PKCS#11 Compliance Report
CloudHSM SDK 3.2.1

PKCS#11 Compliance Report: CloudHSM SDK 3.2.1

Table of Contents

PKCS#11 Compliance Report: CloudHSM SDK 3.2.1

Publication date: November 1, 2020 (Document Revisions on page 34)

Executive Summary

PKCS#11, also known as Cryptoki, is a widely adopted C-language API and interoperability standard for communicating with cryptographic libraries. PKCS#11 developers rely on libraries complying with the specification, particularly under error conditions. When a PKCS#11 library diverges from the standard, customer impact can range from delays in development to potential data corruption if an application makes incorrect assumptions or is imperfectly tested.

To address this issue, Galois partnered with AWS's automated reasoning group and the CloudHSM product team to synthesize a suite of PKCS#11 compliance tests based on Galois's formal model of the PKCS#11 standard that Galois developed.

We tested AWS CloudHSM's PKCS#11 library version 3.2.1 using this suite. We ran 40,861 tests, resulting in 54 identified issues. Some issues are due to limitations in the software library while others reflect underlying HSM behavior. Developers should review known issues identified in this report and follow recommendations to manage impact, to ensure their application behaves as expected.

Introduction

PKCS#11, also known as Cryptoki, is a widely adopted C-language API and interoperability standard for communicating with cryptographic libraries. AWS CloudHSM's PKCS#11 library is built to comply with the PKCS#11 v2.40 specification. To verify our compliance, we partnered with Galois, who specialize in applying formal analysis techniques to model systems, analyze them and seek to prove them correct. We test compliance of our library using a model-based PKCS#11 testing framework from Galois. We supplement this framework with our own tests for functions which Galois does not yet support, such as key derivation and trusted key management.

Our testing is intended to cover all functions, mechanisms, attributes and key types that CloudHSM supports. You can find a complete list of supported features in our online documentation. This report summarizes the findings of this compliance testing on PKCS#11 SDK version 3.2.1.

Understanding the Methodology

The PKCS#11 specification is comprehensive and prescriptive, but also extremely complex. The base specification alone describes approximately 50 functions through over 100 pages of documentation. Add-on specifications provide additional functionality, but also impose additional complexity. As the standard evolves through collaboration and expansion, new areas of imprecision and ambiguity are introduced which make it difficult for vendors to implement libraries that are 100% functionally accurate and compliant with the specification.

PKCS#11 is hard to model and test. Each cryptographic operation supported by the Cryptoki API is defined (directly or by inference) in terms of a set of functions necessary for performing the operation from beginning to end. For each function, the standard gives a description of how it interacts statefully with other functions in the operation group. While the basic usage pattern appears intuitive on the surface, failures in function invocations often result in deceptive state transitions. This behavior is not explicitly specified, but rather, must be inferred by examining the return codes listed for a function. Return codes are intended to provide a behavioral contract for each function, with each value having an associated set of constraints over the operation state and function arguments. The conditions of this contract are often not made explicit when API functions are described in the standard. Furthermore, the list of behaviors is not guaranteed to be exhaustive. Adding an additional layer of complexity, the standard also defines a partial order over the set of all return codes. This order dictates what return code applies when the conditions for multiple errors are met in a single function invocation.

To address this problem at its core, Galois has developed an API specification and testing framework that captures the complex behaviors of Cryptoki in a mathematical specification language. This specification, in turn, is used to automatically synthesize a test suite that enforces compliance with (a mathematical model of) the PKCS#11 standard. The approach, known as model-based testing, was used to generate a corpus of over 40,000 PKCS#11 compliance tests that provide complete test coverage over the formal model.

To ensure the model faithfully represents the requirements in the specification, the work was done in close collaboration with Amazon Web Services' automated reasoning group and the CloudHSM product team. To ensure customers are provided with high-assurance implementations of Cryptoki, we have integrated this test suite into CloudHSM's code pipeline.

Advantages and Limitations

The tests help enforce compliance with the PKCS#11 standard and reduce defects in production software. Error handling scenarios defined in the standard often represent corner cases which, if not properly handled by implementation code, will cause program crashes or other serious defects. Model-based testing is designed to generate test cases for all the behaviors in the standard, so it naturally uncovers a wider range of edge cases than manually created test suites. As a result, model-based testing drives PKCS#11 libraries to be more portable, more robust and have fewer security vulnerabilities.

