

ACVP KDA TwoStep Specification

Russell Hammett
HII Technical Solutions Division
302 Sentinel Drive, Suite #300, Annapolis Junction, MD 20701

December 11, 2020

Abstract

This document defines the JSON schema for testing KDA-TwoStep SP800-56C implementations with the ACVP specification.

Keywords

The following are keywords to be used by search engines and document catalogues.

ACVP; cryptography

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

Additional Information

For additional information on NIST's Cybersecurity programs, projects and publications, visit the [Computer Security Resource Center](#). Information on other efforts at [NIST](#) and in the [Information Technology Laboratory](#) (ITL) is also available.

Feedback

Feedback on this publication is welcome, and can be sent to: code-signing@nist.gov.

1. Introduction

The Automated Crypto Validation Protocol (ACVP) defines a mechanism to automatically verify the cryptographic implementation of a software or hardware crypto module. The ACVP specification defines how a crypto module communicates with an ACVP server, including crypto capabilities negotiation, session management, authentication, vector processing and more. The ACVP specification does not define algorithm specific JSON constructs for performing the crypto validation. A series of ACVP sub-specifications define the constructs for testing individual crypto algorithms. Each sub-specification addresses a specific class of crypto algorithms. This sub-specification defines the JSON constructs for testing KDA-TwoStep SP800-56C implementations using ACVP.

2. Supported KDA TwoStep

The following key derivation algorithms **MAY** be advertised by the ACVP compliant cryptographic module:

- KDA / TwoStep / SP800-56Cr1
- KDA / TwoStep / SP800-56Cr2

3. Test Types and Test Coverage

The ACVP server performs a set of tests on the KDAs specific to the KAS protocol in order to assess the correctness and robustness of the implementation. A typical ACVP validation session **SHALL** require multiple tests to be performed for every supported permutation of KDA capabilities. This section describes the design of the tests used to validate implementations of KDA algorithms.

3.1. Test Types

There are two test types for KDA testing:

- “AFT”—Algorithm Function Test. In the AFT test mode, the IUT **SHALL** act as a party in the Key Agreement with the ACVP server. The server **SHALL** generate and provide all necessary information for the IUT to perform a successful key agreement; both the server and IUT **MAY** act as party U/V.
- “VAL”—Validation Test. In the VAL test mode, The ACVP server **MUST** generate a complete (from both party U and party V’s perspectives) key agreement, and expects the IUT to be able to determine if that agreement is valid. Various types of errors **MUST** be introduced in varying portions of the key agreement process that the IUT **MUST** be able to detect and report on.

3.2. Test Coverage

The tests described in this document have the intention of ensuring an implementation is conformant to [\[SP 800-56C Rev. 1\]](#) and [\[SP 800-56C Rev. 2\]](#).

3.2.1. Requirements Covered

- SP 800-56C—5 Two-Step Key Derivation. All functionality described in the specification is covered by ACVP testing.

3.2.2. Requirements Not Covered

- SP 800-56Ar3 / SP 800-56Br2—ASN.1 encoding for the KDA is not currently supported.

4. Capabilities Registration

ACVP requires crypto modules to register their capabilities. This allows the crypto module to advertise support for specific algorithms, notifying the ACVP server which algorithms need test vectors generated for the validation process. This section describes the constructs for advertising support of KDA-TwoStep SP800-56C algorithms to the ACVP server.

The algorithm capabilities **MUST** be advertised as JSON objects within the ‘algorithms’ value of the ACVP registration message. The ‘algorithms’ value is an array, where each array element is an individual JSON object defined in this section. The ‘algorithms’ value is part of the ‘capability_exchange’ element of the ACVP JSON registration message. See the ACVP specification [\[ACVP\]](#) for more details on the registration message.

4.1. Prerequisites

Each algorithm implementation **MAY** rely on other cryptographic primitives. For example, RSA Signature algorithms depend on an underlying hash function. Each of these underlying algorithm primitives must be validated, either separately or as part of the same submission. ACVP provides a mechanism for specifying the required prerequisites:

Prerequisites, if applicable, **MUST** be submitted in the registration as the `prereqVals` JSON property array inside each element of the `algorithms` array. Each element in the `prereqVals` array **MUST** contain the following properties

Table 1 — Prerequisite Properties

JSON Property	Description	JSON Type
algorithm	a prerequisite algorithm	string
valValue	algorithm validation number	string

A “valValue” of “same” **SHALL** be used to indicate that the prerequisite is being met by a different algorithm in the capability exchange in the same registration.

An example description of prerequisites within a single algorithm capability exchange looks like this

```
"prereqVals":  
[  
  {  
    "algorithm": "Alg1",  
    "valValue": "Val-1234"  
  },  
  {  
    "algorithm": "Alg2",  
    "valValue": "same"  
  }  
]
```

]

Figure 1

4.2. Property Registration

The KDA-TwoStep SP800-56C mode capabilities are advertised as JSON objects within a root “algorithm” object.

A registration **SHALL** use these properties:

Table 2 — Registration Properties

JSON Value	Description	JSON Type	Valid Values
algorithm	The algorithm under test	value	KDA
mode	The mode under test	value	TwoStep
revision	The algorithm testing revision to use.	value	“Sp800-56Cr1” or “Sp800-56Cr2”
prereqVals	Prerequisite algorithm validations	array of prereqAlgVal objects	See Section 4.2.1
capabilities	The two step configuration objects	array of Section 4.2.2	See Section 4.2.2
z	The domain of values representing the min/max lengths of Z the implementation can support.	Domain	
l	The length (in bits) of the largest derived keying material the implementation can produce (up to a max of 2048).	number	
performMultiExpansionTests	Should multi expansion runs of the KDA be tested (in addition to the single expansion tests)? Only applicable for [SP 800-56C Rev. 2]	boolean	true, false

4.2.1. Prerequisite Algorithms for KDA Validations

Some algorithm implementations rely on other cryptographic primitives. For example, IKEv2 uses an underlying SHA algorithm. Each of these underlying algorithm primitives must be validated, either separately or as part of the same submission. ACVP provides a mechanism for specifying the required prerequisites:

Table 3 — Prerequisite Algorithms

JSON Value	Description	JSON Type	Valid Values
algorithm	a prerequisite algorithm	value	CMAC, DRBG, HMAC, SHA
valValue	algorithm validation number	value	actual number or “same”
prereqAlgVal	prerequisite algorithm validation	object with algorithm and valValue properties	see above

