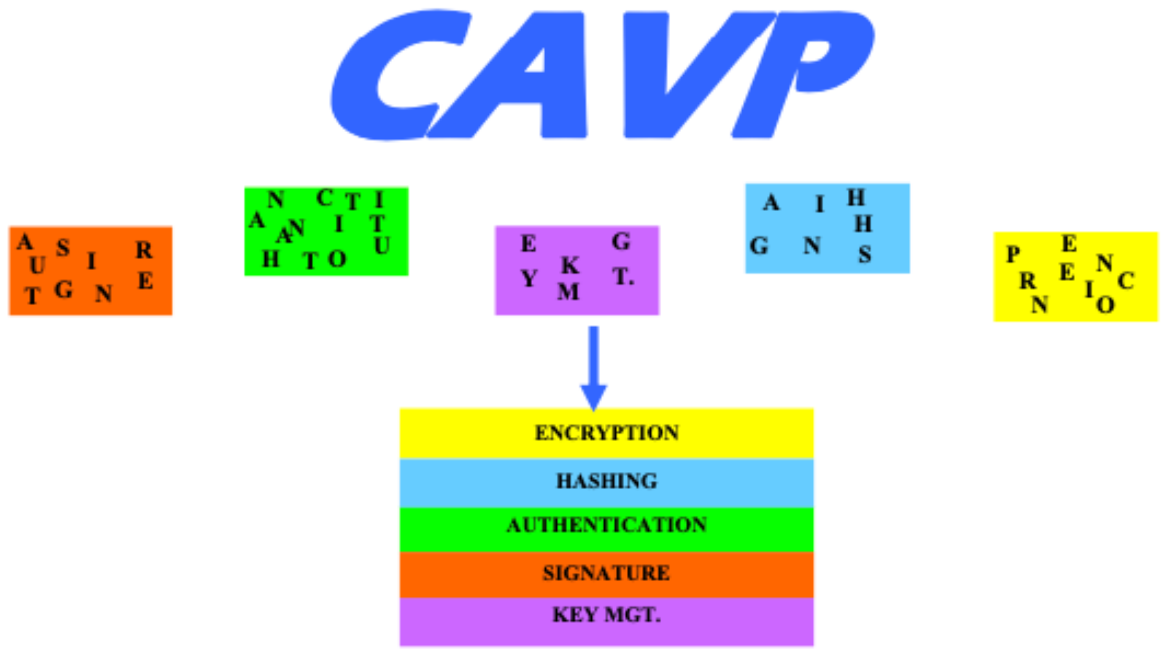
Frequently Asked Questions For the Cryptographic Algorithm Validation Program

National Institute of Standards and Technology



Last Update: February 24, 2022

Previous Update: May 6, 2016

Initial Release: December 3, 2009

Change Log:

2/24/22

- **New FAQ written; old questions removed**

1. INTRODUCTION 4
2. ACRONYMS 4
3. GENERAL ALGORITHM FAQS (CAN BE APPLIED TO ALL ALGORITHMS) 4
4. AES FAQ 12
5. TRIPLE-DES FAQ 13
6. DSA FAQ 14
7. RSA FAQ 15
8. HMAC FAQ 15
9. ECDSA FAQ 15
10. CMAC FAQ 16
11. GCM FAQ 17
12. DRBG FAQ 17

1. Introduction

Below is a compilation of questions received from the Cryptographic Security Testing (CST) laboratories relating to the validation of cryptographic algorithm implementations.

This is intended for use by the CST laboratories when validating cryptographic algorithms submitted by vendors. Vendors may find the information useful when developing implementations and working with the CST laboratories for cryptographic algorithm implementation validation.

2. Acronyms

Selected acronyms and abbreviations used in this document are defined below.

ACVP	Automated Cryptographic Validation Protocol
ACVTS	Automated Cryptographic Validation Test System
CAVP	Cryptographic Algorithm Validation Program
CST	Cryptographic Security Testing
IG	Implementation Guidance
OE	Operating Environment

3. General Algorithm FAQs (Can be applied to all algorithms)

GEN.0 Where can I find the list of all supported algorithms?