That said, there are limitations to this approach. Since we use testing to enforce compliance, it is not possible to guarantee implementations that pass our test suite are completely free of compliance related errors. For each requirement in the specification, we use a representative set of example configurations and values to test the requirement. For example, when testing a transition in an encryption operation, our test suite picks sample values for the plaintext and cryptographic key. If a library has a compliance error that is specific to individual data elements, such as a specific bit pattern in the plaintext, our test suite is not likely to uncover it. This is an inherent limitation of black-box testing approaches that don't assume any knowledge of the system under test. On the other hand, most compliance errors are not data specific, and the black-box approach allows our tests to be run against any PKCS#11 library. Additionally, our test framework does not cover behaviors that cannot be reliably reproduced by client applications, such as network outages and hardware/token failures.

Coverage

The Galois test suite models and tests the following functions:

- Encrypt (C_EncryptInit, C_Encrypt, C_EncryptUpdate, C_EncryptFinal)
- Decrypt (C_DecryptInit, C_Decrypt, C_DecryptUpdate, C_DecryptFinal)
- Digest (C_DigestInit, C_Digest, C_DigestUpdate, C_DigestFinal)
- Sign (C_SignInit, C_Sign, C_SignUpdate, C_SignFinal)
- Verify (C_VerifyInit, C_Verify, C_VerifyUpdate, C_VerifyFinal)
- Sign Recover (C_SignRecoverInit, C_SignRecover)
- Verify Recover (C_VerifyRecoverInit, C_VerifyRecover)
- Slot and Session Management (C_GetSlotList, C_GetSlotInfo, C_GetTokenInfo, C_GetMechanismList, C_GetMechanismInfo, C_GetFunctionList, C_GetInfo, C_GetSessionInfo, C_GetOperationState)
- Object Management (C_CreateObject, C_DestroyObject, C_CopyObject, C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal, C_GetAttributeValue)
- User Management (C_Login, C_Logout)
- Library Management (C_Initialize, C_Finalize)
- Key Generation (C_GenerateKey, C_GenerateKeyPair)

- Key Management (C_WrapKey, C_UnwrapKey)

In addition, the test framework provides test cases that ensure the following properties hold for the attributes associated with cryptographic keys.

1. Attribute values specified by key creation or import templates are correctly configured in the generated keys.
2. The proper default attribute values are assumed when they are not specified by the key creation or import templates.
3. Keys can only be created or imported from templates when all of the required attributes (as defined by the key type) are supplied.
4. Keys cannot be created or imported from templates when the template contains an invalid attribute type.
5. Keys cannot be created or imported when templates contain conflicting values for attributes.

For functions that the Galois framework does not yet support, such as key derivation and trusted key management, we supplement the framework with internal unit tests.

Summary of Results

These test results are for CloudHSM PKCS#11 SDK version 3.2.1. We ran 48,302 tests, and identified 54 issues in the following categories:

- **Non-Compliant Output** (1): The CKM_AES_KEY_WRAP mechanism applies PKCS5 padding instead of no-padding. You can learn about choosing the right mechanism for your AES key wrapping needs using CloudHSM PKCS#11 SDK in our online documentation.
- **State Enforcement** (6): The library does not enforce state for multi-part sign, verify, encrypt and decrypt operations. As an example, if an application calls C_Encrypt after C_EncryptUpdate (or vice-versa), the library will deliver incorrect results instead of returning an error. We expect to address these issues in time. Applications should be written in conformance with the PKCS#11 specification, utilizing either a single-part operation or a series of multi-part operations, but not both, in a given context.
- **Unsupported Key Attributes** (20): The HSM does not support or correctly enforce all attributes specified by the PKCS#11 standard. Of the 20 issues identified, 4 are noteworthy. The HSM does not, at present, enforce CKA_SIGN and CKA_VERIFY on symmetric or secret keys (for clarity, these attributes are correctly enforced for asymmetric keys such as RSA and EC keys). The HSM also does not support CKA_ALWAYS_AUTHENTICATE set to TRUE for private keys, although no key on the HSM (irrespective of the value of this attribute) can be accessed without authenticating. The HSM does not restrict EC keys from being used to derive other keys, generally as part of an ECDH operation, even if CKA_DERIVE is set to FALSE.
- **Unsupported Arguments** (3): The HSM does not permit certain values that the PKCS#11 specification permits, due to FIPS security requirements. The HSM does not permit a NULL password

(i.e. pPin) in C_Login, and the library does not handle NULL input. The HSM does not permit customer-defined AES-GCM initialization vectors in encrypt and wrap calls. CloudHSM recommends the safer but CloudHSM-proprietary CKM_CLOUDHSM_AES_GCM mechanism instead of CKM_AES_GCM.

