MIS Table Descriptions – V2.8.4 (5/20/2024)

This supplement to SP800-140B contains detailed descriptions of the tables of information required. The columns represent information that will be entered into Web Cryptik or included in the json file. “[O]” designates that information for the column is optional and may be left blank. The json property is not optional but would be an empty string.

These tables correspond with the arrays within the web-cryptik-schema. The table numbers correspond with Web Cryptik, not necessarily the table numbers within the generated Security Policy.

If the module doesn’t have relevant information for a required column, enter “N/A”. Optional columns may be left blank.

The number in parentheses corresponds to the section in the SP in which this table will be displayed.

Note that this is the information that will be collected through Web Cryptik and saved in the json file. The way this information is displayed in the Security Policy can be different (different columns/sorting/etc.) and is determined in the SP Building process.

The name in brackets following the table name corresponds to the json array of objects where that table is stored.

[U] – The column contains unique values
[O] – The column is optional and may be left empty
[M] – The column uses multiple, separate entries as opposed to separating entries in one text field

If the value for a column is selected from a list of values, those values are displayed in the column below

If the value for a column is selected from the values from another table, the table and corresponding column are listed with angle-brackets, for example “>CAVP_OEs>Name”.

CAVP Certs Selection Tables

These tables are built based on the information from the CAVP certificate system and are selected from existing data, as opposed to being entered directly.

CAVP Certs – [cavpCertSet.cavpCertList]

Name	Implementation Name	Version	Type	Vendor

Notes

- This information all comes from the CAVP system
- Name – CAVP Cert Name
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
    - validationId

CAVP OEs – [cavpCertSet.cavpOeList]

Name	OE ID

Notes

- Unique ID fields saved in the MIS but not displayed in Web Cryptik
- OE ID is a unique, internal number assigned to the CAVP Tested Operational Environments

CAVP OE Algorithms – [cavpCertSet.cavpOeAlgoList]

CAVP Cert Name	OE ID	Algorithm	Capabilities

Notes

- There will be one row per unique Algorithm/OE so for one algorithm, there will be a row for each OE tested and implemented by the module
- Capabilities will only be entered if not all of the tested capabilities are implemented by the module
- Implementation Name from the CAVP Cert is saved in the MIS but not displayed in Web Cryptik
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
    - validationOeAlgorithmId
    - canonicalAlgorithmId
    - validationId

CAVP Algorithms – [cavpCertSet.cavpImplAlgoList]

Algorithm	Implementation Name	CAVP Cert Name	Category [O]

Notes

- This is generated as a result of the OeAlgos selected into the above table. The entries in this table will be used when the other tables select an algorithm
- There will be one row per unique Algorithm/Implementation. The implementation will be taken from each CAVP cert. The canonicalAlgorithmId and the validationId are used to do the grouping.
- The corresponding algoDisplayName and implName are show in the table.
- If the same implementation is tested in more than one CAVP Cert, then multiple rows would be included for one implementation. In that case, all but one of the multiple tests would need to be removed from the OeAlgo selection process and therefore would be removed from the table.
- Category – If entered, the list of algorithms in the SP will be grouped by Category and split into separate tables
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
    - algoIdValId – concatenation of the canonicalAlgorithmId and the validationId and creates a unique ID for each row
    - canonicalAlgorithmId
    - validationId
- the Properties List field in the MIS is always saved as an empty array and filled in to create the algorithm table in the SP

CAVP ITAR Algorithms – [cavpCertSet.cavpItarAlgoList]

CAVP Cert Name	Algorithm	Capabilities [O]	Category [O]

Notes

- Table is only present if module is ITAR
- Cert numbers entered by hand instead of selected as above
- Algorithm column selected from list of all tested algorithm names
- All OEs and capabilities are assumed to be implemented in the module unless separately entered in the Capabilities field
- Category works in the same manner as above
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
    - canonicalAlgorithmId

Table 1: Tested Module Identification – Hardware (2.3) – [testedHwList]

Model/Part Number(s) [U]	Hardware Version(s)	Firmware Version(s)	Processor(s)	Non-Security Relevant Distinguishing Features [O]

Notes

- Number of rows correspond to module count
- Processor(s) – Needs to match processor information identified in OEs for corresponding CAVP testing
- Examples of distinguishing features may be ports and interfaces, memory storage devices and sizes, field replaceable and stationary accessories (power supplies, fans), etc.