The list of all supported algorithms can be found at https://pages.nist.gov/ACVP/#supported. The ACVTS production environment `/algorithms` endpoint can also be queried to provide a list of all supported algorithms (see https://pages.nist.gov/ACVP/draft-fussell-acvp-spec.html#name-algorithms).

GEN.1 Where can the documentation for the ACVP, ACVTS, and the supported algorithms be found?

The documentation for ACVP can be found at https://pages.nist.gov/ACVP/draft-fussell-acvp-spec.html. ACVTS documentation is at https://github.com/usnistgov/ACVP/wiki and https://github.com/usnistgov/ACVP-Server/wiki. The documentation for supported algorithms is at https://pages.nist.gov/ACVP/#supported.

GEN.2 Is it acceptable if an implementation of an algorithm is presented in such a manner that the end user using the implementation must make calls to several functions in order to perform a major function of the algorithm (for example, Signature Generation)?

Generally, no. NIST expects that all the parts of an implementation of an algorithm will be contained within one executable (or its equivalent in firmware or hardware) and that one call to the algorithm implementation will determine which of the underlying functions are executed, how these underlying functions are executed, and in what order these functions are executed. For example, in a PKCS1.5 implementation, as the PKCS#1 v2.1 document states, we would expect that a call to RSASSA-PKCS1-V1_5_SIGN would call EMSA_PKCS1_V1_5_ENCODE and RSASP1. The ACVTS testing has been designed to assure that the functionality of the underlying functions within an algorithm implementation is operating correctly. If all the parts are supplied to an end user with the ability to put them together any way possible, there is no guarantee that they will be called in the order specified by the standard for that algorithm. Therefore, we cannot validate this implementation as a completed implementation. Providing instructions in the Security Policy as to the fixed order of operations of component functions is also not acceptable.

However, there are cases where two or more distinct entities in a system cooperate to execute an algorithm, such as the case of a smart card and a smart card reader. In this case the functions that comprise the digital signature algorithm are divided between the two parts of the system, card and reader, and the order of operations is fixed so that there is no way for the component functions of the algorithm to be called out of order.

The testing of individual algorithm components was introduced in 2011. Several situations have led to this new type of testing. For example, PIV cards have limited processing space and therefore some parts of the algorithm are performed off card.

In the case of FIPS 180 and FIPS 202 that define the SHA family of hash functions, a common implementation interface uses Init, (at least one) Update, and Final as three successive calls. This is an allowed exception to this rule. The same exception applies to FIPS 198 that defines HMAC, and SP 800-185 that defines eXtensible Output Functions (XOFs).

Where the standard identifies individual operations that comprise the cryptographic operation, such as SP 800-90A for DRBG, follow the requirements of the standard.

GEN.5 Are there any prerequisites to having some algorithms validated?

Yes. Some cryptographic algorithms make use of other cryptographic algorithms. For example, HMAC makes use of SHA. In such cases, the underlying algorithms must be tested in addition to testing the algorithm itself. The reason for this is that the tests for the original algorithm test the algorithm specifications, components, features, and/or functionality of that algorithm while relying on the correctness of the underlying algorithms. They focus on testing the original algorithm and do not adequately test the underlying algorithms. For example, the validation tests for HMAC thoroughly test the HMAC processing, but they do not thoroughly test the underlying SHA implementation. Therefore, SHA validation tests need to be performed as a prerequisite to the HMAC validation testing to provide this assurance for the underlying SHA implementation.

As of the date of this FAQ's writing, CAVP plans to create a page accessible from https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program to map ACVTS-supported algorithms to their prerequisites.

GEN.6 An algorithm implementation has restrictions on its use because of the application that contains it. For example, an AES-CCM implementation adheres to an additional standard that limits AAD fields. Can I validate the algorithm implementation?