- **Key Restrictions** (2): The HSM permits AES keys to be used in place of secret keys for HMAC sign and verify operations. Applications, as a general best practice, should ensure they utilize a given key for encryption or signing - not both.
- **Initialization Checks** (9). The library does not perform a required check when an operation is initialized. As a result, subsequent operations may return an error even if the initialization succeeds. These are minor issues, which we expect to address in time. Applications, as a general best practice, should check for success or error on every function call.
- **Error Codes** (13): The library returns a less specific error code, or returns an out of order error code when multiple errors occur together. These are minor issues, which we expect to address in time. Applications, as a general best practice, should handle any error returned by a function call.

Note that PKCS#11 compliance does not require all mechanisms and functions that is specifies to be implemented. It simply requires that, where implemented, functions and mechanisms behave according to the specification. For unsupported functions such as C_GetOperationState or C_GetObjectSize, we have verified that CloudHSM PKCS#11 returns CKR_FUNCTION_NOT_SUPPORTED.

Details of Results

The rest of this document provides detailed results, grouped by function category as listed in Cryptoki. The functions we test in each category are listed at the beginning of the corresponding section. If you see no further details, this means the function/mechanism passed all related tests. We disclose any deviations we observed from the specification by summarizing the expected behavior (the PKCS#11 Rule) and the observed behaviors (the Result). We include an assessment of the classification and impact for each deviation, so you can prioritize workarounds as appropriate.

Session, User, Object and Library Management Functions

We ran 9,786 tests covering the following functions:

- C_Initialize
- C_Finalize
- C_Login
- C_Logout
- C_GetSlotList
- C_GetSlotInfo
- C_GetTokenInfo
- C_GetMechanismList
- C_GetMechanismInfo
- C_GetFunctionList
- C_GetInfo
- C_GetSessionInfo
- C_GetOperationState
- C_CreateObject
- C_DestroyObject
- C_CopyObject
- C_FindObjectsInit
- C_FindObjects
- C_FindObjectsFinal
- C_GetAttributeValue

We identified 2 issues, as below:

Issue ID	SESS-1
PKCS#11 Rule	The *pPin* parameter to C_Login maybe set to NULL_PTR
Result	The HSM does not permit the *pPin* parameter to C_Login to be NULL, because you must login to the HSM with a username and password. The library does not handle NULL_PTR, and causes undetermined behavior.
Classification	Unsupported argument
Impact	Applications that pass untrusted values to C_Login without validation will crash. CloudHSM does not support “protected authentication path” which would require setting *pPin* to NULL_PTR. Applications must ensure that *pPin* is set to a valid value when logging in.

Issue ID	SESS -2

PKCS#11 Rule	When C_FindObjectsInit is called after a successful call to C_FindObjectsInit, the library should return CKR_OPERATION_ACTIVE.
Result	The library returns CKR_OK instead of CKR_OPERATION_ACTIVE.
Classification	State Enforcement
Impact	The library does not enforce the correct order of operations and may result in incorrect results. Applications should be written to keep track of initialization state, and not rely on the library to infer state.

Signing and MACing functions

We ran 10,446 tests covering the following functions:

- C_SignInit
- C_Sign
- C_SignUpdate
- C_SignFinal
- C_SignRecoverInit
- C_SignRecover

We identified 8 issues as below:

Issue ID	SIGN-1
PKCS#11 Rule	C_SignInit shall return CKR_KEY_FUNCTION_NOT_PERMITTED for a key with the CKA_SIGN attribute set to CK_FALSE.
Result	The HSM incorrectly allows AES and generic secret keys with CKA_SIGN set to CKA_FALSE to be used with signing mechanisms, so the operation succeeds instead of failing.
Classification	Unsupported Key Attributes
Impact	AES and Generic Secret keys may be incorrectly used for signing mechanisms in spite of the CKA_SIGN attribute being set to False. For clarity, RSA and EC signing operations will correctly fail if CKA_SIGN set to CK_FALSE.

Issue ID	SIGN -2
PKCS#11 Rule	When C_Sign is called after a successful call to C_SignUpdate, the library should return CKR_OPERATION_ACTIVE.
Result	The library returns CKR_OK instead of CKR_OPERATION_ACTIVE.
Classification	State Enforcement
Impact	The library does not enforce the correct order of operations and may result in incorrect results. Applications should be written to comply with the specification, and not rely on the library to catch this error.

Issue ID	SIGN-3
PKCS#11 Rule	C_SignInit shall fail if the user is not logged in.
Result	When C_SignInit is called with the mechanism CKM_SHA_1_HMAC, the library returns CKR_OK even if no user is logged into the HSM.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call requiring a logged-in session is made. Applications should be prepared to handle errors in C_Sign and other functions even if C_SignInit succeeded.