Display if Module Type is Hardware

Table 2: Tested Module Identification – Software/Firmware/Hybrid (Executable Code Sets) (2.3) – [testedSwFwHyList]

Package/File Names [U]	Software/ Firmware Version	Non-Security Relevant Distinguishing Features [O]	Integrity Test Implemented

Notes

- Number of rows corresponds to module count
- Version column may contain multiple entries

Display if Module Type is Software, Firmware, or Hybrid

Table 2a: Tested Module Identification – Hybrid Disjoint Hardware (2.3) – [testedHyHwList] (new)

Model/Part Number(s)	Hardware Version(s)	Firmware Version(s) [O]	Processor(s) [O]	Non-Security Relevant Distinguishing Features [O]

Notes

- Number of rows corresponds to module count
- Version column may contain multiple entries

Display if Module Type is Hybrid

Table 3: Tested Operational Environments – Software/Firmware/Hybrid (2.3) – [opEnvSwFwHyTestedList]

Operating System (Guest OS if Hypervisor)	Hardware Platform	Processor(s)	PAA/PAI	Hypervisor and Host OS [O]	Version(s) [M]
			"Yes" "No"		>testedSwFwHyList >swFwVersion

Notes

- One row for each Tested OE
- Processor(s) – Should match processor information identified in OEs for corresponding CAVP testing
- Version: Which versions were tested on this OE. Multi-select

Display if Module Type is Software, Firmware, or Hybrid

Table 4: Vendor Affirmed Operational Environments – Software/Firmware/Hybrid (2.3) – [opEnvSwFwHyVAList]

Operating System	Hardware Platform

Notes

- One row for each Vendor Affirmed OE

Display if Module Type is Software, Firmware, or Hybrid, otherwise hide

Table 5: Modes of Operation (2.4) - [modeOfOpList]

Name [U]	Description	Type	Status Indicator [O]
		"Approved" "Non-Approved"	

Notes

- Name – Unique name used as identifier
- Status Indicator – Description of the source of the status indicator, for example from service or global indicator.

Approved Algorithms (2.5) – [generated from cavpCertSet.cavpAlgoList]

Algorithm Name (Implementation)	CAVP Cert Name	Algorithm Properties	Reference

Notes

- This table is generated from the CAVP Algorithms table created during the CAVP selection process above
- Algorithms can be entered using a Category field. If this is present, the algorithms will be grouped by Category and separate tables entered with the Category as a table header

- Properties will be filled in from a combination of the selected properties and the tested properties
- Reference will be filled in based on the algorithm

Table 6: Vendor Affirmed Algorithms (2.5) – [vendorAffirmedAlgoList]

Algorithm Name	Algorithm Properties [O] [M]	Implementation	Reference
	Name: Value Name: Value	>cavpCertSet >cavpCertList >implName or Other	

Notes

- Algorithm Name – Entered for now but in the future will be selected from a list of available values
- Algorithm Properties – Name/Value pairs
    - Over time, specific capabilities will be identified for the possible entries
- Implementation – Selected from list of Implementations represented by CAVP Tests. If there isn't an appropriate match, select "Other" and enter a brief description of the Implementation or N/A
- Reference – Describe and provide reference to justification, a publication or IG reference.
- Note the separately provided guidance on the use of CKG

Table 7: Non-Approved, Allowed Algorithms (2.5) – [nonApprovedAllowedAlgoList]

Algorithm Name	Algorithm Properties [O] [M]	Implementation	Reference
	Name: Value Name: Value	>cavpCertSet >cavpCertList >implName or Other	

Notes

- Algorithm – Entered for now but in the future will be selected from a list of available values
- Algorithm Properties – Name/Value pairs
    - Over time, specific capabilities will be identified for the possible entries
- Implementation – Selected from list of Implementations represented by CAVP Tests. If there isn't an appropriate match, select "Other" and enter a brief description of the Implementation
- Reference – describe and provide reference to justification, a pub or IG reference, for example

Table 8: Non-Approved, Allowed Algorithms with No Security Claimed (2.5) – [nonApprovedAllowedAlgoNSCList]

Algorithm	Caveat	Use/Function

Notes

- None

Table 9: Non-Approved, Not Allowed Algorithms (2.5) – [nonApprovedNotAllowedAlgoList]

Algorithm	Use/Function

Notes

- None

Table 10: Security Function Implementations (SFI) (2.6) – [secFunImplList]