There are two major ways for an algorithm implementation to have restrictions on its use. The first is through limiting sizes of data or input for specific application. A common case here is only allowing complete bytes of data to be hashed, rather than any number of bits that FIPS 180-4 supports. This is allowed and covered by many algorithm validation tests. If a specific size or combination of parameters is not available for an algorithm implementation, please contact the CAVP. The second way is by requiring specific fields to have additional structure not specified in the NIST standard. This may be the case when adhering to an additional standard beyond just the NIST standard such as for a protocol. In this case, the CAVP attempts to support many common protocols for testing beyond the base algorithm implementation. It is encouraged to ensure the base algorithm implementation is available for testing. If it is not available, other means can be taken to ensure the algorithm implementation is tested to receive a validation. Contact the CAVP for these cases.

GEN.7 Guidance on the relationship between the operating environment for cryptographic algorithm implementation validations and the operating environment for cryptographic modules.

FIPS 140-3 IG 2.3.A, Binding of Cryptographic Algorithm Validation Certificates, identifies the configuration control and operational environment requirements for the cryptographic algorithm implementation(s) embedded within a cryptographic module when the latter is undergoing testing for compliance to FIPS 140-3. This IG includes the following statement:

For a validated cryptographic algorithm implementation to be embedded within a software, firmware or hardware cryptographic module that undergoes testing for compliance to FIPS 140-3, the following requirements must be met:

1. the implementation of the validated cryptographic algorithm has not been modified upon integration into the cryptographic module undergoing testing; and

2. the operational environment under which the validated cryptographic algorithm implementation was tested by the CAVP must be identical to the operational environment that the cryptographic module is being tested under by the CST laboratory.

Please refer to FIPS 140-3 IG 2.3.A for more information. The corresponding reference for FIPS 140-2 is IG 1.4.

GEN.8 What should be done if an algorithm implementation is housed on two different version numbers of a chip?

Generally, any two implementations of an algorithm that have different version numbers must be validated separately regardless of the physical differences. Two test sessions must be generated by ACVTS to test the two operating environments. However, the vendor is not required to choose a packaged IC as the physical boundary. The CAVP allows validation of a die, so if a die is validated for an algorithm or algorithms, then any packaged ICs that contain the die do not need to be validated again for the same algorithms.

GEN.9 Suppose an algorithm implementation has been validated. What happens when a change is made inside the implementation's boundary? It is claimed that no cryptographic functions were changed. Is the algorithm validation still valid?

No. Any change inside the defined boundary of the implementation creates a new implementation, which must be validated. It does not matter what the change is.

For a software algorithm, CAVP validates the binary executable or dynamic library that contains the executable code of the algorithm implementation. The implementation's boundary is effectively the entire binary file that contains the algorithm implementation. So, if that binary file is different, the algorithms in it must be tested again. It does not matter what the change is. Even if the algorithm source code has not changed, the algorithm implementation needs to be retested.

GEN.10 If a vendor claims that their implementation runs on multiple operating systems, how should this be validated?

CAVP validations list the operating system and processor, known as the operating environment (OE), on which the testing was conducted. Only the OEs that are tested appear on the validation certificate and can be referenced in module validation. One set of tests is required for each OE to appear on the validation certificate. The OE(s) used for CMVP module validation must match the OE(s) used for algorithm validation.

A vendor is allowed to test on more OEs than required if it so chooses.

GEN.11 A vendor has indicated that the version number of a previously validated algorithm implementation has changed. They indicate that nothing within the algorithm implementation boundary has been changed. What should be done?

The laboratory would need to verify that none of the code within the algorithm boundary has been modified. The appropriate request to NIST ACVTS would be made by the laboratory requesting an update to the implementation name.

GEN.12 What information is required in the Operational Environment?

When submitting the algorithm test results to the CAVP, the operational environment **on which the testing was performed** must be specified. Reference FIPS 140-3 IG 2.3.A; the corresponding reference for FIPS 140-2 is IG 1.4.