Issue ID	SIGN-4
PKCS#11 Rule	C_SignRecoverInit shall return CKR_MECHANISM_PARAM_INVALID if any invalid input parameters are passed.
Result	When the *pParameter* value is set to a non-NULL value for a mechanism that does not accept a parameter value, C_SignRecoverInit returns CKR_OK instead of CKR_MECHANISM_PARAM_INVALID.
Classification	Initialization Checks
Impact	The library does not validate all the inputs to a function. If appropriate, the application should validate input before calling this function.

Issue ID	SIGN-5
PKCS#11 Rule	C_SignInit shall only allow generic keys to be used with CKM_SHA_1_HMAC, CKM_SHA_224_HMAC, CKM_SHA_256_HMAC, CKM_SHA_384_HMAC and CKM_SHA_512_HMAC mechanisms
Result	The library allows keys of type CKK_AES to be used with mechanisms to compute HMAC
Classification	Key Restrictions
Impact	There is no impact assuming users follow proper key management policies such as using a given key only for one operation type.

Issue ID	SIGN-6
PKCS#11 Rule	When an invalid key handle is supplied to C_SignInit, it should fail with CKR_KEY_HANDLE_INVALID.
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID.
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	SIGN-7
PKCS#11 Rule	When an invalid key handle is supplied to C_SignRecoverInit, it should fail with CKR_KEY_HANDLE_INVALID.
Result	The library returns CKR_OBJECT_HANDLE_INVALID, instead of the more specific CKR_KEY_HANDLE_INVALID, when an invalid key handle is passed to it.
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	SIGN-8
PKCS#11 Rule	C_SignRecoverInit shall return CKR_KEY_TYPE_INCONSISTENT if the key type is not consistent with the mechanism
Result	C_SignRecoverInit returns CKR_MECHANISM_INVALID if the mechanism is not consistent with the key type.
Classification	Error Codes
Impact	The library returns a misleading error code that could hinder troubleshooting.

Functions for verifying signatures and MACs

We ran 616 tests covering the following functions:

- C_VerifyInit
- C_Verify
- C_VerifyUpdate
- C_VerifyFinal
- C_VerifyRecoverInit
- C_VerifyRecover

We identified 8 issues as below:

Issue ID	VERIFY-1
PKCS#11 Rule	C_VerifyInit shall return CKR_KEY_FUNCTION_NOT_PERMITTED for a key with the CKA_VERIFY attribute set to CK_FALSE.
Result	The HSM incorrectly allows AES and generic secret keys with CKA_VERIFY set to CKA_FALSE to be used with HMAC signature verification mechanisms.
Classification	Unsupported Key Attributes
Impact	AES and Generic Secret keys may be incorrectly used for verification mechanisms in spite of the CKA_VERIFY attribute being set to False. For clarity, RSA and EC verify operations will correctly fail if CKA_VERIFY is set to CK_FALSE.

Issue ID	VERIFY-2
PKCS#11 Rule	A call to C_VerifyInit must fail if the user is not logged in.
Result	When C_VerifyInit is called with the mechanism CKM_SHA_1_HMAC, the library returns CKR_OK even if no user is logged into the HSM. Other mechanisms are not affected by this issue.
Classification	Initialization Checks
Impact	Applications will misleadingly continue working until a different call requiring a logged in session is made. Applications should be prepared to handle errors in C_Verify and other verify functions even if C_VerifyInit succeeded.

Issue ID	VERIFY-3
PKCS#11 Rule	C_VerifyRecoverInit shall return CKR_MECHANISM_PARAM_INVALID if any invalid input parameters are passed
Result	When the *pParameter* value is set to a non-NULL value for a mechanism that does not accept a parameter value, C_VerifyRecoverInit returns CKR_OK instead of CKR_MECHANISM_PARAM_INVALID.
Classification	Initialization Checks
Impact	The library does not validate all the inputs to a function. If appropriate, the application should verify input before calling C_VerifyRecoverInit.

