Supporting Document

Evaluation Activities for collaborative PP-Module for DBMS Cryptographic Functions

Version 0.5, 2026-07-20

Table of Contents

Chapter 1. Revision History

Table 1. Revision history

Version	Date	Description
0.1	2026-01-25	Initial Evaluation Activities for the DBMS Cryptographic Functions PP-Module.
0.2	2026-01-25	Aligned with the catalogue-consumer model: evaluation by reference to the CC Crypto Catalogue and the Functional Package for TLS.
0.3	2026-06-26	Structural completeness update: per-SFR Evaluation Activities, summary table, test-environment and evidence requirements; aligned Base PP references with cPP_DBMS Version 2.0; and adopted the shared AsciiDoc rendering assets.
0.4	2026-06-30	Added Evaluation Activities for the new mandatory FDP_DIT_EXT.1 Data-in-Transit Protection. Aligned the SD with formal CCDB-018 consumption: Catalogue algorithm Evaluation Activities remain Catalogue-owned, DBMS activities are identified as integration-only, and the Enterprise Enhanced use case adds only an ST selection-conformance check. Added a transitional mechanism for using Certification Body-recognized CAVP, CMVP, or equivalent validation results as evidence for covered algorithm-correctness objectives until the Catalogue Evaluation Methods are available.
0.4	2026-07-07	FDP_DIT_EXT.1 application note generalized to modules claimed alongside this module and aligned with both the Crypto + Cloud and Crypto + DBaaS review families.
0.4	2026-07-08	FDP_DIT_EXT.1 TSS activity aligned with the restructured element: the evaluator verifies the TLS Functional Package claim is present for every conforming TOE regardless of any supplementary protocol, and that supplementary protocols are claimed from an applicable Functional Package and implemented within the TOE boundary.

Version	Date	Description
0.4	2026-07-08	Extended FCS_CKM_EXT.1 activities for the new "Externally Managed" Key Origin: TSS coverage of transient handling, no-plaintext-persistence, by-reference operation allocation, and external-service unavailability behavior; guidance for key-management integration; new Test 2a (no persistent plaintext MEK; loss of the external service denies access to protected data per the TSS). FDP_ITC_EXT.1 section retitled and scoped to both external Key Origin paths.
0.4	2026-07-08	FCS_CKM_EXT.1 TSS activity extended: where FCS_COP.1/SKC is selected for DEK protection, the evaluator examines any compensating integrity mechanisms described in the TSS (not required under General-Purpose).
0.4	2026-07-08	FDP_DAR_EXT.1 TSS activity extended for the rewritten FDP_DAR_EXT.1.2: the evaluator verifies the components selected there are included in the ST and are the operations actually performing the FDP_DAR_EXT.1.1 encryption. Editorial sweep: hardcoded section numbers replaced with section names; FDP_DIT_EXT.1 tests adopt the Step: convention; Test Environment Requirements extended with certificate-presentation and downgrade tooling for the channel tests and external key-management coverage; inline bibliography anchor reworded; Cloud SD reference pinned to Version 0.4.
0.4	2026-07-08	Added Transitional Coverage for FPT_TST.1/RBG and FPT_FLS.1/RBG: pending the Catalogue Evaluation Methods, Certification Body-approved CMVP (FIPS 140-3 / ISO/IEC 19790) module-validation evidence may cover DRBG self-test execution and module error-state behavior under the same acceptance conditions as algorithm validation evidence, recorded in the Cryptographic Validation Coverage Matrix; the TOE-integration residue (RBG-dependent services refused on self-test failure, per the TSS) remains evaluated. Added both iterations to the Evaluation Activity Summary Table, and aligned the FCS_CKM_EXT.1 and FDP_ITC_EXT.1 summary rows with the Externally Managed Key Origin.