Name [U]	Type	Description	SF Properties [O] [M]	Algorithms [M]	Algorithm Properties [O] [M]
	SFI Type List		Name: Value Name: Value	>CAVP_Algos >Name (Impl)	Name: Value Name: Value
				Algo 2	Name: Value Name: Value
				Algo 3	Name: Value

Notes

- Column Information
  - Name – a unique name that relates to the Security Function. It can be KTS1, or KTS xxx
  - Type – one or more values from the defined set of Security Functions. This list can be found on the SP800-140B webpage: https://csrc.nist.gov/projects/cmvp/sp800-140b
  - Description – how this is used
  - SF Properties – Name/Value pairs. If there are specific capabilities or characteristics associated with this SF implementation. This could include a reference to a specific Publication Section, IG, etc. This is where appropriate bit strength caveats should be included for KAS and KTS. There is a separate column for Algorithm Properties so this column is used for properties specifically associated with the higher-level security function.
  - Algorithms – the set of tested algorithms that comprise the implementation to include prerequisites.
  - Algorithm Properties – If a subset of the available capabilities are used, specify.
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
  - sfId
- What is meant by Implementations of Security Functions
  - A module can (and often does) have more than one implementation for a given Security Function type
    - A KTS that uses an authenticated encryption mode vs. separate encryption and authentication would both be KTS but would have two implementation entries
    - A SigVer could be used for role/identity authentication and also for an integrity test
    - Block Cipher could include modes for storage (XTS) or as part of a KTS

        - The same algorithm could be used with different key sizes to support different sizes
    - For many modules, there would likely be one SFI for a SF type.
- Why these wouldn't just map directly to Services
    - At times, these could map directly to services, particularly for modules like software libraries.
    - Documenting in this manner will clarify which algorithms are actual services provided and which are supporting or prerequisite
    - When the same category SF algorithms are used for different functions and therefore different services, there should be separate SFIs. Many modules have multiple DigSigVer implementations. For example, one for authentication during an SSH connection and one for the module startup integrity test. These should be separately defined as implementations and then mapped to different services.
    - Requiring the Services to map directly to the Security Functions seems to overreach into the vendor's design of their module. The Services and corresponding level of granularity should be left to the vendor to determine.
- There should only be entries for top-level functions. For example, if SHA2-256 is only used for Hash DRBG, then it shouldn't be included as a separate Secure Hash entry. And, if the DRBG is only a supporting function (for example, just a prerequisite to Symmetric Key Generation), then DRBG shouldn't be a separate entry in this table. The Services table will include the Security Function Implementations, so often that will likely determine what is a top-level entry.
- All the supporting and prerequisite algorithms for that implementation would be included in the Algorithms column.
- Every tested and allowed algorithm should be included somewhere in this table.
- Every SFI should be included in the Services table.

Table 11: Entropy Certificates (2.8) – [esvCertList]

Vendor Name	Certificate Number

Notes

- Selected from existing ESV certificates
- Unique ID fields saved in the MIS but not displayed in Web Cryptik
    - certId

Table 11a: Entropy Certificates (ITAR) (2.8) – [esvItarCertList]

Certificate Number

Notes

- Only present if module is ITAR
- Entered by hand verified internally to CMVP

Table 12: Entropy Sources (2.8) – [entropySourceList]

Name [U]	Type	Operating Environment(s)	Sample Size	Entropy per sample	Conditioning Components (CAVP number if vetted)
	"Physical" "Non-Physical"				

Notes

- In the future, this information will be gathered from the ESV system

Table 13: Ports and Interfaces (3.1) – [portInterfaceList]

Physical Port	Logical Interface(s) [M]	Data that passes over the port/interface
	"Data Input" "Data Output" "Control Input" "Control Output" "Status Output" "Power" "None"	

Notes

- Logical Interface(s) – multi-select from dropdown

Table 14: Authentication Methods (4.1) – [authMethodList]

Name [U]	Description	Mechanism	Strength Each Attempt	Strength Per Minute [O]
		>CAVP_Algos >Name (Impl) or >SFI>Name or Other (then typed in)		

Notes

- Mechanism can be module algorithm, SFI, or alternative

Display if Security Level for Section 4 is 2, 3, or 4

Table 15: Roles (4.2) – [roleList]

Name [U]	Type	Operator Type	Authentication Methods [M]

	“Role” “Identify” “Multi-Factor Identity”	CO User Maintenance etc.	>Authentication_Methods>Name or Blank/Disabled if Security Level 1

Notes

- Type – which type of authentication is applicable for the role
- Operator Type – the role within ISO 19790 7.4.2 that this role corresponds to. It may be Crypto Officer, User, Maintenance, or other as defined by the module.