Issue ID	VERIFY-4
PKCS#11 Rule	When an invalid key handle is supplied to C_VerifyInit, it shall fail with CKR_KEY_HANDLE_INVALID
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	VERIFY-5
PKCS#11 Rule	When an invalid key handle is supplied to C_VerifyInit, it shall fail with CKR_KEY_HANDLE_INVALID.
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID.
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	VERIFY-6

PKCS#11 Rule	C_VerifyInit shall only allow keys of type CKK_GENERIC_SECRET to be used with CKM_SHA_1_HMAC, CKM_SHA_224_HMAC, CKM_SHA_256_HMAC, CKM_SHA_384_HMAC and CKM_SHA_512_HMAC mechanisms.
Result	The library allows keys of type CKK_AES to be used with mechanisms to compute HMAC.
Classification	Key Restrictions
Impact	There is no impact assuming users follow proper key management policies such as using a given key only for one operation type

Issue ID	VERIFY-7
PKCS#11 Rule	When an invalid key handle is supplied to C_VerifyRecoverInit, it should fail with CKR_KEY_HANDLE_INVALID
Result	The library returns CKR_OBJECT_HANDLE_INVALID, instead of the more specific CKR_KEY_HANDLE_INVALID, when an invalid key handle is passed to it
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	VERIFY-8
PKCS#11 Rule	C_VerifyRecoverInit shall return CKR_KEY_TYPE_INCONSISTENT if the key type is not consistent with the mechanism.
Result	C_VerifyRecoverInit returns CKR_MECHANISM_INVALID if the mechanism is not consistent with the key type.
Classification	Error Codes
Impact	The library returns a misleading error code that could hinder troubleshooting.

Message digest functions

We ran 2308 tests covering the following functions:

- C_DigestInit
- C_Digest
- C_DigestUpdate
- C_DigestFinal

We identified 3 issues as below:

Issue ID	DIGEST-1
PKCS#11 Rule	When C_Digest is called after a successful call to C_DigestUpdate, the library shall return CKR_OPERATION_ACTIVE.
Result	The library incorrectly returns CKR_OK when C_Digest is called after C_DigestUpdate.
Classification	State Enforcement
Impact	The library does not enforce the correct sequence of operations which will lead to incorrect results. Applications should be written in compliance with the PKCS#11 specification, and not rely on the library to enforce sequence of operations.

Issue ID	DIGEST- 2
PKCS#11 Rule	C_Digest shall terminate the active digest context after a sequence of C_Digest_Init/C_Digest is called.
Result	The library incorrectly returns CKR_OK when C_Digest or C_DigestUpdate is called after a successful call to C_Digest.
Classification	State Enforcement
Impact	Applications will generate incorrect digest values if they make the incorrect sequence of digest operation calls. Applications should be written in compliance with the PKCS#11 specification, and not rely on the library to enforce sequence of operations.

Issue ID	DIGEST-3
PKCS#11 Rule	C_DigestInit shall fail if no user is logged in

Result	C_DigestInit returns CKR_OK even if no user is logged into the HSM.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call requiring a logged-in session is made. Applications should be prepared to handle errors in C_Digest and other digest functions even if C_DigestInit succeeded.

Encryption functions

We ran 8011 tests covering the following functions:

- C_EncryptInit
- C_Encrypt
- C_EncryptUpdate
- C_EncryptFinal

We identified 6 issues as below:

Issue ID	ENCRYPT-1
PKCS#11 Rule	If C_EncryptUpdate is called after a successful call to C_Encrypt, the library shall return CKR_OPERATION_ACTIVE.
Result	The library returns CKR_OK when an invalid sequence of calls is made, like C_EncryptUpdate after a successful call to C_Encrypt.
Classification	State Enforcement
Impact	The library returns success instead of an error. An application that does not comply with the specification will experience silent loss of data. Your application should not rely on the library to enforce correct sequence of operations.

Issue ID	ENCRYPT-2
PKCS#11 Rule	The CKM_AES_GCM mechanism shall use the IV specified in the pIV variable in the CK_GCM_PARAMS structure if pIV is not NULL.
Classification	Unsupported Arguments
Result	The library ignores the initialization vector provided at the pIV and overwrites it with an HSM generated IV.
Impact	The library enforces that the user-provided IV be all-zeros, and overwrites the user-provided IV with an HSM-generated value. If your application does not save this returned IV, you will be unable to decrypt the content, resulting in data loss. NOTE: CloudHSM recommends the safer but CloudHSM-proprietary CKM_CLOUDHSM_AES_GCM mechanism instead of CKM_AES_GCM.

Issue ID	ENCRYPT-3
PKCS#11 Rule	C_EncryptInit shall return CKR_MECHANISM_INVALID if the mechanism is not consistent with the function.
Result	C_EncryptInit returns CKR_OK when called with an invalid mechanism, say CKM_AES_KEY_WRAP. A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call is made. Applications should be prepared to handle errors in encryption functions even if C_EncryptInit succeeded.