4.2.2. Two Step KDA Capabilities

Table 4 — TwoStepCapabilities Options

JSON Value	Description	JSON Type	Valid Values
macSaltMethod	How the salt is determined (default being all 00s, random being a random salt).	array of string	default, randombased auxiliary functions.
fixedInfoPattern	The pattern used for fixedInfo construction.	string	See Section 4.2.3
encoding	The encoding type to use with fixedInfo construction. Note concatenation is currently supported. ASN.1 should be coming.	array of string	concatenation
kdfMode	The strategy for running the KDA.	string	counter, feedback, double pipeline iteration
macMode	The macMode supported by the KDA.	array of string	CMAC-AES128, CMAC-AES192, CMAC-AES256, HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-

JSON Value	Description	JSON Type	Valid Values
			SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512
fixedDataOrder	The counter locations supported by the KDA.	array of string	none, before fixed data, after fixed data, before iterator
counterLength	The counter lengths supported for the KDA.	array of integer	8, 16, 24, 32
supportedLengths	The supported derivation lengths.	domain	Single range (of literal) expected. Registered value must support the L value provided.
supportsEmptyIv	The KDA supports an empty IV (feedback mode).	boolean	true, false
requiresEmptyIv	The KDA requires an empty IV (feedback mode).	boolean	true, false

Note this capabilities object is very similar to the capability object from SP800-108.

4.2.3. FixedInfoPatternConstruction

IUTs **MUST** be capable of specifying how the FixedInfo is constructed for the KDA construction. Note that for the purposes of testing against the ACVP system, both uPartyInfo and vPartyInfo are **REQUIRED** to be registered within the fixed info pattern.

Pattern candidates:

- literal[0123456789ABCDEF]
 - uses the specified hex within “[]”. literal[0123456789ABCDEF] substitutes “0123456789ABCDEF” in place of the field
- uPartyInfo
 - uPartyId { || ephemeralKey } { || ephemeralNonce } { || dkmNonce } { || c }
 - “optional” items such as ephemeralKey **MUST** be included when available for ACVP testing.

- vPartyInfo
 - vPartyId { || ephemeralKey } { || ephemeralNonce } { || dkmNonce } { || c }
 - “optional” items such as ephemeralKey **MUST** be included when available for ACVP testing.
- context
 - Random value chosen by ACVP server to represent the context.
- algorithmId
 - Random value chosen by ACVP server to represent the algorithmId.
- label
 - Random value chosen by ACVP server to represent the label.
- l
 - The length of the derived keying material in bits, **MUST** be represented in 32 bits for ACVP testing.
- t
 - A random value used to represent a secondary shared secret. Only applicable to [\[SP 800-56C Rev. 2\]](#).

Example (Note that party U is the server in this case “434156536964”, party V is the IUT “a1b2c3d4e5”):

- “concatenation” : “literal[123456789CAFECAFE]||uPartyInfo||vPartyInfo”

Evaluated as:

- “123456789CAFECAFE434156536964a1b2c3d4e5”

4.3. Registration Example

```
{
  "algorithm": "KDA",
  "mode": "TwoStep",
  "revision": "Sp800-56Cr1",
  "prereqVals": [
    {
      "algorithm": "DRBG",
      "valValue": "123456"
    },
    {
      "algorithm": "SHA",
      "valValue": "123456"
    }
  ],
}
```

```

    {
      "algorithm": "HMAC",
      "valValue": "123456"
    }
  ],
  "capabilities": [
    {
      "macSaltMethods": [
        "random",
        "default"
      ],
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
      "encoding": [
        "concatenation"
      ],
      "kdfMode": "feedback",
      "macMode": [
        "HMAC-SHA3-224",
        "HMAC-SHA2-512"
      ],
      "supportedLengths": [
        512
      ],
      "fixedDataOrder": [
        "after fixed data",
        "before iterator"
      ],
      "counterLength": [
        32
      ],
      "requiresEmptyIv": false,
      "supportsEmptyIv": false
    }
  ],
  "l": 2048,
  "z": [
    512
  ]
}

```

Figure 2 — Registration JSON Example SP800-56Cr1

```

{
  "algorithm": "KDA",
  "mode": "TwoStep",

```

```
"revision": "Sp800-56Cr2",
"prereqVals": [
  {
    "algorithm": "DRBG",
    "valValue": "123456"
  },
  {
    "algorithm": "SHA",
    "valValue": "123456"
  },
  {
    "algorithm": "HMAC",
    "valValue": "123456"
  }
],
"performMultiExpansionTests": true,
"capabilities": [
  {
    "macSaltMethods": [
      "random",
      "default"
    ],
    "fixedInfoPattern": "uPartyInfo||vPartyInfo||t||l",
    "encoding": [
      "concatenation"
    ],
    "kdfMode": "feedback",
    "macMode": [
      "HMAC-SHA3-224",
      "HMAC-SHA2-512"
    ],
    "supportedLengths": [
      512
    ],
    "fixedDataOrder": [
      "after fixed data",
      "before iterator"
    ],
    "counterLength": [
      32
    ],
    "requiresEmptyIv": false,
    "supportsEmptyIv": false
  }
],
"l": 2048,
```

```
"z": [  
  512  
]  
}
```

Figure 3 — Registration JSON Example SP800-56Cr2

5. Test Vectors

The ACVP server provides test vectors to the ACVP client, which are then processed and returned to the ACVP server for validation. A typical ACVP validation test session would require multiple test vector sets to be downloaded and processed by the ACVP client. Each test vector set represents an individual algorithm defined during the capability exchange. This section describes the JSON schema for a test vector set used with KDA-TwoStep SP800-56C algorithms.

The test vector set JSON schema is a multi-level hierarchy that contains meta data for the entire vector set as well as individual test vectors to be processed by the ACVP client. The following table describes the JSON elements at the top level of the hierarchy.

