The Key Wrap Validation System

Original: June 20, 2014

Timothy A. Hall

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1 Introduction

This document, *The Key Wrap Validation System (KWVS)*, specifies the procedures for validating implementations of the AES Key Wrap (KW), AES Key Wrap with Padding (KWP) and TDEA Key Wrap (TKW) authenticated encryption (AE) and authenticated decryption (AD) algorithms as specified in NIST SP 800-38F, *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping* [1]. The KWVS 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 KW, KWP and TKW. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of NIST SP 800-38F are presented. The requirements described include a specification of the data communicated between the IUT and the KWVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the KWVS.

Sets of KW, KWP and TKW test vectors are available at http://csrc.nist.gov/groups/STM/cavp/documents/mac/kwtestvectors.zip.

2 Scope

This document specifies the tests required to validate IUTs for conformance to the AES Key Wrap (KW), AES Key Wrap with Padding (KWP) and TDEA Key Wrap (TKW) as specified in [1]. When applied to an IUT, the KWVS provides testing to determine the correctness of the implementation of the KW, KWP or TKW algorithm specifications. As detailed in the standard, there is both an authenticated encryption function and an authenticated decryption function. A separate test suite has been designed for each of these functions and verifies that an IUT has implemented the components of the function according to the specifications in the standard.

3 Conformance

The successful completion of the tests contained within the KWVS is required to be validated as conforming to the NIST SP 800-38F standard. Testing for the cryptographic module in which KW, KWP or TKW is implemented is defined in FIPS PUB 140-2, Security Requirements for Cryptographic Modules [2].

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING
authenticated decryption function	The function of KW, KWP or TKW that decrypts the purported ciphertext into corresponding plaintext and verifies the authenticity and integrity of the data. The output is either the plaintext or an indication that the plaintext is not authentic (FAIL).
authenticated encryption function	The function of KW, KWP or TKW that encrypts plaintext into ciphertext and provides a means for the associated authenticated decryption function to verify the authenticity and integrity of the data.
designated cipher function	The underlying block cipher, along with a key encryption key, used for authenticated encryption. It may be either the forward transformation or the inverse transformation.
forward transformation	The permutation of blocks that is determined by the choice of a block cipher and a key.
inverse transformation	The inverse of the permutation of blocks that is determined by the choice of a block cipher and a key.
key-encryption key	The key for the underlying block cipher of KW, KWP or TKW. May be called a key-wrapping key in other documents.
key-wrap algorithm	A deterministic, symmetric-key authenticated-encryption algorithm that is intended for the protection of cryptographic keys. Consists of two functions: authenticated encryption and authenticated decryption.

4.2 Abbreviations

ABBREVIATION	MEANING
AD	Authenticated decryption
AE	Authenticated encryption
AES	Advanced Encryption System
AESAVS	Advanced Encryption System Algorithm Validation System

$CIPH_K$	The designated cipher function with key-encryption key K
$CIPH^{-1}_K$	The inverse of the designated cipher function with key-encryption key K
FIPS	Federal Information Processing Standard
KEK	key-encryption key
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
TKW	TDEA Key Wrap
IUT	Implementation Under Test

5 Design Philosophy of Key Wrap Validation System

The KWVS is designed to test conformance to KW, KWP and TKW specifications 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 KWVS has the following design philosophy:

1. The KWVS is designed to allow the testing of an IUT at locations remote to the KWVS. The KWVS and the IUT communicate data via *REQUEST* and *RESPONSE* files. The KWVS 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 KWVS uses 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 Key Wrap Validation System (KWVS) Test

The KWVS tests the implementation of the KW, KWP and TKW algorithms for conformance to the NIST SP 800-38F standard. When applied to an IUT, the KWVS provides testing to determine the correctness of the implementation of the authenticated encryption and/or the authenticated decryption function specifications. A separate test suite has been designed for each of these functions. The validation test suite for each function verifies that an IUT has implemented the components of the function according to

the specifications in the standard.