Version	Date	Description
0.4	2026-07-18	Aligned with the module's post-quantum additions. The Catalogue-component trigger table notes that pure ML-DSA carries no Hash/XOF dependency and adds a row for the optional FCS_COP.1/KeyEncap (with FCS_RBG.1 and FCS_CKM.6). The transitional CAVP mechanism is stated to apply equally to the anticipated post-Version 1.0 rows (ML-DSA, ML-KEM), both CAVP-validatable under final FIPS PUB 203/204. New ST-verification item for the optional KeyEncap component: RBG and key-destruction inclusion, identification of the DBMS function the KEM serves, and the protocol-KEM exclusion.
0.4	2026-07-18	Aligned with the module's conversion of FCS_COP.1/KeyEncap to Selection-Based via FDP_ITC_EXT.1.4. Trigger table and ST-verification item now key off the FDP_ITC_EXT.1.4 selection and cover both the KeyWrap and KeyEncap supplementary-protection options. FDP_ITC_EXT.1 TSS activities gain the .4 coverage: which flows are wrapped or encapsulated, under which component, key establishment and protection for the mechanism, and the location of unwrapping or decapsulation.
0.4	2026-07-20	Aligned the ML-KEM evaluation model with role-specific KeyGen, encapsulation, and decapsulation. Added FCS_CKM.1/KEM activities for TOE-performed KeyGen; made RBG evidence conditional on KeyGen or encapsulation; added activities and tests for the "Established through Key Encapsulation" Master Key origin; corrected FDP_ITC_EXT.1.4 coverage to evaluate a KEM-derived KEK used with KeyWrap or AEAD; and added boundary/allocation checks for externally performed KMIP or KMS operations. CAVP evidence now maps KeyGen separately from EncapDecap.
0.5	2026-07-20	Added TSS, guidance, and test coverage for the explicit FCS_CKM_EXT.1.2 Master Key protection models, including cryptographic protection, non-exportable hardware-backed storage, and transient-only handling. Aligned evaluation evidence with the module's normative audit-event refinement and rolled the coordinated review set to Version 0.5.

Chapter 2. Introduction

2.1. Technology Area and Scope of Supporting Document

This Supporting Document (SD) defines Evaluation Activities (EAs) for the DBMS-specific SFRs and integration behavior in the **collaborative PP-Module for DBMS Cryptographic Functions** (DBMS_MOD_CRYPTO), version 0.5. The technology area addressed is Database Management Systems (DBMS) that implement cryptographic protections for data at rest, data in transit, and cryptographic key management.

This SD provides Evaluation Activities that evaluators shall perform to determine whether a TOE satisfies the DBMS-specific SFRs and integration requirements specified in the DBMS Cryptographic Functions PP-Module. It does not originate or reproduce algorithm-level Evaluation Activities for CCDB-018 components. CCDB-018 assigns those activities to its companion evaluation-methods document. Until that document is available, Section 3.1.2, “Use of Scheme-Recognized Cryptographic Validation Results” permits Certification Body-recognized cryptographic validation results to be used as evidence for the algorithm-correctness objectives they demonstrably cover; it does not define replacement algorithm tests. This SD also complements the Supporting Document for the collaborative Protection Profile for Database Management Systems (cPP_DBMS SD), which remains applicable for all SFRs inherited from the Base PP.

The Evaluation Activities in this SD are derived from the Common Evaluation Methodology (CEM:2022) work units and are aligned with the framework defined in CC:2022 Part 4 for specifying objective, repeatable, and reproducible evaluation methods.

2.2. Structure of the Document

This Supporting Document is organized to mirror the structure of the PP-Module and follows the SFR class hierarchy defined in CC:2022 Part 2.

The **General Guidance for Evaluators** section provides overarching guidance applicable across multiple SFRs, including:

- Relationship between this module and the Crypto Catalogue
- Handling selections and assignments in the ST
- Verification of consumed Catalogue components
- Evaluator coordination with the Crypto Catalogue evaluation activities

The **Assurance Activities Overview** section describes:

- Mapping to Base PP SARs
- Documentation requirements

Subsequent sections define Evaluation Activities organized by SFR class:

- FCS: DBMS integration of consumed Catalogue components and DBMS-specific key management
- FDP: User Data Protection

Within each class section, Evaluation Activities are specified for each SFR, organized into:

- **TSS Activities**: Verification of the TOE Summary Specification
- **Guidance Activities**: Verification of operational and preparative guidance
- **Test Activities**: Independent functional testing procedures

2.3. Terminology

The following terms are used throughout this Supporting Document:

Evaluation Activity (EA)

An activity derived from one or more CEM work units that specifies the actions evaluators shall perform to determine whether the TOE meets a specific SFR or SAR. Evaluation Activities are objective, repeatable, and produce determinate results.