Table 5 — Top Level Test Vector JSON Elements

JSON Values	Description	JSON Type
acvVersion	Protocol version identifier	string
vsId	Unique numeric vector set identifier	integer
algorithm	Algorithm defined in the capability exchange	string
mode	Mode defined in the capability exchange	string
revision	Protocol test revision selected	string
testGroups	Array of test groups containing test data, see Section 5.1	array

An example of this would look like this

```
{
  "acvVersion": "version",
  "vsId": 1,
  "algorithm": "Alg1",
  "mode": "Model",
  "revision": "Revision1.0",
  "testGroups": [ ... ]
}
```

Figure 4

5.1. Test Groups JSON Schema

The testGroups element at the top level in the test vector JSON object is an array of test groups. Test vectors are grouped into similar test cases to reduce the amount of data transmitted in the vector set. For instance, all test vectors that use the same key size would be grouped together. The Test Group JSON object contains meta data that applies to all test vectors within the group. The following table describes the KDA-TwoStep SP800-56C JSON elements of the Test Group JSON object

Table 6 — Test Group Properties

JSON Values	Description	JSON Type
tgId	Test group identifier	integer
testType	Describes the operation the client should perform on the tests data	string
tests	Array of individual test cases	See Section 5.2
kdfConfiguration	Describes the KDA configuration values used for the group	See Section 5.1.1
kdfConfiguration	Describes the KDA configuration values used for single expansion groups.	See Section 5.1.1
kdfMultiExpansionConfiguration	Describes the KDA configuration values used for multi expansion groups.	See Section 5.1.2

The ‘tgId’, ‘testType’ and ‘tests’ objects **MUST** appear in every test group element communicated from the server to the client as a part of a prompt. Other properties are dependent on which ‘testType’ the group is addressing.

5.1.1. KDA Configuration JSON Schema

Describes the KDA configuration for use under the test group.

Table 7 — KdfConfiguration JSON Object

JSON Value	Description	JSON Type
kdfType	The type of KDA to use for the group.	value — twostep
satMethod	The strategy used for salting.	value — default (all 00s), random
fixedInfoPattern	The pattern used for constructing the fixedInfo.	value — See Section 4.2.3 .
fixedInfoEncoding	The pattern used for constructing the fixedInfo.	value — See Section 4.2.3 .
macMode	The MAC function used in the KDA.	value
counterLocation	The counter location.	value
counterLen	The counter length.	value
ivLen	The iv length.	value

JSON Value	Description	JSON Type
l	the bit length of keying material to derive from the KDA	value

5.1.2. KDA Multi Expansion Configuration JSON Schema

Describes the KDA Multi Expansion configuration for use under the test group.

Table 8 — KdfConfiguration JSON Object

JSON Value	Description	JSON Type
kdfType	The type of KDA to use for the group.	value# — #TwoStep
saltMethod	The strategy used for salting.	value — default (all 00s), random
saltLen	The bit length of the salt.	integer
hmacAlg	The MAC function used in the KDA.	value
l	the bit length of keying material to derive from the KDA	value

5.2. Test Case JSON Schema

Each test group contains an array of one or more test cases. Each test case is a JSON object that represents a single test vector to be processed by the ACVP client. The following table describes the JSON elements for each KAS/KTS ECC test vector.

Table 9 — Test Case JSON Object

JSON Value	Description	JSON Type
tcId	Numeric identifier for the test case, unique across the entire vector set.	kdfParameter
Object representing inputs into the KDA for single expansion tests.	See Section 5.2.1.	fixedInfoPartyU
Fixed information specific to party U for single expansion tests.	See Section 5.2.2.	fixedInfoPartyV
Fixed information specific to party V for single expansion tests.	See Section 5.2.2.	kdfMultiExpansionParameter

5.2.1. KDA Parameter JSON Schema

KDA specific options used for the test case.

Table 10 — KDF Parameter JSON Object

JSON Value	Description	JSON Type
kdfType	The type of KDA utilized.	value
salt	The salt used for the test case.	value
iv	The iv used for the test case.	value
algorithmId	The random “algorithmID” used for the test case when applicable to the fixedInfo pattern.	value
context	The random “context” used for the test case when applicable to the fixedInfo pattern.	value
label	The random “label” used for the test case when applicable to the fixedInfo pattern.	value
z	shared secret z value to be used for the test case.	value
l	the bit length of keying material to derive from the KDA	value

5.2.2. FixedInfo PartyU/V JSON Schema

Fixed information that is included for party U/V for fixed info construction

Table 11 — Fixed Info JSON Object

JSON Value	Description	JSON Type
partyId	The party identifier	value
ephemeralData	Ephemeral data (randomly) included as a part of the parties fixed info construction	value

5.2.3. KDA Multi Expansion Parameter JSON Schema

KDA specific options used for the test case.

Table 12 — KDF Multi Expansion Parameter JSON Object

JSON Value	Description	JSON Type
salt	The salt used for the test case.	value
z	shared secret z value to be used for the test case.	value

iterationParameters	the per iteration parameters for multi expansion	See Section 5.2.3.1 .
---------------------	--	---------------------------------------

5.2.3.1. KDA Multi Expansion IterationParameters JSON Schema

The per multi expansion iteration specific parameters used within a test case.

Table 13 — KDF Multi Expansion IterationParameters JSON Object

JSON Value	Description	JSON Type
l	The length of keying material to derive for the current iteration.	value
fixedInfo	The fixed information for the current iteration. Note that [SP 800-56C Rev. 2] does not go into detail regarding how this per iteration fixed info should be constructed, but it obviously needs to be different each iteration.	value