The key wrap algorithm validation process requires additional prerequisite testing of the underlying block cipher, whether AES or TDEA. The prerequisite block cipher testing must use a fundamental mode of operation – any of the ones in NIST SP 800-38A, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques* [3], except counter mode – that exercises the forward cipher function (forward transformation), inverse cipher function (inverse transformation) or both, depending upon which are used in the key wrap algorithm. If the IUT supports both authenticated encryption (AE) and authenticated decryption (AD), then both the forward and inverse cipher functions (transformations) must be tested. Example: an IUT supports KW-AE and KW-AD with the AES-128 forward transformation as the designated cipher function. The prerequisite testing must cover both the AES-128 forward and inverse transformations.

6.1 Configuration Information

To initiate the validation process of the KWVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of the KW, KWP or TKW algorithm. 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 KWVS to perform the specific tests. More specifically, the request for validation includes:

1. Cryptographic algorithm implementation information

    a. Vendor Name;

    b. Product Name;

    c. Product Version;

    d. Implementation in software, firmware, or hardware;

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

    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 KWVS tests.

    a. Algorithms supported – KW, KWP or TKW

    b. Authenticated Encryption (AE) and/or Authenticated Decryption (AD)

    c. Designated cipher function (CIPH)

1. KW and KWP – AES cipher function (i.e., forward transformation) and/or AES inverse cipher function (i.e., inverse transformation)

2. TKW – TDEA cipher function (i.e., forward transformation) and/or TDEA inverse cipher function (i.e., inverse transformation)

d. Key lengths supported

1. KW and KWP – 128, 192 and 256.

2. TKW – N/A since only one key length approved.

e. Five plaintext lengths for testing

1. KW – two lengths that are non-zero multiples of 128 bits (two semiblock lengths), two that are odd multiples of the semiblock length (64 bits), and the largest supported plaintext length less than or equal to 4096 bits.

2. KWP – four lengths that are multiples of 8 bits and the largest supported length less than or equal to 4096 bits.

3. TKW – two lengths that are non-zero multiples of 64 bits (two semiblock lengths), two that are odd multiples of the semiblock length (32 bits) and the largest supported length less than or equal to 4096 bits.

6.2 The Validation Test for the Authenticated Encryption Function

6.2.1 KW-AE

A separate request file is generated for each supported combination of designated cipher function and AES key length. For example, KW_AE_128.req is the file name for KW-AE using the AES-128 forward transformation as the designated cipher function; KW_AE_192_inv.req is the file name for KW-AE using the AES-192 inverse transformation as the designated cipher function.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a plaintext (P). The IUT uses these values to generate the ciphertext (C). For example, the start of a section in KW_AE_128.req would look like this:

```
[PLAINTEXT LENGTH = 128]

COUNT = 0
K = 681da528a1e9d2c74efb885e8d8c9b7d
P = 52aac5595d471bb91aea4b98ccc21123

COUNT = 1
.
.
```

.

All of 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 request file. The corresponding lines in KW_AE_128.rsp would be this:

```
[PLAINTEXT LENGTH = 128]

COUNT = 0
K = 681da528a1e9d2c74efb885e8d8c9b7d
P = 52aac5595d471bb91aea4b98ccc21123
C = d938b5b6bad14eecae3e3f488d933fff3ea95ff5cb666bec

COUNT = 1
.
.
.
```

The KWVS will verify the correctness of the IUT's values by comparing them to the known values generated by the KWVS. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

6.2.2 KWP-AE

The validation test for KWP-AE is essentially identical to that for KW-AE. Again, a separate request file is generated for each supported combination of designated cipher function and AES key length. For example, KWP_AE_192.req is the file name for KWP-AE using the AES-192 forward transformation as the designated cipher function; KWP_AE_256_inv.req is the file name for KWP-AE using the AES-256 inverse transformation as the designated cipher function.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a plaintext (P). The IUT uses these values to generate the ciphertext (C). For example, the start of a section in KWP_AE_192.req would look like this:

```
[PLAINTEXT LENGTH = 8]

COUNT = 0
K = 55daa2b3417a8ffae202e28f68d15065c707045d81c21bd7
P = ca

COUNT = 1
.
.
.
```