TOE Summary Specification (TSS)

The section of the Security Target that describes how the TOE Security Functionality (TSF) meets each Security Functional Requirement. Evaluators verify that the TSS provides sufficient detail to understand the security mechanisms implemented.

Guidance Documentation

The set of documents provided by the developer that describe how to securely install, configure, and operate the TOE. This includes operational guides, preparative procedures, and configuration references for cryptographic key management.

Test

A procedure performed by the evaluator to verify that the TSF behaves as specified. Tests include test objectives, test setup, test procedures, and expected results.

Selection

An operation performed by the ST author to choose one or more items from a list provided in an SFR component. The evaluator verifies that selections are valid and that the TSS describes how the selected functionality is implemented.

Assignment

An operation performed by the ST author to specify a value or set of values for a parameter in an SFR component. The evaluator verifies that assignments are appropriate and that the TSS describes how the assigned values are implemented.

Crypto Catalogue

The **Specification of Functional Requirements for Cryptography** (CCDB-018) [[Crypto_Catalog]], the Common Criteria standard defining SFRs for cryptographic algorithm implementations. The DBMS Cryptographic Functions PP-Module consumes and reproduces the applicable Catalogue components while retaining their identifiers, dependencies, ECD

provenance, and algorithm-level evaluation ownership.

DEK (Data Encryption Key)

A symmetric cryptographic key used to encrypt and decrypt database user data. DEKs are generated or derived by the TOE and stored only in protected form under the Master Encryption Key (MEK).

MEK (Master Encryption Key)

The root cryptographic key in the DBMS key hierarchy. The MEK wraps DEKs. MEK lifecycle is governed by `FCS_CKM_EXT.1`.

BYOK (Bring Your Own Key)

A key management model in which the customer supplies their own MEK to the TOE. BYOK is implemented via `FCS_CKM_EXT.1` (import selection) and `FDP_ITC_EXT.1` (trusted import channel).

2.4. Relationship to Other Documents

This Supporting Document must be used in conjunction with the following documents.

2.4.1. Primary PP-Module

- **collaborative PP-Module for DBMS Cryptographic Functions (DBMS_MOD_CRYPTO)**, version 0.5

  This SD provides Evaluation Activities for the SFRs defined in the PP-Module. Evaluators shall verify conformance to all mandatory SFRs and any selection-based SFRs included in the Security Target.

2.4.2. Base PP and Supporting Document

- **collaborative Protection Profile for Database Management Systems (cPP_DBMS)**

  The Base PP, Version 2.0 dated 27 April 2026, must be claimed in conjunction with this PP-Module. All SFRs and SARs from the cPP_DBMS apply to TOEs claiming conformance to the DBMS Cryptographic Functions PP-Module.

- **Supporting Document for cPP_DBMS (cPP_DBMS SD)**

  The Base PP SD provides Evaluation Activities for SFRs inherited from the cPP_DBMS. When evaluating a TOE claiming conformance to the DBMS Cryptographic Functions PP-Module, evaluators shall apply both this SD and the cPP_DBMS SD. Where SFRs are refined or iterated by the PP-Module, the Evaluation Activities in this SD take precedence or supplement those in the Base PP SD.

2.4.3. Crypto Catalogue

- **Specification of Functional Requirements for Cryptography (CCDB-018)**, Version 1.0, January 2025 [[Crypto_Catalog]]

The PP-Module consumes the Crypto Catalogue for standard cryptographic algorithm SFRs (AES, SHA, DRBG, KDF, etc.) and reproduces the components used by the module. CCDB-018 states that algorithm Evaluation Activities will be specified in the companion Evaluation Methods for Cryptographic Security Functional Requirements [[Crypto_Eval_Methods]] and that Requirements Documents may reference those methods rather than reproduce them. At publication of this SD, no publicly available, scheme-recognized version of that companion document has been identified. Evaluators shall apply it when available. Until then, evaluators shall use Section 3.1.2, "Use of Scheme-Recognized Cryptographic Validation Results" and any additional method approved by the responsible Certification Body. This SD does **not** duplicate or originate Catalogue-level algorithm testing.