For Software implementations, the following information must be listed:

1. Processor – This field shall identify the vendor, processor family, and microarchitecture. Examples that satisfy the requirement for the processor field are Intel Xeon W (Rocket Lake), ARM Cortex-R (ARMv8-R), and AMD Ryzen 5 (Zen 3).

   No further specificity is required **unless** the vendor or the lab knows that the software implementation executes differently on different processors within the same family. In this case, the listing must be more specific. **AES.2** describes such a case. A vendor may also choose to be more specific than required, e.g., to gain a marketing benefit or to meet the requirements of a validation authority.

2. Operating system – This field shall identify the vendor and operating system family, or major version number where more appropriate. Examples that satisfy the requirement for the OS field are Microsoft Windows 11, Apple macOS Monterey, Red Hat Enterprise Linux 8, and Wind River VxWorks 7.

   No further specificity is required **unless** the vendor or lab knows that the software implementation executes differently on different OSes within the same OS family or major version number. In this case, the listing must be more specific. A vendor may also choose to be more specific than required if so desired.

   Any virtual machine (VM) used during testing shall be listed in the OS field of the Operating Environment (OE). If the VM was running between the software implementation and the OS, as in the case of a Java VM, it should be listed along with the OS using the same vendor and family/version number requirements. See **GEN.17** for the case of a VM running underneath the target OS.

For Firmware implementations, the following information must be listed:

- Processor – This field has the same requirements as the Software Processor field above.

For Hardware implementations, the environment is the actual hardware device itself. The device should be named and described when submitted to the CAVP.

If a cryptographic algorithm implementation cannot be tested in its hardware environment, per FIPS 140-3 CMVP Management Manual section 7.2 (the corresponding reference for FIPS 140-2 is IG G.11), a simulator may be used to test the algorithm implementation. The algorithm implementation would be extracted from the rest of the hardware implementation and tested with a simulator. In this case, the implementation would be firmware and the operating environment would list the name of the simulator used to test the implementation. Examples of simulator names include Cadence NC- Verilog, Mentor Graphics ModelSim 10, and Synopsys VCS.

GEN.14 Can a vendor still get a hard copy algorithm validation certificate?

No. Effective January 1, 2008, the Cryptographic Algorithm Validation Program (CAVP) stopped issuing algorithm validation certificates for cryptographic algorithm validations. The cryptographic algorithm implementation validation entry (found on CSRC) will serve as the official posting of the validation.

GEN.17 Does CAVP allow algorithm tests to be performed on a target operating system running on top of a virtual machine? Or must the target OS be the machine's native OS? What about containers?

The CAVP allows algorithm tests to be performed on a guest operating system running on a hypervisor; however, the full virtualization environment shall be specified in the OE listing as described below.

For a Type 1 (or native) hypervisor, where the hypervisor runs directly on the hardware, the OE listing shall include the guest OS, hypervisor, and processor using the following format: "Guest OS on hypervisor on Processor." An example is "Microsoft Windows 11 on VMWare ESXi 7.0 on Intel Xeon W (Rocket Lake)."

For a Type 2 (or hosted) hypervisor, where the hypervisor runs on a host operating system (OS), the OE listing shall include the guest OS, hypervisor, host OS, and processor using the following format: "Guest OS on hypervisor on Host OS on Processor." An example is "Microsoft Windows 11 on Parallels Desktop 17 on macOS Monterey on Intel Core i7 (Kaby Lake)."

Algorithm tests may also be performed using containers. The OE listing shall include the container, runtime, host OS, and processor using the following format: "container on runtime on host OS on processor." An example is "Ubuntu 20.04 Docker Image on Docker Engine 20 on Ubuntu 20.04 on Intel Xeon W (Rocket Lake)."

Use the wording in the formats and examples above exactly. Contact the CAVP if you have a virtualized environment that is not accurately or sufficiently described by the above.