All of 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 request file. The corresponding lines in KWP_AE_192.rsp would be this:

```
[PLAINTEXT LENGTH = 8]

COUNT = 0
K = 55daa2b3417a8ffae202e28f68d15065c707045d81c21bd7
P = ca
C = 669ff8e71d9bb69b962b57acf3ca38b9

COUNT = 1
.
.
.
```

The KWVS will verify the correctness of the IUT's values by comparing them to the known values generated by the KWVS. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

6.2.3 TKW-AE

The validation test for TKW-AE is essentially identical to that for KW-AE and KWP-AE. TKW supports only a single TDEA key length, so a separate request file is generated for each supported designated cipher. For example, TKW_AE.req is the file name for TKW-AE using the TDEA forward transformation as the designated cipher function; TKW_AE_inv.req is the file name for TKW-AE using the TDEA inverse transformation as the designated cipher function.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a plaintext (P). The IUT uses these values to generate the ciphertext (C). For example, the start of a section in TKW_AE_inv.req would look like this:

```
[PLAINTEXT LENGTH = 64]

COUNT = 0
K = d4e2fd089696709594a4616ba22dd9ea5f7b8a9d989adabb
P = a1f88bab6f450dd5

COUNT = 1
.
.
.
```

All of 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 request file. The corresponding

lines in TKW_AE_inv.rsp would be this:

```
[PLAINTEXT LENGTH = 64]

COUNT = 0
K = d4e2fd089696709594a4616ba22dd9ea5f7b8a9d989adabb
P = a1f88bab6f450dd5
C = 244b1d08047c40451f23fa13

COUNT = 1
.
.
.
```

The KWVS will verify the correctness of the IUT's values by comparing them to the known values generated by the KWVS. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

6.3 The Validation Test for the Authenticated Decryption Function

6.3.1 KW-AD

A separate request file is generated for each supported combination of designated cipher function and AES key length. For example, KW_AD_128.req is the file name for KW-AD using the AES-128 forward transformation as the designated cipher function. Note that if the forward transformation is the designated cipher function ($CIPH_K$), then the inverse transformation is used in the authenticated decryption function, since the KW-AD function always uses the *inverse* of the designated cipher function, denoted $CIPH^{-1}_K$.

KW_AD_192_inv.req is the file name for KW-AD using the AES-192 inverse transformation as the designated cipher function, which means that the KW-AD function calls the AES-192 *forward* transformation internally.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a ciphertext (C). The IUT uses these values to recover the plaintext (P) and verify the authenticity of the data. For example, the start of a section in KW_AD_128.req would look like this:

```
[PLAINTEXT LENGTH = 128]

COUNT = 0
K = bc4ebbe0c61a487569290f7491f07fb8
C = ff0e62d10b92be910afa5512e8946b482dcb0a57bf2bf15e

COUNT = 1
.
```

```
.
.
```

Note that the ciphertext input is one semiblock (64 bits) longer than the plaintext length. All of 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 request file. The corresponding lines in KW_AD_128.rsp would be this:

```
[PLAINTEXT LENGTH = 128]

COUNT = 0
K = bc4ebbe0c61a487569290f7491f07fb8
C = ff0e62d10b92be910afa5512e8946b482dcb0a57bf2bf15e
P = 939a378cb7ee4398cf2364f3506d723d

COUNT = 1
.
.
.
```

For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that are not authentic; that is, they fail authentication. For a trial where the IUT determines that the ciphertext fails to authenticate, the value FAIL is returned instead of the plaintext. For example:

```
.
.
.
COUNT = 3
K = 37658f2799b944c127c06d2ac57756c8
C = ecde0610df7b21bcbc49fb044009251be3ee87ed673439bd
FAIL

COUNT = 4
.
.
.
```

The KWVS verifies that the correct responses are returned by the IUT. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

6.3.2 KWP-AD

The validation test for KWP-AD is essentially identical to that for KW-AD. A separate request file is generated for each supported combination of designated cipher function and

AES key length. For example, KWP_AD_192.req is the file name for KW-AD using the AES-192 forward transformation as the designated cipher function. Note that if the forward transformation is the designated cipher function (i.e., $CIPH_K$), then the inverse transformation is used in KWP-AD, since KWP-AD always uses the *inverse* of the designated cipher function, denoted $CIPH^{-1}_K$.

