algorithmic specifications:

4 Pseudorandom Function (PRF)
5 Key Derivation Functions
5.1 KDF using Counter Mode
5.2 KDF using Feedback Mode
5.3 KDF using Double-Pipeline Iteration Mode.

The validation test suite for SP 800-108 tests the requirements in each of these sections.

B.2 Additional Requirements of SP800-108

There is one additional requirement that is identified by a "shall" statement in SP 800-108 that is addressable at the algorithm level and therefore is tested by the KBKDFVS.

Section	Shall Statement	CAVP testing
7.4 Input Data Encoding	The encoding method shall define a one-to-one mapping from the set of all possible input information for that data field to a set of the corresponding binary strings.	*CAVS requires the IUT to supply the fixed input data, the Label, the Context, and the binary representation of L. CAVS then checks that the Label, Context and L are included in the fixed input data. Because the specific order of these components is not specified in the special publication, this is not checked*