Issue ID	ENCRYPT-4
PKCS#11 Rule	C_EncryptInit shall return CKR_MECHANISM_PARAMS_INVALID if the parameters are not consistent with the mechanism.
Result	1. C_EncryptInit returns CKR_OK when invalid parameters, such as an invalid IV length, are passed with the CKM_AES_GCM mechanism. A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD. 2. C_EncryptInit returns CKR_OK when invalid parameters, such as an invalid hash mechanism, is passed with the CKM_RSA_PKCS_OAEP mechanism. A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call is made. Applications should be prepared to handle errors in encryption functions even if C_EncryptInit succeeded.

Issue ID	ENCRYPT-5
PKCS#11 Rule	When an invalid key handle is supplied to C_EncryptInit, it shall fail with CKR_KEY_HANDLE_INVALID.
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID.
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	ENCRYPT-6
PKCS#11 Rule	C_EncryptInit shall return CKR_MECHANISM_INVALID if the mechanism and key type are not consistent with the function.
Result	C_EncryptInit returns CKR_KEY_TYPE_INCONSISTENT if the mechanism and key type are both invalid for the encryption operation. For example, this occurs when the mechanism type is CKM_RSA_X9_31_KEY_PAIR_GEN and the key type is CKK_AES.
Classification	Error Codes
Impact	The library returns a misleading error code that could hinder troubleshooting.

Decryption Functions

We ran 9474 tests covering the following functions:

- C_DecryptInit
- C_Decrypt
- C_DecryptUpdate
- C_DecryptFinal

We identified 5 issues as below:

Issue ID	DECRYPT-1
PKCS#11 Rule	If C_DecryptUpdate is called after a successful call to C_Decrypt, the library shall return CKR_OPERATION_ACTIVE
Result	The library returns CKR_OK when an invalid sequence of calls is made, like C_DecryptUpdate after a successful call to C_Decrypt.
Classification	State Enforcement
Impact	The library returns success instead of an error. An application that does not comply with the specification will receive unpredictable results from decryption. Your application should not rely on the library to enforce correct sequence of operations.

Issue ID	DECRYPT-2
PKCS#11 Rule	C_DecryptInit shall return CKR_MECHANISM_INVALID if the mechanism is not consistent with the function.
Result	C_DecryptInit returns CKR_OK when called with an invalid mechanism, say CKM_AES_KEY_WRAP. A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call is made. Applications should be prepared to handle errors in decryption functions even if C_DecryptInit succeeded.

Issue ID	DECRYPT-3
PKCS#11 Rule	C_EncryptInit shall return CKR_MECHANISM_PARAMS_INVALID if the parameters are not consistent with the mechanism.

Result	1. C_EncryptInit returns CKR_OK when invalid parameters, such as an invalid IV length, are passed with the CKM_AES_GCM mechanism. A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD. 2. C_EncryptInit returns CKR_OK when invalid parameters, such as an invalid hash mechanism, is passed with the CKM_RSA_PKCS_OAEP mechanism. when A subsequent call to C_Encrypt returns CKR_ARGUMENTS_BAD.
Classification	Initialization Checks
Impact	Applications will misleadingly continue to run until a subsequent call is made. Applications should be prepared to handle errors in decryption functions even if C_DecryptInit succeeded.

Issue ID	DECRYPT-4
PKCS#11 Rule	When an invalid key handle is supplied to C_DecryptInit, it shall fail with CKR_KEY_HANDLE_INVALID
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	DECRYPT-5
PKCS#11 Rule	C_DecryptInit shall return CKR_MECHANISM_INVALID if the mechanism and key type are not consistent with the function.
Result	C_DecryptInit returns CKR_KEY_TYPE_INCONSISTENT if the mechanism and key type are both invalid for decryption. For example, this occurs when the mechanism type is CKM_RSA_X9_31_KEY_PAIR_GEN and the key type is CKK_AES.
Classification	Error Codes
Impact	The library returns a misleading error code that could hinder troubleshooting.

Key Derivation

We ran 13 tests covering the following function:

- C_DeriveKey

Note: This section has tests results for the CKM_ECDH_DERIVE mechanism. CloudHSM's PKCS#11 SDK also supports the proprietary mechanism CKM_SP800_108_COUNTER_KDF, used to perform key derivation as specified in NIST SP800-108. This mechanism is not in scope for this report because it is not part of the PKCS#11 specification.

We identified 3 issues as below:

Issue ID	DERIVE-1
PKCS#11 Rule	When C_DeriveKey is called with a base key whose CKA_DERIVE attribute is set to FALSE, it shall fail with CKR_KEY_FUNCTION_NOT_PERMITTED.
Result	The HSM allows keys with CKA_DERIVE set to FALSE to be used to derive a key. This is an implication of how ECDH is done in several steps with the intermediate results available on the client, as described in the Known Issues section of CloudHSM public documentation.
Classification	Unsupported Key Attributes
Impact	It is not possible to restrict EC keys from being used to derive other keys.