KWP_AD_256_inv.req is the file name for KWP-AD using the AES-256 inverse transformation as the designated cipher function, which means that the KWP-AD function calls the AES-256 *forward* transformation internally.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a ciphertext (C). The IUT uses these values to recover the plaintext (P) and verify the authenticity of the data. For example, the start of a section in KWP_AD_192.req would look like this:

```
[PLAINTEXT LENGTH = 8]

COUNT = 0
K = 2ba5871a33716da20105b609e7bc0c91e847a61aac52fb16
C = c6b3e9d3fb374029722441c6f08a6996

COUNT = 1
.
.
.
```

Note that the ciphertext input is two semiblocks long. All of 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 request file. The corresponding lines in KWP_AD_128.rsp would be this:

```
[PLAINTEXT LENGTH = 8]

COUNT = 0
K = 2ba5871a33716da20105b609e7bc0c91e847a61aac52fb16
C = c6b3e9d3fb374029722441c6f08a6996
P = 07

COUNT = 1
.
.
.
```

For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that fail authentication. For a trial where the IUT determines that the ciphertext fails to authenticate, the value FAIL is returned instead of the plaintext. For example:

```
.
.
.
COUNT = 2
K = e2ef0e67ebcb1d73383611c82913124b8b1bc83506a21024
C = e3bbd26e5b6a031d518957dd64286e98
FAIL

COUNT = 3
.
.
.
```

The KWVS verifies that the correct responses are returned by the IUT. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

6.3.3 TKW-AD

The validation test for TKW-AD is nearly identical to that for KW-AD and KWP-AD. TKW supports only a single TDEA key length, so a separate request file is generated for each supported designated cipher. For example, TKW_AD.req is the file name for TKW-AD using the TDEA forward transformation as the designated cipher function. Note that if the forward transformation is the designated cipher function (i.e., $CIPH_K$), then the inverse transformation is used in TKW-AD, since TKW-AD always uses the *inverse* of the designated cipher function, denoted $CIPH^{-1}_K$.

TKW_AD_inv.req is the file name for TKW-AD using the TDEA inverse transformation as the designated cipher function, which means that the TKW-AD function calls the TDEA *forward* transformation internally.

Within each request file, there is a section for each of the five plaintext lengths. Each section has 100 trials. Each trial (or count) has two input values: a key (K) and a ciphertext (C). The IUT uses these values to recover the plaintext (P) and verify the authenticity of the data. For example, the start of a section in TWP_AD_inv.req would look like this:

```
[PLAINTEXT LENGTH = 64]

COUNT = 0
K = e3a77f4b7ac02ea03198d3e607cf20a955aac44af52ff060
C = 6368be85c3c8dbd05156b519

COUNT = 1
.
.
.
```

Note that the ciphertext input is one semiblock (32 bits) longer than the plaintext length. All of 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 request file. The corresponding lines in TKW_AD_inv.rsp would be this:

```
[PLAINTEXT LENGTH = 64]

COUNT = 0
K = e3a77f4b7ac02ea03198d3e607cf20a955aac44af52ff060
C = 6368be85c3c8dbd05156b519
P = 3a72755de0f3b528

COUNT = 1
.
.
.
```

For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that fail authentication. For a trial where the IUT determines that the ciphertext fails to authenticate, the value FAIL is returned instead of the plaintext. For example:

```
.
.
.
COUNT = 2
K = 33c808b242c8a6973f8b7520db586ae9c828853acd438f06
C = 01b72d953f74c090af730af2
FAIL

COUNT = 3
.
.
.
```

The KWVS verifies that the correct responses are returned by the IUT. If they match, the KWVS records a value of PASSED; otherwise, the KWVS records a value of FAILED.

Appendix A References

[1] *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, Special Publication 800-38F, National Institute of Standards and Technology, December 2012.

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

[3] *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, Special Publication 800-38A, National Institute of Standards and Technology, December 2001.