The Key Agreement Schemes Validation System (KASVS)

March 11, 2009

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

Revisions since 12/24/08

(Revised parts are underlined.)

1. Section 2 Scope, Paragraph about the prerequisites:

   a. The KASVS validation process also requires prerequisite testing of the underlying algorithms used in the implementation. They include:

      1. The underlying DSA and/or ECDSA algorithm’s domain parameter and/or key pair functions if the assurances selected indicate that this function should be in the implementation. Please refer to Table 1 in this document to determine what, if anything, needs to be tested as a prerequisite,

      3. The supported MAC algorithms (CCM, CMAC, and/or HMAC) if Key Confirmation is supported, and

TABLE OF CONTENTS

1 Introduction

This document, *The Key Agreement Scheme (KAS) Validation System (KASVS)*, specifies the procedures involved in validating implementations of the Key Agreement Schemes, and, if applicable, Key Confirmation as specified in SP 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography* [1]. The KASVS 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 agreement scheme, either alone or accompanied with key confirmation. It includes specifications for the two categories of tests that make up the KASVS, i.e., the Function test and the Validity test. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of SP800-56A are presented. The requirements described include a specification of the data communicated between the IUT and the KASVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the KASVS.

A set of KAS test vectors is available on the http://csrc.nist.gov/cryptval/ website for testing purposes.

2 Scope

This document specifies the tests required to validate implementations of SP 800-56A for conformance to the key agreement schemes, either alone or accompanied with key confirmation, as specified in [1]. When applied to an Implementation Under Test (IUT), the KASVS provides testing to determine the correctness of the implementation of the key agreement scheme specifications and, if applicable, the key confirmation specifications. As detailed in the Recommendation, Discrete Logarithm Cryptography (DLC) includes Finite Field Cryptography (FFC) and Elliptic Curve Cryptography (ECC). A separate validation test suite has been designed for each of these types of cryptography. These validation test suites contain validation testing for each key agreement scheme. The validation testing verifies that an IUT has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the validation test suite also verifies that an IUT has implemented the components of key confirmation as specified in the Recommendation. This includes the parsing of the DKM, the generation of MacData and the calculation of MacTag.

The KASVS validation process requires the definition of all assurances included in the implementation. These assurances are described in SP 800-56A, Section 5.5.2, 5.6.2, and 5.6.3.

The KASVS validation process also requires prerequisite testing of the underlying algorithms used in the implementation. They include:

1. The underlying DSA and/or ECDSA algorithm's domain parameter and/or key pair functions if the assurances selected indicate that this function should be in the implementation. Please refer to Table 1 in this document to determine what, if anything, needs to be tested as a prerequisite,

2. The supported SHA algorithm(s),

3. The supported MAC algorithms (CCM, CMAC, and/or HMAC) if Key Confirmation is supported, and

4. The supported random number generations including the approved RNG algorithms and the DRBG algorithm(s).

3 Conformance

The successful completion of the tests contained within the KASVS, the SHAVS, the RNGVS and/or DRBGVS and, if applicable, the DSAVS, the ECDSAVS CMACVS, CCMVS, and/or HMACVS is required to claim conformance to SP800-56A. Testing for the cryptographic module in which a key agreement scheme(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**
Assurance of identifier	Confidence that identifying information (such as a name) is correctly associated with an entity
Assurance of possession of a private key	Confidence that an entity possesses a private key associated with a public key.
Assurance of validity	Confidence that either a key or a set of domain parameters is arithmetically correct
CMT laboratory	Cryptographic Module Testing laboratory that operates the KASVS

Key agreement	A key establishment procedure where the resultant secret keying material is a function of information contributed by two participants, so that no party can predetermine the value of the secret keying material independently from the contributions of the other parties.
Key confirmation	A procedure to provide assurance to one party (the key confirmation recipient) that another party (the key confirmation provider) actually possesses the correct secret keying material and/or shared secret.