GEN.20 Source code for a cryptographic algorithm is compiled into two separate, non-identical binary files. Can the two binary (executable) files be considered a single implementation? (For example, AES is compiled statically into both an encrypted key storage system and a network encryption system inside the same product; or ECDSA verification is compiled into a pre-boot loader and into the main program that the pre-boot loader loads.)

No. The CAVP validation for a software or firmware implementation is specific to the binary executable file that the tested implementation resides in. Compiling the same source code into two different executables is considered two separate algorithm implementations and each one needs to be independently tested. It does not matter that the same source code is used.

GEN.21 Can a vendor request that an algorithm implementation be validated but not posted on the validation list until a later date?

No. When a validation number is assigned to the implementation, it must be posted on the CAVP algorithm validation list.

If the vendor does not want this algorithm implementation to be publicly recognized until a later date, the vendor can assign a temporary implementation name when the implementation is submitted to the CAVP by the testing lab. The fact that a temporary name has been assigned to this implementation is transparent to the CAVP; the name of an algorithm implementation is the responsibility of the vendor. The rest of the implementation information displayed on CAVP web site – the vendor information, the versioning, Operational Environment, and the Description/Notes field – shall represent the information about what was tested.

Please see section 12 of the ACVP specification (https://pages.nist.gov/ACVP/draft-fussell-acvp-spec.html) for how to update the information for an existing validation.

An exception to this guidance is ITAR validations. See the ***CAVP Management Manual*** for instructions on processing ITAR requests.

GEN.28 I wish to have one or more algorithms from my implementation validated on a separate certificate number from the main implementation certificate. How do I do so?

If a user wishes to have a separate certificate for some subset of algorithms in their implementation, simply create a second implementation metadata object and certify against it.

When doing so, the user must include some differentiating information in the new implementation's name.

The following provides several examples of how a user may differentiate between the implementation names:

Original Implementation Name: Acme Crypto Module

Separate Implementation Name: "Acme Crypto Module (AES)" or "Acme Crypto Module - AES" or "Acme Crypto Module - Separate Unit Name Here"

4. AES FAQ

AES.2 A software implementation of AES uses the AES-NI instruction set. How do I validate it?

There are two separate cases:

1. The implementation relies on the AES-NI instructions and only runs on processors that support them. One set of AES test vectors is needed to validate this implementation. The OE listing for the processor must indicate that the processor supports AES-NI, e.g., "Microsoft Windows 11 on Intel Xeon W (Rocket Lake) with AES-NI."

2. The implementation uses AES-NI on processors that support it and does not use AES-NI (i.e., implements AES entirely in software) on processors that do not support it. Thus, there are two distinct execution paths in the code for AES depending upon whether the processor supports AES-NI. In this case, two sets of test vectors are needed to validate the implementation: one runs on a processor with AES-NI that uses the AES instructions, and one runs on a processor without AES-NI.

The above requirements apply to any mode of AES and to any algorithm that uses AES, such as the CTR_DRBG and CMAC.

The above also applies to AES accelerator functions implemented in other processors, e.g., ARM.

5. Triple-DES FAQ

TDES.2 Our TDES implementation does not allow the use of weak keys, but the Known Answer Tests (KATs) in the validation suite use weak keys and, therefore, the implementation needs to be able to accept them. How should we test this?

For validation testing, tighten the algorithmic boundary so that it does not include the weak key check. Make sure the implementation does not allow weak keys outside the validation testing.

TDES.3 Why do the TDES Known Answer Tests (KATs) use weak keys?

The TDES Known Answer Tests (KATs) were based on the DES KATs. Likewise, the DES KATs were based on the standard DES test set described in NIST SP 500-20, "Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard," written in 1977. These tests were generated before the realization of the weak keys. The purpose of this test is to test every element of the DES (TDES) components. When talking about the key tests, we are referring to the testing of the key permutation tables PC1 and PC2. As detailed in Section 3.1.1.3, "The Variable Key Known Answer Test for the Encryption Process" in NIST SP 800-17:

"When this test is performed for an IUT of the DES algorithm, the 56 possible basis vectors which yield unique keys are presented to PC1 verifying the key permutation, PC1. Since the key schedule consists of left shifts, as *i* ranges over the index set, a complete set of basis vectors is presented to PC2 as well, so this is verified."