Table 16: Approved Services (4.3) – [approvedServiceList]

Name [U]	Description	Indicator	Inputs	Outputs	Security Function Implementations [M]	Roles [M]	Roles SSP Access [M] [O]
					>SFI>Name or “None”	>Roles>Name or “Unauthenticated”	>SSPs>Name

Notes

- Indicator – Since this is a required field, if the service doesn’t have an indicator, choose “Other” and enter “N/A”
- Roles SSP Access
    - For each role entry, this column has entries for each SSP accessed by that role using that service with the appropriate access indicators
        - Generate: The module generates or derives the SSP.
        - Read: The SSP is read from the module (e.g. the SSP is output).
        - Write: The SSP is updated, imported, or written to the module.
        - Execute: The module uses the SSP in performing a cryptographic operation.
        - Zeroise: The module zeroises the SSP.

Example of the “Roles” and “Role SSP Access” columns:

Name	**Roles**	**Roles SSP Access**
AES encryption	CO	AES cryptographic keys: Execute
	User	AES cryptographic keys: Execute
Configure secret information	CO	Authentication ID: Write AES cryptographic keys: Write DRBG internal state: Execute, Write
Output secret information	CO	Key seed: Read CO authentication Information: Execute
	User	Key seed: Write CO authentication Information: Write

Table 17: Non-Approved Services (4.4) – [nonApprovedServiceList]

Name [U]	Description	Algorithms Accessed [M]	Role
		>Non_Approved_Algos >Name or “None”	

Notes

- None

Table 18: Mechanisms and Actions Required (7.1) – [phSecMechanismList]

Physical Security Mechanism	Recommended Frequency of Inspection/Test	Inspection/Test Guidance Details

Notes

- None

Display if Module Type is Hardware

Table 19: EFP/EFT Information (7.5) – [efpEftInfoList]

	Temperature or voltage measurement	Specify EFP or EFT	Specify if this condition results in a shutdown or zeroisation
Low Temperature			
High Temperature			
Low Voltage			
High Voltage			

Notes

- EFP or EFT is required for modules with physical Security Level 3.
- EFP is required for modules with physical Security Level 4.

Display if Module Type is Hardware and Security Level for Section 7 is 3 or 4.

Table 20: Hardness Testing Temperature Ranges (7.6) – [hardnessTestTempList]

	Hardness tested temperature measurement
Low Temperature	

High Temperature	

Notes

- The module is hardness tested at the lowest and highest temperatures within the module's intended temperature range of operation

Display if Module Type is Hardware and Security Level for Section 7 is 3 or 4

Table 21: Storage Areas (9.1) – [storageAreaList]

Name [U]	Description	Persistence Type
		"Dynamic" "Static"

Notes

- Name should correspond to a specific item in the block diagram

Table 22: SSP Input-Output Methods (9.2) – [sspInputOutputList]

Name [U]	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm [O]
	>StorageArea >Name Or Entered	>StorageArea >Name Or Entered	"Encrypted" "Plaintext"	"Manual" "Automated" "Wireless" "N/A"	"Direct" "Electronic" "N/A"	>SFI>Name or >CAVP_Algos >Name (Impl)

Notes

- Name – Unique, descriptive name
- From/To
    - Clearly indicate one as inside and the other as outside the cryptographic boundary
    - Include any input/output devices
    - For internal references, provide a component/structure that is clearly identified in the block diagram and/or a storage area from the list
- Distribution Type – Reference IG 9.5.A
- Entry Type – Reference IG 9.5.A
- SFI or Algorithm – If one of these are used in the input/output action

Table 23: SSP Zeroization Methods (9.3) – [sspZeroizationList]

Method [U]	Description	Rationale	Operator Initiation Capability

Notes

- These are the options for the Zeroization column in the SSPs table
- Rationale – from ISO-19790: "Specification of the zeroisation method(s) employed by a module and the rationale as to how the method(s)" prevent(s) the retrieval and reuse of the zeroised values.