4.2 Abbreviations

ABBREVIATION	**MEANING**
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CCMVS	CCM Validation System
CMACVS	CMAC Validation System
DKM	Derived Keying Material
DLC	Discrete Logarithm Cryptography
DSA	Digital Signature Algorithm
DSAVS	Digital Signature Algorithm Validation System
ECDSA	Elliptic Curve Digital Signature Algorithm
ECDSAVS	ECDSA Validation System
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code
HMACVS	HMAC Validation System
IUT	Implementation Under Test
KAS	Key Agreement Scheme
KC	Key Confirmation
KDF	Key Derivation Function
KES	Key Establishment Scheme

MAC	Message Authentication Code
SHA	Secure Hash Algorithm
SHAVS	SHA Validation System
Z	A shared secret that is used to derive secret keying material using a key derivation function; a DLC primitive – either Diffie-Hellman or MQV.

5 Design Philosophy of Key Agreement Schemes Validation System

The KASVS is designed to test conformance to the key agreement and key confirmation 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 KASVS has the following design philosophy:

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

The KASVS tests the implementation of the key agreement and the key confirmation processes for its conformance to SP800-56A. When applied to an IUT, the KASVS provides testing to determine the correctness of the implementation of the key agreement scheme specifications. As detailed in the Recommendation, Discrete Logarithm Cryptography (DLC) includes Finite Field Cryptography (FFC) and Elliptic Curve Cryptography (ECC). A separate validation test suite has been designed for each of these types of cryptography. Within each test suite, validation testing has been designed for each key agreement scheme. The validation test suite for each key agreement scheme verifies that an IUT has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the validation test suite also verifies that the components of key confirmation as specified in the Recommendation have been implemented correctly. This includes the parsing of the DKM, the generation of MacData and the calculation of MacTag. There are several assurances that are defined in SP800-56A indicating where specific assurances are to be obtained - within the implementation or external to the implementation. This information is supplied by an IUT to aid in the testing of the scheme's components. Please refer to Sections 5.5.2, 5.6.2.1, 5.6.2.2, 5.6.2.3, 5.6.3.1, 5.6.3.2.1, and 5.6.3.2.2 of the Recommendation for more information on the assurances required for testing. A list of these is also supplied in Section 6.1 of this document.

The KAS validation process requires prerequisite testing of the underlying algorithms used in the implementation. They include:

1. The underlying DSA and/or ECDSA algorithm's domain parameter and/or key pair functions if the assurances selected indicate that this function should be in the implementation. Please refer to Table 1 in this document to determine what, if anything, needs to be tested as a prerequisite,

2. The supported SHA algorithm(s),

3. The supported MAC algorithms (CCM, CMAC, and/or HMAC) if Key Confirmation is supported, and

4. The supported random number generations including the approved RNG algorithms and the DRBG algorithm(s).

6.1 Configuration Information

To initiate the validation process of the KASVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of the key agreement scheme with or without key confirmation. 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 KASVS 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 KASVS tests.

   a. The underlying cryptographic schemes supported by the IUT, i.e., FFC and/or ECC. The FFC schemes are based on ANS X9.42 and the ECC schemes are based on ANS X9.63.

   b. For each underlying algorithm, a list of each assurance supported by the IUT. Based on the assurances supported by an IUT, the scope of the validation testing necessary to thoroughly test the implementation is determined.

      Below is a list of the assurances. Each assurance is either tested in the CAVS KAS testing, tested by requiring a prerequisite, or out-of-scope of the CAVP KAS validation testing. (Note that the section numbers corresponding to the SP800-56A document are included. Please refer to SP800-56A for more details on the assurances.)

5.5.2 Assurances of Domain Parameter Validity	
#1. Domain parameters generated by party itself	FFC: PREREQUISITE: DSA PQG Generation and Verification functions
	ECC: ECDSA uses NIST-approved curves, so no need to test
#2. Explicit domain parameter validation (specified in FIPS186-3 or ANSX9.62-2)	FFC: PREREQUISITE: DSA PQG Verification function
	ECC: ECDSA uses NIST-approved curves, so no need to test
#3. Party received assurance from trusted third party.	Out-of-scope of CAVP KAS testing.
5.6.2 Assurances of the Arithmetic Validity of a Public Key	
5.6.2.1 Owner Assurances of Static Public Key Validity	
#1. Owner Full Validation.	FFC: Testing done within KAS validation testing
	ECC: PREREQUISITE: ECDSA PKV