TDES.5 After January 1, 2016, does the CAVP still have validation testing for 2-Key TDES (or Keying Option 2)?

SP800-131A Revision 1 dated November 2015 states that, as of January 1, 2016, TDES Keying Option 2, Encrypt mode is non-compliant. TDES Keying Option 2, Decrypt mode is allowed for legacy use only.

Therefore, for 2-key TDES, the CAVP will only test the decrypt state.

6. DSA FAQ

DSA.1 If a vendor is having problems getting one of the DSA functions to work properly, where can a known set of values be obtained to help in the testing?

Test vectors for all validation tests for every supported algorithm can be obtained from the ACVTS Demo Server. For tests that include intermediate values, see the appropriate algorithm section at https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values.

If the implementation being tested does not compute the same signature or result, then it can be concluded that there is something wrong with the vendor's implementation.

DSA.4 In section 4.2 Selection of Parameter Sizes and Hash Functions for DSA of FIPS 186-4 it states:

"When the length of the output of the hash function is greater than N (i.e., the bit length of q), then the leftmost N bits of the hash function output block shall be used in any calculation using the hash function output during the generation or verification of a digital signature."

Who is supposed to do this truncation? The calling application (calling a crypto library's signature generation API) or the crypto library itself?

If the crypto library is the implementation getting validated, then the entire digital signature (generation or verification) algorithm needs to be performed within the crypto library. This means that the crypto library must execute all the applicable shall statements, including the truncation of the hash function output.

If the crypto library is not the complete implementation and the calling application calls the hash function separately from the signing function, it is the application's responsibility.

7. RSA FAQ

RSA.1 If a vendor is having problems getting one of the RSA functions to work properly, where can a known set of values be obtained to help in the testing?

Test vectors for all validation tests for every supported algorithm can be obtained from the ACVTS Demo Server. For tests that include intermediate values, see the appropriate algorithm section at https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values.

If the implementation being tested does not compute the same signature or result, then it can be concluded that there is something wrong with the vendor's implementation.

8. HMAC FAQ

HMAC.4 If an HMAC implementation uses a SHA implementation that cannot be accessed independent of HMAC, does the SHA algorithm have to be tested? Why?

Yes. SHA testing is a prerequisite of HMAC testing. When an implementation of the HMAC algorithm is validated, the CAVP requires that the SHA algorithm used by the HMAC implementation be validated.

HMAC algorithm implementations rely on the correctness of the underlying SHA implementation used. The HMAC testing alone does not provide adequate testing of the SHA algorithm; the HMAC tests focus on testing the HMAC processing only. As this is the case, CAVP requires testing of the underlying SHA implementation and SHA is considered a prerequisite for HMAC testing.

This requirement cannot be bypassed.

See GEN.5 for more information about algorithm prerequisites.

9. ECDSA FAQ

ECDSA.3 Can an ECDSA implementation be validated if it does not use any NIST-recommended curves?

No. To validate an implementation of ECDSA, the algorithm implementation must implement at least one NIST-recommended curve. It can have non-recommended NIST curves if there is at least one NIST-recommended curve.

ECDSA.6 In section 6.4 ECDSA Digital Signature Generation and Verification of FIPS 186-4 it states:

"When the length of the output of the hash function is greater than the bit length of n, then the leftmost n bits of the hash function output block shall be used in any calculation using the hash function output during the generation or verification of a digital signature. "

Who is supposed to do this truncation? The calling application (calling a crypto library's signature generation API) or the crypto library itself?