5.3. Example Test Vectors JSON

The following is a example JSON object for KAS-FFC test vectors sent from the ACVP server to the crypto module.

```
{
  "vsId": 0,
  "algorithm": "KDA",
  "mode": "TwoStep",
  "revision": "Sp800-56Cr1",
  "isSample": true,
  "testGroups": [
    {
      "tgId": 1,
      "testType": "AFT",
      "tests": [
        {
          "tcId": 1,
          "kdfParameter": {
            "kdfType": "twoStep",
            "salt":
"0ADFBF33C04265B205163B6CDE97BB1B29FE1821B56E7022A74E4054ADC52AC3F339219C6B4C9B158EE95ED4
"z":
"576F1B40A4A640C72DD070C5D80380558D1A642DEEAA8F1978428ADE63D7DA4553780E25103F97E493F2626A
"l": 512,
            "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
            "fixedInputEncoding": "concatenation",
```

```

        "iv":
"3A29BC54ADE2C0AD617FFD70112445F87E15B21D67156E10CCBC07D0ECF8649163738EFEEFFAC01488875B8A
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32
    },
    "fixedInfoPartyU": {
        "partyId": "806604B8DA9F7E391F8307486CD67D0A"
    },
    "fixedInfoPartyV": {
        "partyId": "20FE389DE49E9CA909D48E9F0BA5BAB2",
        "ephemeralData":
"02A918E3A64B8BC17907569DAA75BEC4DA331303460127A86C6CCEF2B62164BAA1459C3AE46A9E6302EE6A4B
    }
},
{
    "tcId": 2,
    "kdfParameter": {
        "kdfType": "twoStep",
        "salt":
"BE4FEA6C28C7F6D571B0EC89CAB2E586053CDE9208CA409D164F03FD58938CFB47E63C3BE589DD512A6F659E
        "z":
"025E3E3EF6B67354DE9778F793214A7A9CC8C0DD87DA70E27F7C47E63F9E33B89AB20AB6339FFBD8F827C19F
        "l": 512,
        "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
        "fixedInputEncoding": "concatenation",
        "iv":
"375FA1A86CD4CD1A7833F88BE886545D6ED652BC3F6B173B5A768B01BA740BF1B3AA263B078321DC0FBD1B8A
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32
    },
    "fixedInfoPartyU": {
        "partyId": "181AA0E72E042B9803A213DF7CBB8051",
        "ephemeralData":
"C933CCD623CBC35246E713BD690F1C5FEA12722F281EE07BDE3219F2843BC29E58B4F76873CCFB1DA866BC4C
    },
    "fixedInfoPartyV": {
        "partyId": "AA783D9D6B897E04BE1F2004C3F21A35",
        "ephemeralData":
"4B5F015CE5974ED4FB992A634A47BEDA77F470EFEE10E7667FEAE1E7B8F53A204B15A97B893E6551FD3FE64F
    }
},
{

```

```

    "tcId": 3,
    "kdfParameter": {
      "kdfType": "twoStep",
      "salt":
"3040A6C24E8CDA5E53DDF7C42335FA82383FC3B1F2C95E3D335D9720BD3AB251DF07712EF3824ECDB00C7267
      "z":
"653DFF10216BDC7C6817F78FAE59FF15400061951EEB9AF309819AE7A2904829C29127688FC1A1A40F62F9CE
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
      "fixedInputEncoding": "concatenation",
      "iv":
"E680F9C99500ED6AB6F40D35783F81DC44AAD381F12EAABB5ABD697C2C57981D97710E0A465EE6D5591BE43C
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32
    },
    "fixedInfoPartyU": {
      "partyId": "427A430E34956099A11A34996C68B585",
      "ephemeralData":
"CE4FCE03CD168399D90D499BA988D7A16738730F0E713B3783A297D3858F7F28623959E2622F15B7E5E0A793
    },
    "fixedInfoPartyV": {
      "partyId": "44D5D13558C15CF6EE886E69E005A8EC"
    }
  },
  {
    "tcId": 4,
    "kdfParameter": {
      "kdfType": "twoStep",
      "salt":
"F9916CA1F2B351D833AD84391A48DC3753D21E156C32F961F26897376A3ED5AB18689B99B3CF1BFE55EDF56A
      "z":
"BAF85001AAF675464043635BEC546B935A6542DCF401B3744ED1E633F912827A925869FD0DC1DA9392042ACD
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
      "fixedInputEncoding": "concatenation",
      "iv":
"10743126EF5675BA21A5D4F025A8D21157703C447D93EFC47F5853827CFC3967F55076E3E6910B2726DE4E53
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32
    },
    "fixedInfoPartyU": {
      "partyId": "50B610281A9C8768B5B7E84E39E866A4"
    }
  }

```

```

    },
    "fixedInfoPartyV": {
      "partyId": "BB234C26EFBEB3D13F17366B14917967",
      "ephemeralData":
"42D9BB758F33884617A1554CB1F149FE7CB9164F62E775DDCF5B48F640C0E15E783A41C5F9357CF9D12F38A5
    }
  },
  {
    "tcId": 5,
    "kdfParameter": {
      "kdfType": "twoStep",
      "salt":
"956E5333A9A29004CB4C3CAC3750FACCB46106900B7E7CDA166D6DCB7BE8A2679F195203C4DDD8E34EF6E90D
      "z":
"6F3EDA174C2DD8C611D126300C6AB0D61D6AEB804D527C87801BDE3AD3F6AD2C8B7E4976633AB154A92AD791
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
      "fixedInputEncoding": "concatenation",
      "iv":
"632C996B274A3C313C93645B0B32C5D21960BCC4DF105924E970C19F6BEDE80E86E3C4CA3B771D552AF3E101
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32
    },
    "fixedInfoPartyU": {
      "partyId": "6929DD85147EE842DE5F921E247B598F"
    },
    "fixedInfoPartyV": {
      "partyId": "3DAC69CEB1620DE4DB065A71DEC68C1C",
      "ephemeralData":
"555F54BEC5FCF9269F53C55E6EC8EEB324581E5D12A75274481D01FA0320A1D6D17E742CFD85394975FF1E90
    }
  }
],
"kdfConfiguration": {
  "kdfType": "twoStep",
  "l": 512,
  "saltLen": 512,
  "saltMethod": "random",
  "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
  "fixedInfoEncoding": "concatenation",
  "kdfMode": "feedback",
  "macMode": "HMAC-SHA2-512",
  "counterLocation": "after fixed data",
  "counterLen": 32,

```

```

        "ivLen": 512
    }
},
{
    "tgId": 33,
    "testType": "VAL",
    "tests": [
        {
            "tcId": 161,
            "kdfParameter": {
                "kdfType": "twoStep",
                "salt":
"A12858E5715B0A82E684B2D4433768BD56FD2FB701AB7367B0095DEBB1511FB467AEDBE9E70DFEBC4E537557
                "z":
"366F5863F9B2F6F56FCEDD44D1CF1585C83D7CFBC146C0B8A5835D05F9F4BF1225ECB140BD6BB30CA3DF9B27
                "l": 512,
                "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
                "fixedInputEncoding": "concatenation",
                "iv":
"F611B1D01C10B85CF2FDA62E2E70CE73301D9FCEFFB523D6BDCE5176C8A50A4A0ECA9702E02F6A9B01926AA4
                "kdfMode": "feedback",
                "macMode": "HMAC-SHA2-512",
                "counterLocation": "after fixed data",
                "counterLen": 32
            },
            "fixedInfoPartyU": {
                "partyId": "8FA71E55677BC77BEFF5399FF56E8580"
            },
            "fixedInfoPartyV": {
                "partyId": "505F428AAB0F0385F035E11AE22A38A2",
                "ephemeralData":
"649B3A141042D4D2DBBB0A40820688C02B6C8AD363C50209ED792E8131DC45171B85F144E636190539A46553
            },
            "dkm":
"ABD671702D3E4F6040DD4EF8CE7F93C048071AA2A4857E9D90DF14EADA758E5CCB702A14039EA0BDE758F688
        },
        {
            "tcId": 162,
            "kdfParameter": {
                "kdfType": "twoStep",
                "salt":
"45A57043BECD4048EE868278CF78907DB3C3D6B2740E4C36FFD6CE98775D7966AA80DB17ACFD9DD2D47B8573
                "z":
"E8257C5D497449D2BA04D3FB79A0BD15FAA76511DD726E80211A8EC6830EBB64CAAEA785704C3AC68FCE36A1
                "l": 512,
                "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",