	function
#2. TTP Full Validation.	Out-of-scope of CAVP KAS testing.
#3. Owner Generation.	FFC: PREREQUISITE: DSA Key Pair Generation functions
	ECC: PREREQUISITE: ECDSA Key Pair function
#4. TTP Generation.	Out-of-scope of CAVP KAS testing.
5.6.2.2 Recipient Assurances of Static Public Key Validity	
#1. Recipient Full Validation.	FFC: Testing done within KAS validation testing
	ECC: PREREQUISITE: ECDSA PKV function
#2. TTP Full Validation.	Out-of-scope of CAVP KAS testing.
#3. TTP Generation.	Out-of-scope of CAVP KAS testing.
5.6.2.3 Recipient Assurances of Ephemeral Public Key Validity	
#1. Recipient Full Validation.	FFC: Testing done within KAS validation testing
	ECC: PREREQUISITE: ECDSA PKV function
#2. TTP Full Validation.	Out-of-scope of CAVP KAS testing.
#3. Recipient ECC Partial Validation.	Testing done within KAS validation testing
#4. TTP ECC Partial Validation.	Out-of-scope of CAVP KAS testing.
5.6.3 Assurances of the Possession of a Static Private Key	
5.6.3.1 Owner Assurances of Possession of a Static Private Key	
#1. Owner Receives Assurance via Explicit Key	IUT must implement Key Confirmation

Confirmation	within KAS validation testing
#2. Owner Receives Assurance via Use of an Encrypted Certificate.	Out-of-scope of CAVP KAS testing.
#3. Owner Receives Assurance via Key Regeneration	FFC: PREREQUISITE: DSA Key Pair Generation function
	ECC: PREREQUISITE: ECDSA Key Pair and PKV functions
#4. Owner Receives Assurance via Trusted Provision.	Out-of-scope of CAVP KAS testing.
#5. Owner Receives Assurance via Key Generation	FFC: PREREQUISITE: DSA Key Pair Generation function
	ECC: PREREQUISITE: ECDSA Key Pair and PKV functions
5.6.3.2 Recipient Assurances of Owner's Possession of a Static Private Key	
5.6.3.2.1 Recipient Obtains Assurance through a Trusted Third Party.	Out-of-scope of CAVP KAS testing.
5.6.3.2.2 Recipient Obtains Assurance Directly from the Claimed Owner	FFC: IUT must implement either Hybrid1Flow, MQV1, and/or OneFlow.
	ECC: IUT must implement either One-Pass Unified Model, One-Pass MQV, or One-Pass Diffie-Hellman.
	BOTH: IUT must implement key confirmation with the IUT as the provider and the responder

Table 1: List of Assurances and How They Relate to the CAVP KAS Validation Testing

3. If FFC is implemented, the following configuration information is required:

   i. Supported key agreement scheme(s):

      - dhHybrid1, MQV2, dhEphem, dhHybridOneFlow, MQV1, dhOneFlow, dhStatic

ii. Supported roles for key agreement:

    - Initiator, Responder

iii. If key confirmation is supported, supported roles for key confirmation:

    - Provider, Recipient

iv. If key confirmation is supported, types of key confirmation:

    - Unilateral, Bilateral

v. Parameter size set(s) supported:

    - FA
    - FB
    - FC

    (Refer to SP800-56A, Section 5.5.1.1, Table 1, FFC Parameter Size Sets for more information.)

vi. SHA algorithm(s) supported for use in the key derivation function testing

vii. If key confirmation is supported, indicate all MACs supported by the IUT, along with the associated information. If key confirmation is not supported, indicate one MAC supported by the IUT, along with the associated information. The MACs to choose from are listed below:

    - A NIST-approved MAC supported by the IUT:
        - CCM:
            - Algorithm: AES, TDES
            - Key Size: 128, 192, 256
            - Nonce Length in bytes: 7, 8, 9, 10, 11, 12, 13
            - Tag Length in bytes:
                - For FA: 10, 12, 14, 16
                - For FB: 14, 16
                - For FC: 16