- **Evaluation Methods for Cryptographic Security Functional Requirements** [[Crypto_Eval_Methods]]

  CCDB-018 identifies this as the companion evaluation-methods document for cryptographic SFR components. Evaluators shall use the version recognized by the responsible Certification Body when available. Before then, a Catalogue-derived claim is complete only when each algorithm-correctness objective is covered by acceptable validation evidence under Section 3.1.2, "Use of Scheme-Recognized Cryptographic Validation Results" or by another specifically identified method approved by that Certification Body. The DBMS-specific integration checks in this SD remain mandatory in either case.

- **Functional Package for Transport Layer Security (TLS)**, Version 2.1 [[TLS_FP]]

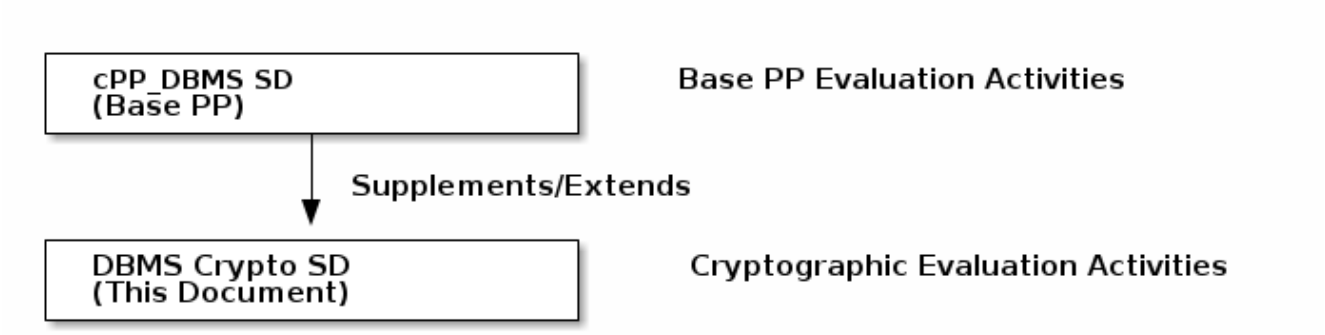
  For TLS channel requirements invoked by `FDP_ITC_EXT.1`, evaluators shall apply the Evaluation Activities defined in the TLS Functional Package for the claimed `FCS_TLSC_EXT.1` or `FCS_TLSS_EXT.1` components. This SD covers only the DBMS-specific key import channel behaviour.

2.4.4. CC Standards

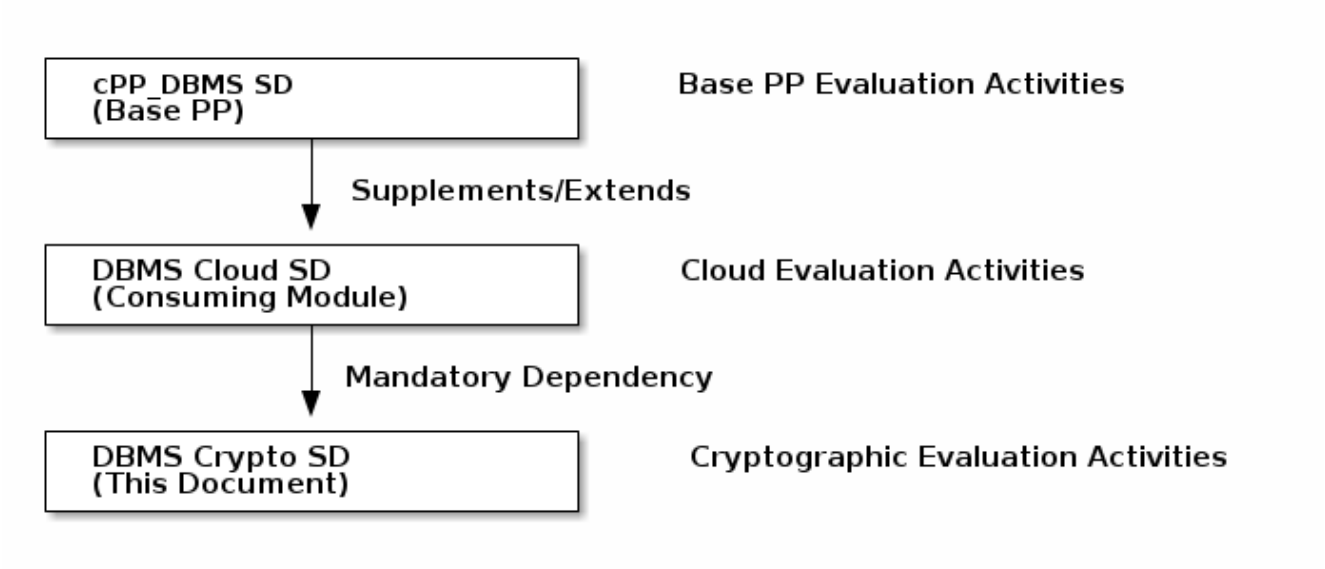
- **Common Criteria for Information Technology Security Evaluation, CC:2022** (Parts 1-5)
- **Common Methodology for Information Technology Security Evaluation, CEM:2022**