```

```

        "fixedInputEncoding": "concatenation",
        "iv":
"F33F177DDBFAE1F942E21D41FAE4D01E4177604A34C43721D068D6108781C9A7C5BC1701572E6FF3E331B163
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32
    },
    "fixedInfoPartyU": {
        "partyId": "2ED2A1A8514BB3302CB0C0501C325C0B"
    },
    "fixedInfoPartyV": {
        "partyId": "BA287A7504EDEAE60FE82BC1A974E793",
        "ephemeralData":
"A68EF3B1402B3A2A8E1C075302196DCB9DC99F240061F2BD5FE8B1AA77C9E2962239BA28C3D66A0D4A3049A9
    },
    "dkm":
"E24017258025637EB0175E0916B5C8152AE22185269096B7AD03A21EFB3DFB8AF863A2D574F5F8C91B5B7E59
    },
    {
        "tcId": 163,
        "kdfParameter": {
            "kdfType": "twoStep",
            "salt":
"4F08A21287BA7AD996B25B9BEED8D734A09E2A958050B6FCE024C6F302B4652C129834785ADF840B9A6BF83E
            "z":
"4FA641D15099935020D57449CF2DFE5D9430BCD85BBB1EF4367F474FA699AD7CC6FAC9DCA26D1F0D46DCF9FC
            "l": 512,
            "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
            "fixedInputEncoding": "concatenation",
            "iv":
"D1A84B095C875B585BF3677DD4A0D875EFEE2AD3B37F2412032563FDC57A00AD5ACD9D0B78281EB3D177AA5C
            "kdfMode": "feedback",
            "macMode": "HMAC-SHA2-512",
            "counterLocation": "after fixed data",
            "counterLen": 32
        },
        "fixedInfoPartyU": {
            "partyId": "516D813DDD3ECBBAE738572A7DBBC827",
            "ephemeralData":
"95E1D0001556173E6F3C505D4DD0658B668B397C01B2A4B44AFBC3C9FCF8604826CFE3F7367D1445C50669B3
        },
        "fixedInfoPartyV": {
            "partyId": "0C4804B70E4CC537B9AFB66CCE659586",
            "ephemeralData":
"D2F7F4316DFADBD61CCD3761899A4B697B515CFDF8E2AA6DCC03834F25EA8C4ACE39B72D03971517B6D4FABC

```

```

    },
    "dkm":
"48B0E4139CAF7818490A0FED0AA83509B1E665EA84D6748CC7E03FFB7B4F693D71B9A2B4DAA43F4D9DBA957B
  },
  {
    "tcId": 164,
    "kdfParameter": {
      "kdfType": "twoStep",
      "salt":
"0FA2DB567054A3FD5B4C80F7BC3EB7FDFEFEB2DE24A7E15B9825AAC7965CEF61A725D5EA530C7F9239EED953D
      "z":
"AD5B5575642BF653381470A00D261602A5444C19EFFDFB7965AE9A70911C9611C35DE20B8F0C467EBE359E88
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
      "fixedInputEncoding": "concatenation",
      "iv":
"9D6A8E1B559B812DB1F0B4A88D625EE75943241AF3347C527937A61139F76E73DE2CEA58D66872A9997BA01F
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32
    },
    "fixedInfoPartyU": {
      "partyId": "31A445A08D8089609314ED6CD691D75B",
      "ephemeralData":
"4BF9B00CA328F2792D937663B051ED5B9916249AAF8854F0B40AF266C70E63F9510D9787F016AC920214396E
    },
    "fixedInfoPartyV": {
      "partyId": "6E6764B019CA8D8A5D395DE0E6752830",
      "ephemeralData":
"6357D8183753E697C365BFF5E672BF8AE52394C50FFF15F23AD3045F1A900DBA8B147D79ABD526F3B4FEB930
    },
    "dkm":
"647A0008C671EE83B17A32887ABA2DFD190C33AAE20537599A048171A8B6B760F6674E18E5E465F0104CFBB1
  },
  {
    "tcId": 165,
    "kdfParameter": {
      "kdfType": "twoStep",
      "salt":
"DF6847F01DF268B26A95870776389BE7E74F3E0E58E4EBB169CF9C6CF0B3071096CA8722FCD33BF455C4ADAC
      "z":
"E16F10272E509C404C4669428313CE083196F8CD49FAB74E64EFDD972D0933DE4DAF2809DD952873A1377581
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||l",
      "fixedInputEncoding": "concatenation",

```



```

        "iv":
"C8D7247CBB6AEBF5EE90E13203303B2145A1D5A5CBFB6722CE373C98A1604836A79B2AA043E4871898E534E
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32
    },
    "fixedInfoPartyU": {
        "partyId": "7AD3ED7BC9207D09C904B6B924B9C639",
        "ephemeralData":
"EA712C47B0CA7183EB1F3BCBDD6282E240473D09C9FD3644BA0DCD343AC49CD6C852D17C48C47A4AC9517872
    },
    "fixedInfoPartyV": {
        "partyId": "0CC0877A3BFB4C024B12037825B6339A",
        "ephemeralData":
"68995A7D9A588A87BC1184EA586876E3D9A5484F4C9FA81B36247B77C6C37DBDB1FB70E3407FDA166FDFC722
    },
    "dkm":
"0A3A1C08B55A60DA29E84FB82A54FA386008B07F74F192E179E99ED509E6AEB6580BA7B6E8D3DAC7E0E0A93D
    }
],
    "kdfConfiguration": {
        "kdfType": "twoStep",
        "l": 512,
        "saltLen": 512,
        "saltMethod": "random",
        "fixedInfoPattern": "uPartyInfo||vPartyInfo||1",
        "fixedInfoEncoding": "concatenation",
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32,
        "ivLen": 512
    }
}
]
}