- CMAC:
    - Algorithm and key size: AES128, AES192, AES256
    - Tag Length in bytes:
        - For FA: 10 <= Tag Length <= 16
        - For FB: 14 <= Tag Length <=16
        - For FC: Tag Length = 16
- HMAC:
    - For FA:
        - SHA Algorithm supported: SHA1, SHA224, SHA256, SHA384, SHA512
        - HMAC Key Size in bytes: >= 10 bytes
        - Tag Length in bytes: >= 10 bytes
    - For FB:
        - SHA Algorithm supported: SHA224, SHA256, SHA384, SHA512
        - HMAC Key Size in bytes: >= 14 bytes
        - Tag Length in bytes: >= 14 bytes
    - For FC:
        - SHA Algorithm supported: SHA256, SHA384, SHA512
        - HMAC Key Size in bytes: >= 16 bytes
        - Tag Length in bytes: >= 16 bytes

d. If ECC is implemented, the following configuration information is required:

i. Supported key agreement scheme(s):

a. (Cofactor) Full Unified Model, Full MQV, (Cofactor) Ephemeral

Unified Model, (Cofactor) One-Pass Unified Model, One-Pass MQV, (Cofactor) One-Pass Diffie-Hellman, Cofactor Static Unified Model

ii. Supported roles:

   b. Initiator, Responder

iii. If key confirmation is supported, supported roles for key confirmation:

   c. Provider, Recipient

iv. If key confirmation is supported, supported types of key confirmation:

   d. Unilateral, Bilateral

v. Parameter set(s) supported:

   - EA
   - EB
   - EC
   - ED
   - EE

   (Refer to SP800-56A, Section 5.5.1.2, Table 2, ECC Parameter Size Sets for more information.)

vi. Supported curve (indicate one per parameter set supported). Note, if an IUT supports both prime fields and polynomial fields, a parameter set from each field should be tested:

   - For EA: P192, K163, B163
   - For EB: P224, K233, B233
   - For EC: P256, K283, B283
   - For ED: P384, K409, B409
   - For EE: P512, K571, B571

vii. SHA algorithms supported for use in the key derivation function testing.

viii. If key confirmation is supported, indicate all MACs supported by the

IUT, along with the associated information. If key confirmation is not supported, indicate one MAC supported by the IUT, along with the associated information. The MACs to choose from are listed below:

- A NIST-approved MAC supported by the IUT:
  - CCM:
    - Algorithm: AES, TDES
    - Key Size: 128, 192, 256
    - Nonce Length in bytes: 7, 8, 9, 10, 11, 12, 13
    - Tag Length in bytes:
      - For EA: 10, 12, 14, 16
        - For EB: 14, 16
        - For EC: 16
    - CMAC (Only for use with EA, EB, EC):
      - Algorithm and key size: AES128, AES192, AES256
      - Tag Length in bytes:
        - For EA: 10 <= Tag Length <= 16
        - For EB: 14 <= Tag Length <=16
        - For EC: Tag Length = 16
    - HMAC:
      - For EA:
        - SHA Algorithm supported: SHA1, SHA224, SHA256, SHA384, SHA512
        - HMAC Key Size in bytes: >= 10 bytes
        - Tag Length in bytes: >= 10 bytes
      - For EB:
        - SHA Algorithm supported: SHA224,

SHA256, SHA384, SHA512

        - HMAC Key Size in bytes: >= 14 bytes
        - Tag Length in bytes: >= 14 bytes

    - For EC:
        - SHA Algorithm supported: SHA256, SHA384, SHA512
        - HMAC Key Size in bytes: >= 16 bytes
        - Tag Length in bytes: >= 16 bytes
    - For ED:
        - SHA Algorithm supported: SHA384, SHA512
        - HMAC Key Size in bytes: >= 24 bytes
        - Tag Length in bytes: >= 24 bytes
    - For EE:
        - SHA Algorithm supported: SHA512
        - HMAC Key Size in bytes: >= 32 bytes
        - Tag Length in bytes: >= 32 bytes

6.2 The Function Test

6.2.1 Key Confirmation Not Supported

A separate file is generated for each supported key agreement scheme - role combination. For example, if an IUT supports the key agreement scheme dhHybrid1, and the IUT supports both initiator and responder roles, two files will be generated:

KASFunctionTest_FFCHybrid1_NOKC_init.req and
KASFunctionTest_FFCHybrid1_NOKC_resp.req.

Within each request file, there is a section for each combination of parameter set and SHA algorithm supported, i.e., FA-SHA1, FA-SHA224, FB-SHA224, FC-SHA512. For each combination of parameter set and SHA algorithm, the Function Test provides 10 sets of data to the IUT. In addition to this, if FFC is used, one set of domain parameter values is included for use with these 10 sets of data. If ECC is used, the curve name is included in the file header. Depending on the scheme being tested, this set of data may include a static public key and/or an ephemeral public key, and a nonce. The nonce is used in constructing the value of the MacData. (See Section 5.2.3 of NIST SP800-56A.)