2.4.5. Layering of Supporting Documents

Supporting Documents layer in the same manner as PP-Modules:



When this module is used as a mandatory dependency of the DBMS Cloud Module:



Evaluators of the DBMS Cloud Module shall apply all three SDs. This SD is authoritative for the cryptographic requirements used by the Cloud PP-Configuration, including Data-in-Transit testing for TLS, including HTTPS over TLS where applicable, and Data-at-Rest testing for the selected `FDP_DAR_EXT.1` strategy.

Chapter 3. General Guidance for Evaluators

This section provides guidance applicable across multiple Evaluation Activities in this Supporting Document. Evaluators shall consider this guidance when performing any evaluation of a TOE claiming conformance to the DBMS Cryptographic Functions PP-Module.

3.1. Relationship to the Crypto Catalogue

3.1.1. What Is and Is Not Evaluated by This SD

The PP-Module invokes the **Specification of Functional Requirements for Cryptography** (CCDB-018) [[Crypto_Catalog]] for all standard cryptographic algorithm implementations. This SD does **not** define Evaluation Activities for:

- Algorithm correctness (e.g., Known Answer Tests, DRBG health tests)
- Signature generation and verification algorithm tests
- Hash function correctness tests
- Key derivation function algorithm tests, except for the DBMS-specific integration checks defined for derived DEKs

The evaluator shall apply the evaluation methods associated with all Catalogue components claimed in the ST, including [[Crypto_Eval_Methods]] when a scheme-recognized version is available. Before that document is available, the evaluator shall account for every claimed Catalogue operation using Section 3.1.2, “Use of Scheme-Recognized Cryptographic Validation Results” or another method specifically approved by the responsible Certification Body. This SD provides Evaluation Activities **only** for the DBMS-specific extensions and integration behaviours defined in the PP-Module:

- Key hierarchy management (DEK/MEK), including DBMS-specific use of generated or derived DEKs
- Master Key import (BYOK) via a trusted channel when selected
- Data-at-rest encryption strategy (Storage-Scope/TDE or Granular)

3.1.2. Use of Scheme-Recognized Cryptographic Validation Results

Where the responsible Certification Body recognizes external cryptographic validation results, the evaluator may use them as evidence for algorithm-correctness objectives they demonstrably cover. Acceptable evidence includes an official NIST CAVP validation record [[NIST_CAVP]], a CMVP validation that identifies the applicable CAVP validations, or another equivalent result explicitly accepted by the Certification Body. This evidence-reuse rule does not create an alternative SFR, conformance claim, use case, or cryptographic algorithm.

NIAP Policy #5 [[NIAP_CAVP_POLICY]] permits CAVP or CMVP results as evidence for covered PP/cPP assurance activities. The Canadian Common Criteria Program requires CAVP validation for cryptography claimed in PP-conformant evaluations while separately requiring implementation-presence verification and applicable entropy assessment [[CAN_CC_INSTRUCTIONS]]. These national policies do not make CAVP mandatory for evaluations conducted by other Certification Bodies.

The evaluator shall produce a **Cryptographic Validation Coverage Matrix** containing, for every Catalogue-derived operation in the ST:

- The Catalogue component, iteration, selected algorithm, mode, direction, key size, parameter set, and applicable standard.
- The cryptographic implementation name, vendor, version, build or binary identifier, implementation boundary, and whether the implementation is software, firmware, hardware, or a combination.
- The validation program, official validation record identifier and status, validated capabilities, and supporting primitive validations.
- The validated operating environment, including processor or hardware identifier, operating system, virtualization or container layer, and relevant instruction-set extensions.
- The TOE configuration and operating environment to which the validation is being applied.
- The algorithm-correctness objectives satisfied by the validation and every residual objective requiring another evaluation method.