```

Figure 5 — Vector Set JSON Example SP800-56Cr1

```

{
    "vsId": 0,
    "algorithm": "KDA",
    "mode": "TwoStep",
    "revision": "Sp800-56Cr2",

```

```

    "isSample": true,
    "testGroups": [
      {
        "tgId": 1,
        "testType": "AFT",
        "tests": [
          {
            "tcId": 1,
            "kdfParameter": {
              "kdfType": "twoStep",
              "salt":
"73669F6D823C5AAB0C3E2B8CF7E8BDD71D4E659BCD6CFF92CD6C9708D2232470100A0E7513A1D932FDEAC0DE
            "t": "86B6B4E25737D8B42CAE4B135A46AA5E",
            "z":
"B5730243292A071FFF6CD3448D4644C01BD59435BBAE5B1BE7E7D2D9B2D8824C4CCEF5778C33CAC0AD7A5D19
            "l": 512,
            "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
            "fixedInputEncoding": "concatenation",
            "iv":
"DBBE8E4B36080300B04DAE64E8F66FF88EB85F8ECE30CE8063CF5F1D1609F999F25FEA5BAD2716E544831326
            "kdfMode": "feedback",
            "macMode": "HMAC-SHA2-512",
            "counterLocation": "after fixed data",
            "counterLen": 32
          },
          "fixedInfoPartyU": {
            "partyId": "2C78BEAFBDC7896D3E1865C0D4CEC989"
          },
          "fixedInfoPartyV": {
            "partyId": "E719A3BEA7D292D815790740AF32CB18"
          }
        ],
        {
          "tcId": 2,
          "kdfParameter": {
            "kdfType": "twoStep",
            "salt":
"29DEC0ED49BD992D49D2760D463234815F663D0CE5A2957B5B733DC573714F882437057801D5185F19160955
            "t": "69C874B72E75FC0895A815DFEE0870FC",
            "z":
"F8F7368992EF282A515FEFCE6397E1346346E11B30CCECF605502A9A718363EA89EA2925446E00F3771DF409
            "l": 512,
            "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
            "fixedInputEncoding": "concatenation",
            "iv":
"E40E3ABAC9CDB2FEA2579A4176969C8CFCE2124AA238A4CE826D1C79F75178B7BB6D297F2B45287C1216F0D2

```

```

        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32
    },
    "fixedInfoPartyU": {
        "partyId": "115BED271DE2A0036BF85621EBDC6FC4"
    },
    "fixedInfoPartyV": {
        "partyId": "F5C07CBE3A72463203FBBB879144100A"
    }
},
],
"kdfConfiguration": {
    "kdfType": "twoStep",
    "l": 512,
    "saltLen": 512,
    "saltMethod": "random",
    "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
    "fixedInfoEncoding": "concatenation",
    "kdfMode": "feedback",
    "macMode": "HMAC-SHA2-512",
    "counterLocation": "after fixed data",
    "counterLen": 32,
    "ivLen": 512
}
},
{
    "tgId": 2,
    "testType": "AFT",
    "tests": [
        {
            "tcId": 6,
            "kdfMultiExpansionParameter": {
                "kdfType": "twoStep",
                "kdfMode": "feedback",
                "macMode": "HMAC-SHA2-512",
                "counterLocation": "after fixed data",
                "counterLen": 32,
                "salt":
"A0DCAD75DAF16543013DF2D838B0F043C70E66795DF15C725F2F8E73939C3744DBDC09EC9D28E5BB892BCC7C
                "iv":
"86F0F24DCF144927D946E05313039C5575261EB45E5C22EFA55C00CA4EDF5080C5E9549C0E4F9BF95203A459
                "z":
"F9CBD2C724EF2353D85C62D671040E5C77EA0899E87FFE07B935F039735938AA829110D49F0E2D1194869C7F
                "iterationParameters": [

```

```

        {
            "l": 512,
            "fixedInfo": "5C95F8AEF753345A12863D223BA83F8A"
        },
        {
            "l": 512,
            "fixedInfo": "E367A16FD5FA56D7337136C8927B7F19"
        },
        {
            "l": 512,
            "fixedInfo": "EF3A620269F1E88C20D55011F7380209"
        },
        {
            "l": 512,
            "fixedInfo": "B9A3B043CB6474CE53467FB89B24BDFD"
        }
    ]
}
},
{
    "tcId": 7,
    "kdfMultiExpansionParameter": {
        "kdfType": "twoStep",
        "kdfMode": "feedback",
        "macMode": "HMAC-SHA2-512",
        "counterLocation": "after fixed data",
        "counterLen": 32,
        "salt":
"829BB5915FA3FB20972CA6C22F1AED6BC40119EA50487BCA520CD60729964EFE218F40AEC3B2D78EF853EE06
        "iv":
"27DC1483266BD740E05CDA5DA3882EED1CF0A1E1E55A0BCCDE9F73E94B0FC5DF14C1AD320C08FF21585F5F26
        "z":
"281ED3C7FE60F4C2FE8B45B0D2EB3D1DD45A78B7C577A7F8FE03D8F59817EA5FC6DB41E5E45526A32A280FF5
        "iterationParameters": [
            {
                "l": 512,
                "fixedInfo": "4208FB9D1C803414A141C4E333B3D6A5"
            },
            {
                "l": 512,
                "fixedInfo": "3235F21101152080DC48AC200BC54297"
            }
        ]
    }
}
},
],

```

```

    "kdfMultiExpansionConfiguration": {
      "kdfType": "twoStep",
      "l": 512,
      "saltLen": 512,
      "saltMethod": "random",
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32,
      "ivLen": 512
    },
    "multiExpansion": true
  },
  {
    "tgId": 65,
    "testType": "VAL",
    "tests": [
      {
        "tcId": 321,
        "kdfParameter": {
          "kdfType": "twoStep",
          "salt":
"06614CC285E42B00C7625244594D27B5BB932A7C5E85EA02B116D191A7682F01DFCC4F207E912127AA11DA2D
      "t": "ACA02E08F6D7BAF6FC470435D5B88EE9",
      "z":
"6B91690E9D667043924421ABFA4056AA51A15793255B975BE5C44E37A40B892588640E90C8F5FBA0F8179C67
      "l": 512,
      "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
      "fixedInputEncoding": "concatenation",
      "iv":
"5D54A0FE3A35C6B76D056883EBF7E7FD4C71200E6C6B0A210EDF91B5297D7F89B5EF90ED66A7A0CCAE9B7301
      "kdfMode": "feedback",
      "macMode": "HMAC-SHA2-512",
      "counterLocation": "after fixed data",
      "counterLen": 32
        },
        "fixedInfoPartyU": {
          "partyId": "4865F140D0239D4967496C99D2C9E8AB"
        },
        "fixedInfoPartyV": {
          "partyId": "05F2D726E8C660784404DD69CD391EA3"
        },
        "dkm":
"0BD0E7F866581520EC749F2EAAE31C46184EAA3B1CA3E3DB706A94CCFF6093B1E4483BF80BF83068816F8CEE
      },
      {