Issue ID	DERIVE-2
PKCS#11 Rule	When an invalid key handle is supplied to C_DeriveKey, it shall fail with CKR_KEY_HANDLE_INVALID.
Result	The library returns CKR_OBJECT_HANDLE_INVALID instead of CKR_KEY_HANDLE_INVALID.
Classification	Error Codes

Impact	The library returns a less specific return code. Applications should handle both error codes.

Issue ID	DERIVE-3
PKCS#11 Rule	When an invalid key type is supplied to C_DeriveKey, it shall fail with CKR_KEY_TYPE_INCONSISTENT.
Result	When an AES key is incorrectly supplied to the C_DeriveKey function as the base key, it returns CKR_GENERAL_ERROR instead of CKR_KEY_TYPE_INCONSISTENT.
Classification	Error Codes
Impact	The library returns a less specific return code. Applications should handle both error codes.

Random Number Generation

We ran 11 tests covering the following functions:

- C_GenerateRandom
- C_SeedRandom

We did not identify any issues.

Key Management Functions

We ran 47 tests covering the following functions:

- C_GenerateKey
- C_GenerateKeyPair
- C_WrapKey
- C_UnwrapKey

We identified 2 issues as below:

Issue ID	MANAGE-1
PKCS#11 Rule	The CKM_AES_KEY_WRAP mechanism shall wrap keys with no padding, as specified in NIST SP800-38F.
Result	The CKM_AES_KEY_WRAP applies PKCS5 padding instead of no-padding.
Classification	Non-Compliant Output
Impact	The library maps a standards-defined mechanism to a custom padding algorithm which could lead to issues in inter-operating with other systems. To learn more, see AES key wrapping with CloudHSM.

Issue ID	MANAGE-2
PKCS#11 Rule	The CKM_AES_GCM mechanism shall use the IV specified in the pIV variable in the CK_GCM_PARAMS structure
Result	The library ignores the initialization vector provided at the pIV and overwrites it with an HSM generated IV.
Classification	Unsupported arguments
Impact	The library enforces that the user-provided IV be all-zeros, and overwrites the user-provided IV with an HSM-generated value. If your application does not save this returned IV, you will be unable to unwrap the key, resulting in data loss. NOTE: CloudHSM recommends the safer but CloudHSM-proprietary CKM_CLOUDHSM_AES_GCM mechanism instead of CKM_AES_GCM.

Attributes

We ran 73 tests for asymmetric key attributes and 76 tests for symmetric key attributes using the following functions:

- C_GenerateKey
- C_GenerateKeyPair
- C_CreateObject

We identified 17 issues as below:

Issue ID	ATTRIB-1
PKCS#11 Rule	The CKA_ALWAYS_AUTHENTICATE attribute shall be supported for all private keys.
Result	The HSM does not support the CKA_ALWAYS_AUTHENTICATE attribute for private keys.
Classification	Unsupported Key Attributes
Impact	Private key operations within an authenticated session cannot be protected by enforcing re-authentication.

Issue ID	ATTRIB-2
PKCS#11 Rule	The CKA_START_DATE and CKA_END_DATE attributes shall be supported for all keys.
Result	The HSM does not support the CKA_START_DATE and CKA_END_DATE attributes.
Classification	Unsupported Key Attributes
Impact	The PKCS#11 specification does not attach any meaning to the values of these attributes. The lifecycle of keys can be controlled using other means.

Issue ID	ATTRIB-3
PKCS#11 Rule	The CKA_KEY_GEN_MECHANISM attribute shall be supported for all keys.
Result	The HSM does not support the CKA_KEY_GEN_MECHANISM attribute.

Classification	Unsupported Key Attributes
Impact	You must ascertain the key generation mechanism using the type and class of the key.

Issue ID	ATTRIB-4
PKCS#11 Rule	The CKA_ALLOWED_MECHANISMS attribute shall be supported for all keys.
Result	The HSM does not support the CKA_ALLOWED_MECHANISMS attribute.
Classification	Unsupported Key Attributes
Impact	You must refer to CloudHSM public documentation to surmise which mechanisms are allowed for which keys.