The IUT uses the domain parameter values or the NIST-approved curves to generate a public/private key pair. The IUT uses the appropriate public keys supplied by the KASVS and its own public/private key pair to calculate the shared secret value *Z* and the derived keying material *DKM.* The *Z* value is computed using the appropriate DLC primitive corresponding to the scheme being tested (Section 5.7 of NIST SP800-56A). The *DKM* is computed using the supported KDF (Section 5.8 of NIST SP800-56A). Section 5.8 specifies two key derivation functions - the Concatenation Key Derivation Function (Approved Alternative 1) and the ASN.1 Key Derivation Function (Approved Alternative 2). These two functions differ only in the format of the Other Information *OtherInfo* (OI) field. In the KASVS, the IUT is required to supply the value of the OI field. This allows the CAVS tool to test both key derivation functions in the same manner. Other fields needed in the computation of the key derivation function are the IUTid, supplied by the IUT, and the CAVSid, supplied by the CAVS tool. Note that the accuracy of the format of the OI field is outside the scope of the KASVS validation testing.

The IUT computes a *Tag* to determine if the SP800-56A implementation has been implemented correctly. The IUT specifies an approved MAC algorithm supported by their implementation, i.e., CCM, CMAC, or HMAC. The MAC key is obtained from the *DKM.* The MacData to be MACed shall be the string “Standard Test Message” concatenated with the 16-byte nonce found in the request file (Section 5.2.3 of NIST SP800-56A).

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.

If the IUT indicates that they support full or partial validation of their keys, (denoted in the assurances), the KASVS will perform a validation of the IUT’s public keys. The KASVS will also verify the correctness of the IUT’s *Tag* by calculating the shared secret value using the appropriate DLC primitive and the IUT’s public keys, computing the derived keying material, and computing the *Tag*. The KASVS compares the IUT’s *Tag* value to the KASVS *Tag* value to see if they are the same. If they are, then it can be determined that the implemented key agreement scheme, the DLC primitive implementation, and the KDF implementation are 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 intermediate values generated by the CAVS, such as *Z* and *DKM*, are stored in the log file. The laboratory uses this information to assist the vendor in debugging their IUT.

6.2.2 Key Confirmation Supported

A separate file is generated for each supported combination of the key agreement scheme, key agreement role, key confirmation role and key confirmation type. For example, if an

IUT supports FFC cryptography, the dhStatic key agreement scheme, both key agreement roles (initiator and responder), both key confirmation roles (provider and recipient), and both key confirmation types (unilateral and bilateral), then eight files will be generated:

KASFunctionTest_FFCStatic_KC_init_prov_ulat.req
KASFunctionTest_FFCStatic_KC_init_rcpt_ulat.req
KASFunctionTest_FFCStatic_KC_init_prov_blat.req
KASFunctionTest_FFCStatic_KC_init_rcpt_blat.req
KASFunctionTest_FFCStatic_KC_resp_prov_ulat.req
KASFunctionTest_FFCStatic_KC_ resp _rcpt_ulat.req
KASFunctionTest_FFCStatic_KC_ resp _prov_blat.req
KASFunctionTest_FFCStatic_KC_ resp _rcpt_blat.req.

Within each *REQUEST* file, there is a section for each combination of parameter set and SHA algorithm supported, i.e., FA-SHA1, FA-SHA224, FB-SHA224, FC-SHA512. Within each combination of parameter set and SHA algorithm, the Function Test provides a section for each supported combination of MAC algorithm and key size, i.e., CCM AES128, CCM AES256. In addition to this, if FFC is used, one set of domain parameter values is included for use with these sets of data. If ECC is used, the curve name is included in the file header. In each MAC algorithm-key size section, the Function Test provides 10 sets of data to the IUT. Depending on the scheme being tested, this set of data may include a static public key and/or an ephemeral public key.