```

```

        "tcId": 322,
        "kdfParameter": {
            "kdfType": "twoStep",
            "salt":
"05566C08DD5415A956FC2BA699547882DD22BA74CD02F4C69EAFD7EF9AD8D7AFF7218C337A88477ABD554CC2
            "t": "AB9864A12B7531DD07EB22CC4D2FE623",
            "z":
"8CDBA4BA67446579B5AB939A9BE589349B1479FA9A1E2E00E9C8D74AFA177FCA99E9A126427FB1A9A956BFFF
            "l": 512,
            "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
            "fixedInputEncoding": "concatenation",
            "iv":
"E1C558C98C2A62FD8ACD9C735CB894CC220C2FEBECE60C45C7CE1DD426BBFC9D5A96D182DAF181DA83508737
            "kdfMode": "feedback",
            "macMode": "HMAC-SHA2-512",
            "counterLocation": "after fixed data",
            "counterLen": 32
        },
        "fixedInfoPartyU": {
            "partyId": "30AEBBA0D104E03D6EA2022A73A78F47",
            "ephemeralData":
"4CF8A744EF46B6DC72FC5E43670E7EA8B2DBDBBC81C8A07090FAD03D96D6C913B6BD15DA8EBA5E76E73171A0
        },
        "fixedInfoPartyV": {
            "partyId": "9A2AB0A90060DB58C115C557977F2EC3"
        },
        "dkm":
"07423F264BB882C8CF3D2D08DDA2E0742AEAFBDF819EED35F8B5FFB4960AB5CCEDE479E2746390D539F65FCE
    }
},
"kdfConfiguration": {
    "kdfType": "twoStep",
    "l": 512,
    "saltLen": 512,
    "saltMethod": "random",
    "fixedInfoPattern": "uPartyInfo||vPartyInfo||l||t",
    "fixedInfoEncoding": "concatenation",
    "kdfMode": "feedback",
    "macMode": "HMAC-SHA2-512",
    "counterLocation": "after fixed data",
    "counterLen": 32,
    "ivLen": 512
}
},
{
    "tgId": 66,

```

```

    "testType": "VAL",
    "tests": [
      {
        "tcId": 326,
        "kdfMultiExpansionParameter": {
          "kdfType": "twoStep",
          "kdfMode": "feedback",
          "macMode": "HMAC-SHA2-512",
          "counterLocation": "after fixed data",
          "counterLen": 32,
          "salt":
"C814F6C4E49E535DFB59CE72EFEA3F8BD770043CA2E1AF6C9D459CD07D38EC129DB838C2AB6C184BC43C0654
          "iv":
"7A512E648ADAE756B97A90D62E3B56CBC3D5B4E12E139B94060F03D9BB1F189A75D961FE7C203D93432AC84E
          "z":
"D568E0018EBEB1139E35F8391C698977017B7B68147000D9C055A97D0823FB855904C7A1BDA996E8FE49C8A9
          "iterationParameters": [
            {
              "l": 512,
              "fixedInfo": "1AE9DA67B06472641A3BF37811051055"
            },
            {
              "l": 512,
              "fixedInfo": "4286E9173B9A3AF2CA0E7D3944E38D1F"
            },
            {
              "l": 512,
              "fixedInfo": "AD285511D6BA8ED8CA415BDDE2860ECE"
            }
          ]
        },
        "dkms": [

"F87E73FA73B856684BAC4F84E0811413EDBDAA126AB9157265BF46EA7EF75D41B9393546F96D44D67E2F6E6
"35F6C469BAB2A9CF929B72EBB8F985EA82863EBFA14CFEAEA7E669D3BC084897984A51EDFC406A1144807DF
"7B8494EC4A8BAA3518EFB0CBF0CD587EFDC56B87A2B23155031D2C579ED42F0C7743D2FC8CB67969FB85C31
        ]
      },
      {
        "tcId": 327,
        "kdfMultiExpansionParameter": {
          "kdfType": "twoStep",
          "kdfMode": "feedback",
          "macMode": "HMAC-SHA2-512",

```

```

        "counterLocation": "after fixed data",
        "counterLen": 32,
        "salt":
"7B02B397B12B1BC773B3487D73D8F4281A5A3795AC11A3229F668C78D2737494BB706BE69A3F6E65C64E6684
        "iv":
"AC9EAC5B613BB418B5E81141729CFDEA261ED2851211D9EBC2B95D363EFA95A06E07B5605C0AFFC04C3EFC92
        "z":
"6F2244DBD5E43AC838CD40529AE2CD4F46337E8A99B67EC35A3EC0E293E6942D8B4060DDC01300F0587042B0
        "iterationParameters": [
            {
                "l": 512,
                "fixedInfo": "47065AC25033394A84068D0B518013E8"
            },
            {
                "l": 512,
                "fixedInfo": "1D5F11557F50A7DB20231D5855D0B645"
            },
            {
                "l": 512,
                "fixedInfo": "871C59E84EADC342ADCF5CFA2F39316F"
            },
            {
                "l": 512,
                "fixedInfo": "382E863C637660EFCACCA9CFF88C7076"
            }
        ],
        "dkms": [
"4332AE745C762A6C4A2BF5A52D6FEE7B85123BFD447C40F6888E26C7EE4A480040ACA97BA55C856C1261DB9
"967CCA41FDDF0E51F68B9B1B43614EE302B2B1CD5ACEAE3A531F9A16C62F02ECBCAA83BA76FE86886010797
"1FC05E9FECBF82DE01C7A81F1DA360054C987A9EFF0C522F67D9B3609C0823A65DFF0E2812BBDF3868C45F4
"8EFB4A8EBFABB877D2CD76723599FF6DB4334A166E676E678AAC161FC548142DBE7D804ADB94824B1AAADDB
        ]
    }
]
}
]
}

```

Figure 6 — Vector Set JSON Example SP800-56Cr2

6. Test Vector Responses

After the ACVP client downloads and processes a vector set, it **MUST** send the response vectors back to the ACVP server. The following table describes the JSON object that represents a vector set response.