Issue ID	ATTRIB-5
PKCS#11 Rule	The CKA_PRIVATE attribute shall be supported for all storage objects
Result	The HSM partially supports the CKA_PRIVATE attribute. It can be explicitly set only to the default value of CK_TRUE.
Classification	Unsupported Key Attributes
Impact	None, since the HSM only supports authenticated sessions due to FIPS security requirements. Objects with CKA_PRIVATE set to CK_FALSE – meaning objects intended to be accessed without authenticating to the HSM – cannot be stored on the HSM.

Issue ID	ATTRIB-6
PKCS#11 Rule	The CKA_MODIFIABLE attribute shall be supported for all storage objects.
Result	The HSM partially supports the CKA_MODIFIABLE attribute. It can be explicitly set only to the default value of CK_TRUE.
Classification	Unsupported Key Attributes

Impact	There are no Read Only objects in the HSM, which implies attributes for any object can be modified.

Issue ID	ATTRIB-7
PKCS#11 Rule	The CKA_COPYABLE attribute shall be supported for all storage objects.
Result	The HSM does not support the CKA_COPYABLE attribute.
Classification	Unsupported Key Attributes
Impact	None, since the HSM does not support copying stored objects.

Issue ID	ATTRIB-8
PKCS#11 Rule	The CKA_ENCRYPT attribute shall be supported for all public and secret keys.
Result	The CKA_ENCRYPT attribute is not supported for EC public or generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, since the HSM does not support encryption mechanisms using EC public or generic secret keys.

Issue ID	ATTRIB-9
PKCS#11 Rule	The CKA_VERIFY_RECOVER attribute shall be supported for all public keys
Result	1. For EC public keys, the CKA_VERIFY_RECOVER attribute is not supported 2. For RSA public keys, the CKA_VERIFY_RECOVER attribute can only be set to the same value as the CKA_VERIFY attribute for that key
Classification	Unsupported Key Attributes
Impact	None, because: 1. For EC public keys, the HSM does not support any signature verification mechanisms that allow data recovery.

	2. For RSA public keys, there are no known issues that require restricting signature mechanisms that allow data recovery compared to those that do not.

Issue ID	ATTRIB-10
PKCS#11 Rule	The CKA_WRAP attribute shall be supported for all public and secret keys.
Result	The CKA_WRAP attribute is not supported for EC public keys and generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any mechanisms that allow wrapping using EC public keys or generic secret keys

Issue ID	ATTRIB-11
PKCS#11 Rule	The CKA_TRUSTED attribute shall be supported for all public and secret keys.
Result	The CKA_TRUSTED attribute is not supported for generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any mechanisms that allow wrapping using generic secret keys

Issue ID	ATTRIB-12
PKCS#11 Rule	The CKA_WRAP_TEMPLATE attributes shall be supported for all public and secret keys.
Result	The CKA_WRAP_TEMPLATE attribute is not supported for generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any mechanisms that allow wrapping using generic secret keys.

Issue ID	ATTRIB-13
PKCS#11 Rule	The CKA_UNWRAP_TEMPLATE attributes shall be supported for all private and secret keys.
Result	The CKA_UNWRAP_TEMPLATE attribute is not supported for generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any mechanisms that allow unwrapping using generic secret keys.

Issue ID	ATTRIB-14
PKCS#11 Rule	The CKA_PUBLIC_KEY_INFO attribute shall be supported for all public and private keys.
Result	The CKA_PUBLIC_KEY_INFO attribute is not supported for any key.
Classification	Unsupported Key Attributes
Impact	Not applicable, because the HSM does not support storing X.509 certificate objects that are required to derive the value of the CKM_PUBLIC_KEY_INFO attribute

Issue ID	ATTRIB-15
PKCS#11 Rule	The CKA_DECRYPT attribute shall be supported for all private and secret keys.
Result	The CKA_DECRYPT attribute is not supported for EC private or generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support decryption mechanisms using EC public or generic secret keys.

Issue ID	ATTRIB-16
PKCS#11 Rule	The CKA_SIGN_RECOVER attribute shall be supported for all private keys.

Result	The CKA_SIGN_RECOVER attribute is not supported for EC private keys
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any signature mechanisms with data recovery for EC private keys.

Issue ID	ATTRIB-17
PKCS#11 Rule	The CKA_UNWRAP attribute shall be supported for all private and secret keys.
Result	The CKA_UNWRAP attribute is not supported for EC private or generic secret keys.
Classification	Unsupported Key Attributes
Impact	None, because the HSM does not support any unwrapping mechanisms using EC private keys or generic secret keys.

Document Revisions

Date	SDK Version	Update Description
29-Oct-2020	3.2.1	Initial publication
1-Nov-2020	3.2.1	Updated Introduction with link to description of Galois methodology

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.