If the crypto library is the implementation getting validated, then the entire digital signature (generation or verification) algorithm needs to be performed within the crypto library. This means that the crypto library must execute all the applicable shall statements, including the truncation of the hash function output.

If the crypto library is not the complete implementation and the calling application calls the hash function separately from the signing function, it is the application's responsibility.

10. CMAC FAQ

CMAC.3 After January 1, 2016, does the CAVP still have validation testing for CMAC using 2-Key TDES?

SP800-131A Revision 1 dated November 2015 states that, as of January 1, 2016, TDES Keying Option 2, encrypt mode is non-compliant.

Therefore, the testing of CMAC encrypt, i.e., gen, using 2-Key TDES is no longer available. Testing of CMAC decrypt, i.e., ver, using 2-Key TDES is available.

11. GCM FAQ

GCM.3 Why does CAVP allow testing for an externally generated IV when Section 9.1 of NIST SP 800-38D says the module must generate IVs within the module boundary? (FIPS 140-3 IG C.H has a similar statement; the corresponding reference for FIPS 140-2 is IG A.5).

The CAVP validates cryptographic algorithm implementations, not cryptographic modules. The cryptographic algorithm boundary does not have to be the same as the cryptographic module boundary. In many cases the cryptographic algorithm implementation is itself a module, but in other cases it is part of (i.e., a component of) a cryptographic module. Thus, for example, a validated AES GCM implementation could be combined with a validated approved RBG implementation that will generate IVs in a crypto module. The requirements in Section 9.1 of NIST SP 800-38D apply to a cryptographic module and module validation under FIPS 140-3, as does the similar text in FIPS 140-3 IG C.H. The corresponding reference for FIPS 140-2 is IG A.5

12. DRBG FAQ

DRBG.1 Is ANSI X9.62-2005 DRBG the same as NIST SP 800-90A HMAC_DRBG?

Yes, they are the same. NIST SP 800-90A has a nonce as one of the inputs to the instantiate function, which is not in ANSI X9.62-2005. However, NIST SP 800-90A Section 8.6.7 specifies that additional entropy input can be used in place of a nonce, making it the same as ANSI X9.62-2005. An implementation of an ANSI X9.62-2005 DRBG can be validated by passing the CAVP HMAC_DRGB tests.

DRBG.3 Does CTR_DRBG use AES (or TDES) in counter mode or, as NIST SP 800-90Ar1 Sections 10.2.1 and 10.3.3 seem to indicate, in ECB mode?

In AES counter mode (or TDES counter mode), the forward cipher function (sometimes called the encrypt function) has a counter value as the input instead of plaintext (ECB mode), a chained ciphertext value (CBC mode), or some other value. The counter has an initial value, not necessarily 0, and is incremented between calls to the forward cipher function. NIST SP 800-90Ar1 Section 10.2.1.5.1 Step 4 implements the counter mode. `V` is the counter; it is incremented at the beginning of each pass through the loop. The `Block_Encrypt` (forward cipher function) operation is performed on the counter and the result is appended to the end of the `temp` bitstring.

The only difference between how counter mode is used here and how it is used for encryption is that when used for encryption, the result of the "encrypt counter" operation is XOR'ed with the plaintext. Here, inside the DRBG, we are not encrypting anything, so result is returned as pseudorandom bits and there is no XOR step. The Update function in NIST SP 800-90Ar1 10.2.1.2 (`CTR_DRBG_Update`), Step 2 implements a counter in the same way and does include an XOR of the result with `provided_data` to produce an updated internal state (`Key` and `V`).

DRBG.5 May an implementation use two distinct sequential calls to Reseed and Generate to pass the CAVP tests and claim support for Prediction Resistance?

No. NIST SP 800-90Ar1 Section 9.3.1, Step 7 defines how prediction resistance shall be implemented as part of the Generate function. Requiring two distinct calls by the user (consumer) of the DRBG, first to Reseed and second to Generate, is not acceptable. This is a particular case of the general requirement in GEN.2.