Table 14 — Vector Set Response Properties

JSON Property	Description	JSON Type
acvVersion	The version of the protocol	string
vsId	The vector set identifier	integer
testGroups	The test group data	array

The testGroups section is used to organize the ACVP client response in a similar manner to how it receives vectors. Several algorithms **SHALL** require the client to send back group level properties in their response. This structure helps accommodate that.

Table 15 — Test Group Response Properties

JSON Property	Description	JSON Type
tgId	The test group identifier	integer
tests	The test case data	array

The testCase section is used to organize the ACVP client response in a similar manner to how it receives vectors. Several algorithms **SHALL** require the client to send back group level properties in their response. This structure helps accommodate that.

The following table describes the JSON object that represents a test case response for a KDA-TwoStep SP800-56C.

Table 16 — Test Case Response Properties

JSON Property	Description	JSON Type
tcId	The test case identifier	integer
testPassed	Was the provided dkm valid? Only valid for the “VAL” test type.	boolean
dkm	The derived keying material. Provided by the IUT for “AFT” test type test cases.	hex

Here is an abbreviated example of the response

6.1. Example Test Vectors Response JSON

```

{
  "vsId": 0,
  "algorithm": "KDA",
  "mode": "TwoStep",
  "revision": "Sp800-56Cr1",
  "isSample": true,
  "testGroups": [
    {
      "tgId": 1,
      "tests": [
        {
          "tcId": 1,
          "dkm":
"F30FA60BE892E9506BB1D7F0F69130A1F9142083E7EA64076D1ECDF5103571858FB742F94D96015EE0C747F3
        },
        {
          "tcId": 2,
          "dkm":
"F6BF05D035D95C05B694DADE4DA41D05DC83E6CA100BFB1F13E24639C7DE5AE6B1ED0FECC1860F2246FF6262
        },
        {
          "tcId": 3,
          "dkm":
"D7DC1B2E454D66529F82999006D67B2595D3862C81DB318B75DC005C888F294A652FBC35336CC5BD4510CE9C
        },
        {
          "tcId": 4,
          "dkm":
"677D6B89F0E747EBA608180A711D5964EAAC73690F72365E871D98AD0440DC6645312773651981B0A9C483D0
        },
        {
          "tcId": 5,
          "dkm":
"875FADC0885DB1B0F0D2C38A27CC1169C4F5F4474F539A302169A99EA00689C587CE6B9873CFE681DAC6B02B
        }
      ]
    },
    {
      "tgId": 33,
      "tests": [
        {
          "tcId": 161,
          "testPassed": true
        },
        {
          "tcId": 162,

```

```
    "testPassed": false
  },
  {
    "tcId": 163,
    "testPassed": true
  },
  {
    "tcId": 164,
    "testPassed": true
  },
  {
    "tcId": 165,
    "testPassed": true
  }
]
}
```

Figure 7 — Example Response JSON

7. Security Considerations

There are no additional security considerations outside of those outlined in the ACVP document.

Appendix A — Terminology

For the purposes of this document, the following terms and definitions apply.

A.1.

Prompt

JSON sent from the server to the client describing the tests the client performs

Registration

The initial request from the client to the server describing the capabilities of one or several algorithm, mode and revision combinations

Response

JSON sent from the client to the server in response to the prompt

Test Case

An individual unit of work within a prompt or response

Test Group

A collection of test cases that share similar properties within a prompt or response

Test Vector Set

A collection of test groups under a specific algorithm, mode, and revision

Validation

JSON sent from the server to the client that specifies the correctness of the response

Appendix B — Abbreviations and Acronyms

ACVP	Automated Crypto Validation Protocol
JSON	Javascript Object Notation

Appendix C — Revision History**Table C-1**

Version	Release Date	Updates
1	2020-12-11	Initial Release

Appendix D — References

- S. Bradner (March 1997) *Key words for use in RFCs to Indicate Requirement Levels* (Internet Engineering Task Force), BCP 14, March 1997. RFC 2119. DOI 10.17487/RFC2119. <https://www.rfc-editor.org/info/rfc2119>.
- P. Hoffman (December 2016) *The “xml2rfc” Version 3 Vocabulary* (Internet Engineering Task Force), RFC 7991, December 2016. RFC 7991. DOI 10.17487/RFC7991. <https://www.rfc-editor.org/info/rfc7991>.
- B. Leiba (May 2017) *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words* (Internet Engineering Task Force), BCP 14, May 2017. RFC 8174. DOI 10.17487/RFC8174. <https://www.rfc-editor.org/info/rfc8174>.
- Lily Chen (October 2009) *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)* (Gaithersburg, MD), October 2009. SP 800-108. <https://doi.org/10.6028/NIST.SP.800-108>.
- Elaine B. Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis (April 2018) *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography* (Gaithersburg, MD), April 2018. SP 800-56A Rev. 3. <https://doi.org/10.6028/NIST.SP.800-56Ar3>.
- Elaine B. Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, Scott Simon (March 2019) *Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography* (Gaithersburg, MD), March 2019. SP 800-56B Rev. 2. <https://doi.org/10.6028/NIST.SP.800-56Br2>.
- Elaine B. Barker, Lily Chen, Richard Davis (April 2018) *Recommendation for Key-Derivation Methods in Key-Establishment Schemes* (Gaithersburg, MD), April 2018. SP 800-56C Rev. 1. <https://doi.org/10.6028/NIST.SP.800-56Cr1>.
- Elaine B. Barker, Lily Chen, Richard Davis (August 2020) *Recommendation for Key-Derivation Methods in Key-Establishment Schemes* (Gaithersburg, MD), August 2020. SP 800-56C Rev. 2. <https://doi.org/10.6028/NIST.SP.800-56Cr2>.
- Fussell B, Vassilev A, Booth H, Celi C, Hammett R (July 01, 2019) *Automatic Cryptographic Validation Protocol* (National Institute of Standards and Technology, Gaithersburg, MD), July 01, 2019.