The IUT uses the domain parameter values or the NIST-approved curves to generate a public/private key pair. The IUT uses the appropriate public keys supplied by the KASVS and its own public/private key pair to calculate the shared secret value *Z* and he derived keying material DKM. The *Z* value is computed using the appropriate DLC primitive corresponding to the scheme being tested (Section 5.7 of NIST SP800-56A). The *DKM* is computed using the supported KDF (Section 5.8 of NIST SP800-56A). Section 5.8 specifies two key derivation functions - the Concatenation Key Derivation Function (Approved Alternative 1) and the ASN.1 Key Derivation Function (Approved Alternative 2). These two functions differ only in the format of the Other Information *OtherInfo* (OI) field. In the KASVS, the IUT is required to supply the value of the OI field. This allows the CAVS tool to test both key derivation functions in the same manner. Other fields needed in the computation of the key derivation function are the IUTid, supplied by the IUT, and the CAVSid, supplied by the CAVS tool. Note that the accuracy of the format of the OI field is outside the scope of the KASVS validation testing.

The IUT computes *Tags* for each implemented approved MAC algorithm supported by their implementation. These include CCM, CMAC, and/or HMAC. For each supported MAC algorithm, a MAC key will be obtained from the DKM. Depending on the key confirmation role (provider or recipient) and the key confirmation type (unilateral or bilateral), MacData will be computed as specified in Section 8 of NIST SP800-56A.

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.

If the IUT indicates that they support full or partial validation of their keys, (denoted in the assurances), the KASVS will perform a validation of the IUT's public keys. The KASVS will also verify the correctness of the IUT's *Tag* by calculating the shared secret value using the appropriate DLC primitive and the IUT's public keys, computing the derived keying material, computing the MacData value, and computing the *Tag*. The KASVS compares the IUT's *Tag* value to the KASVS *Tag* value to see if they are the same. If they are, then it can be determined that the implemented key agreement scheme, the DLC primitive implementation, and the KDF implementation are 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 intermediate values generated by the CAVS, such as *Z*, MacData, and *DKM*, are stored in the log file. The laboratory uses this information to assist the vendor in debugging their IUT.

6.3 The Validity Test

The second test in the NIST SP800-56A suite of validation tests is the Validity test. Its purpose is to test the ability of the IUT to recognize valid and invalid results received from the CAVS tool generated by the key agreement process with or without key confirmation. Incorrect values are generated by the CAVS tool by interjecting errors in different fields. The fields in which errors are introduced include *Z*, *DKM*, *OI*, *MacData*, *Tag*, CAVS' static public key, IUT's static public key, CAVS ephemeral pubic key and the IUT's static private key. Errors introduced in the keys test if the IUT has implemented the pubic key validation function properly. Note that this is only performed if the assurances supported by the IUT support this capability.

6.3.1 Key Confirmation Not Supported

A separate file is generated for each supported key agreement scheme - role combination. For example, if an IUT supports the ECC key agreement scheme dhFullUnified, and the IUT supports both initiator and responder roles, two files will be generated:

KASValidityTest_ECCFullUnif_NOKC_init.req and
KASValidityTest_ECCFullUnif_NOKC_resp.req.

Within each request file, there is a section for each combination of parameter set and SHA algorithm supported (for example, FA-SHA1, FA-SHA224, FB-SHA224, FC-SHA512). For each combination of parameter set and SHA algorithm, the Validity Test provides information identifying the domain parameter values (for FFC) or elliptic curve (ECC) being used. For FFC implementations, the KASVS will generate 24 sets of data for the IUT. For ECC implementations, the KASVS will generate 30 sets of data for the IUT. Within these sets of data, the KASVS will modify some of the values to introduce errors. This will determine whether or not the IUT can detect these errors. In addition to verifying that the IUT can detect errors in the key agreement and key confirmation processing, this test will also provide assurance of the validity of the domain parameters as implemented by the IUT.

Depending on the key agreement scheme being tested, data supplied by the CAVS includes:

1 A header containing:

   a. Parameter Sets Supported

   b. CAVSid

   c. IUTid

   d. The parameters associated with each parameter set, including:

      i. Curve selected (if ECC)

      ii. SHA(s) supported

      iii. MAC algorithm(s) supported

      iv. If the MAC is CCM:

         1. Key sizes supported

         2. CCM Nonce length

         3. CCM Tag length

      v. If the MAC is CMAC:

         1. Key sizes supported

         2. AES/TDES Tag length

      vi. If the MAC is HMAC:

         1. SHA(s) supported

         2. Key sizes supported

         3. Tag length

2 A set of data containing a subset of the following data depending on the scheme implemented:

   a. CAVS values, including:

      i. Static public key and/or

      ii. Ephemeral public key (or nonce)

   b. Nonce value

   c. IUT values, including:

i. Static private key, and

ii. Static public key, and/or

iii. Ephemeral private key, and

iv. Ephemeral public key

d. If CCM is selected: the CCMNonce value

e. Other Information, OI

f. CAVS Tag

The IUT uses this information to validate the CAVS Tag value, returning a PASS or FAIL. The IUT generates a response file containing the values above, plus the tag generated by the IUT (IUTTag) and the Result. The format for the *RESPONSE* file is specified in the *SAMPLE* file. There shall be a *RESPONSE* file for every *SAMPLE* file.

The KASVS verifies that the correct responses were returned by the IUT by comparing the results in the *RESPONSE* file with those in the *FAX* file. If the results match, CAVS records PASS for this test; otherwise, CAVS records FAIL.

6.3.2 Key Confirmation Supported

A separate file is generated for each supported combination of the key agreement scheme, key agreement role, key confirmation role and key confirmation type. For example, if an IUT supports the FFC key agreement scheme dhHybrid1, the key agreement role of initiator, both key confirmation roles (provider and recipient), and both key confirmation types (unilateral and bilateral), four files will be generated:

KASValidityTest_FFCHybrid1_KC_init_prov_ulat.req
KASValidityTest_FFCHybrid1_KC_init_prov_blat.req

KASValidityTest_FFCHybrid1_KC_init_rcpt_ulat.req
KASValidityTest_FFCHybrid1_KC_init_rcpt_blat.req

Within each *REQUEST* file, there is a section for each combination of parameter set and SHA algorithm supported, i.e., FA-SHA1, FA-SHA224, FB-SHA224, FC-SHA512. Within each combination of parameter set and SHA algorithm, the Validity Test provides a section for each supported combination of MAC algorithm and key size, i.e., CCM AES128, CCM AES256. In addition to this, if FFC is used, one set of domain parameter values is included for use with these sets of data. If ECC is used, the curve name is included in the file header.

In each MAC algorithm-key size section, the Validity Test generates 24 sets of data for

FFC implementations and 30 sets of data for ECC implementations. Within these sets of data, the KASVS alters some of the values to introduce errors. This will determine whether or not the IUT can detect these errors. In addition to verifying that the IUT can detect errors in the key agreement and key confirmation processing, this test will also provide assurance of the validity of the domain parameters if implemented by the IUT.

Depending on the key agreement scheme being tested, data supplied by the CAVS includes:

3 A header containing:

   a. Parameter Sets Supported

   b. CAVSid

   c. IUTid

   d. Key Confirmation Types Supported

   e. The parameters associated with each parameter set, including:

      i. Curve selected (if ECC)

      ii. SHA(s) supported

      iii. MAC algorithm(s) supported

      iv. If the MAC is CCM:

         1. Key sizes supported

         2. CCM Nonce length

         3. CCM Tag length

      v. If the MAC is CMAC:

         1. Key sizes supported

         2. AES/TDES Tag length

      vi. If the MAC is HMAC:

         1. SHA(s) supported

         2. Key sizes supported

3. Tag length

4 A set of data containing a subset of the following data depending on the scheme implemented:

   a. CAVS values, including:

      i. Static public key and/or

      ii. Ephemeral public key (or nonce)

   b. IUT values, including:

      i. Static private key, and

      ii. Static public key, and/or

      iii. Ephemeral private key, and

      iv. Ephemeral public key

   c. If CCM is selected: the CCMNonce value

   d. Other Information, OI

   e. CAVS Tag

The IUT uses this information to validate the CAVS Tag value returning a PASS or FAIL. The IUT generates a response file containing the values above, plus the tag generated by the IUT (IUTTag) and the Result. The format for the *RESPONSE* file is specified in the *SAMPLE* file. There shall be a *RESPONSE* file for every *SAMPLE* file.

The KASVS compares the contents of the *RESPONSE* file with the contents of the *FAX* file. If the results match, CAVS records PASS for this test; otherwise, CAVS records FAIL.

Appendix A References

[1] *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, Special Publication 800-56A, National Institute of Standards and Technology, March 2006.

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