Table 24: SSPs (9.4) – [sspList]

Name [U]	Description	Size	Strength	Type	Generated By [O] [M]	Established By [O] [M]
					>SFI>Name or >CAVP_Algos >Name (Impl) or Other (enter text)	>SFI>Name or >CAVP_Algos >Name (Impl)

Used By [O] [M]	Inputs/ Outputs [O] [M]	Storage [M] [O]	Temporary Storage Duration [O]	Zeroization Method(s) [M]	Category	Related SSPs [O] [M]
>SFI>Name or >CAVP_Algos >Name (Impl)	>SSP_IO >Name	>Storage_Areas >Name and "Plaintext" "Encrypted" "Obfuscated" and >CAVP_Algos >Name (Impl) [O]		>Zeroization >Name or "N/A"	CSP PSP Neither	>SSPs>Name and Relationship

Notes

- Size – in bits
- Strength – in bits
- Type
    - Symmetric Key, Public/Private, Authentication, Signature Type, etc.
    - In the future there will be a specific list of options
- For Size and Strength when there are different entries (particularly for a software library), create one row for each similar type/use of SSP and enter in the one text field the multiple size/strength values and/or range.

- Generated By - Indicate if the generation is internal (select option) or external (select "Other" and enter description)
- Storage
    - Indicate if the SSP is stored as Plaintext, Encrypted, or Obfuscated
        - If encrypted, what algorithm/mechanism is used, selected from tested/approved algorithms
- Temporary Storage Duration – If the SSP is stored temporarily, enter the length of time it is stored. If it is not stored temporarily, leave blank.
- Zeroization
    - Selected from the zeroization table
    - Multiple entries if applicable
- Related SSPs
    - Indicate relationship to current SSP – "Derived From", "Wrapped By", "Wraps", "Paired With", etc.
    - Select "None" if there are no related SSPs.

Table 25: Pre-Operational Self-Tests (10.1) – [preOpSelfTestList]

Algorithm or Test	Test Properties	Test Method	Type	Indicator	Details
CAVP_Algos> >Name (Impl) or Other (Typed in)			"SW/FW Integrity" "Bypass" "Critical Function"		

Notes

- Algorithm from set of tested/allowed algorithms/implementations
- Test Properties – the key length, signature, curve, etc. used for the test
- Test Method – KAT, PCT, etc.
- Indicator – the indicator that the test has been run
- Details – Encrypt, Decrypt, Sign, Verify, etc.
- Any relevant information related to the different implementations should be included in the "Notes" section following the table.

Table 26: Conditional Self-Tests (10.2) – [condSelfTestList]

Algorithm or Test	Test Properties	Test Method	Type	Indicator	Details	Condition
CAVP_Algos> >Name (Impl) or Other (Typed In)			"CAST" "PCT" "SW/FW Load" "Manual Entry"			

			“Bypass” “Critical Function”			

Coverage [M] [O]	Coverage Notes [O]
CAVP_Algos> >Name (Impl)	

Notes

- One row per test, not groups of tests
  - AES Encrypt and Decrypt would be two tests since both have to be verified
  - Not additional rows per properties of tests. For example, if tests are done with multiple key sizes then those can all be included in one row
  - Also one row per implementation, if a particular algorithm has more than one implementation within the module
- Algorithm from set of tested/allowed algorithms/implementations
- Test Properties – the key length, signature, etc. used for the test
- Test Method – KAT, PCT, etc.
- Type – CAST, PCT, Software/Firmware Load, Manual Entry, Bypass, Critical Functions
- Indicator – the indicator that the test has been run
- Details – Encrypt, Decrypt, Sign, Verify, etc.
- Condition – under what condition is the test run
- Coverage – for CAST tests, this identifies which of the implemented algorithms from the Approved Algorithms table this test covers. Multiple entries may be selected. All of the algorithms from the Approved Algorithms list should be included in the entire set from the table
- Coverage Notes - a brief statement or reference providing the basis for coverage, like an IG number and section
- Any relevant information related to the different implementations should be included in the “Notes” section following the table.

The following two items are included with the tables from 10.1 and 10.2

Period	Periodic Method

Notes

- Period – how often the periodic test is run, typically “On Demand” for levels 1 and 2
- Periodic Method – how the periodic test is run, such as manually, programmatically, etc.

Table 28: Error States (10.4) – [errorStateList]

State Name [U]	Description	Conditions [M]	Recovery Method	Indicator

Notes

- None