The evaluator shall verify all of the following before accepting validation evidence:

1. The record appears in the official validation listing and is acceptable to the responsible Certification Body at the time of evaluation. Results from demonstration or self-test systems are not validation evidence.
2. The validation covers every algorithm, mode, direction, key size, parameter, state, capability, and prerequisite primitive used to satisfy the mapped ST operation.
3. The validated implementation is the implementation included in or invoked by the evaluated TOE and has not been modified in a way that invalidates the validation.
4. The validated operating environment matches the evaluated configuration, or the Certification Body has approved and the evaluator has documented a specific equivalency rationale.
5. The TSS identifies the implementation and boundary, and the evaluator records the coverage determination and supporting evidence in the evaluation report.

A CAVP or equivalent validation result may replace only algorithm-correctness testing within its demonstrated scope. It does **not**, by itself, satisfy:

- Verification that the validated implementation is present, configured, and invoked by the TOE.
- Entropy-source assessment, entropy delivery, DRBG instantiation or reseeding integration, or health-test handling beyond the validated algorithm behavior.
- DBMS key generation, derivation input binding, import, storage, rotation, destruction, zeroization consequences, or other key-lifecycle integration activities.
- One-time-value generation, uniqueness, unpredictability, persistence, or reuse prevention outside the behavior directly covered by the validation.
- Protocol, trusted-channel, certificate-validation, data-at-rest coverage, guidance, TOE-boundary, or configuration activities.
- Any Catalogue selection or Evaluation Activity element that is not mapped to a covered capability in the validation record.

For every uncovered element, the evaluator shall apply a specifically identified method accepted by the responsible Certification Body. Merely stating that a CAVP record exists, or that the Catalogue Evaluation Methods are not yet available, is not sufficient.

The transitional mechanism applies equally to the PP-Module's anticipated post-Version 1.0 rows — ML-DSA in `FCS_COP.1/SigVer`, ML-KEM KeyGen in `FCS_CKM.1/KEM`, and ML-KEM encapsulation or decapsulation in `FCS_COP.1/KeyEncap`. FIPS PUB 203 and FIPS PUB 204 are final and CAVP validates implementations of both. The validation coverage matrix shall map the ML-KEM KeyGen capability separately from EncapDecap and shall identify which of encapsulation and decapsulation the TOE performs. Algorithm correctness for those selections is otherwise evidenced in the same way as for published Catalogue rows, pending the corresponding Catalogue entries and Evaluation Methods.

When a scheme-recognized version of [[Crypto_Eval_Methods]] becomes available, that document becomes the primary algorithm evaluation method for this SD. Existing validation results may continue to satisfy individual activities only to the extent permitted by that document or by the responsible Certification Body. The DBMS-specific integration activities in this SD remain applicable and are not displaced by publication of the Catalogue methods.

3.1.3. Transitional Coverage for FPT_TST.1/RBG and FPT_FLS.1/RBG

`FPT_TST.1/RBG` and `FPT_FLS.1/RBG` are CC Part 2 iterations completed for the `FCS_RBG.1` dependency. The Catalogue delegates RBG health testing to FPT_TST.1 claimed alongside the FCS_RBG components, and the evaluation methods for that behavior are expected from the Catalogue Evaluation Methods [[Crypto_Eval_Methods]]. Until a scheme-recognized version of that document is available, the evaluator shall proceed as follows, subject to Certification Body approval:

1. The evaluator shall verify the TSS identifies the DRBG self-tests performed and when they run (per the FPT_TST.1.1/RBG selection), and the failure behavior by which the TOE preserves a secure state per FPT_FLS.1.1/RBG.
2. Where the DRBG implementation is within the boundary of a cryptographic module validated under the CMVP (FIPS 140-3 / ISO/IEC 19790) — whose validation examines pre-operational and conditional self-tests, including DRBG health tests, and the module's error states — the

evaluator may accept that module validation as evidence for the self-test execution and failure behavior of the validated module, applying the same acceptance conditions as for algorithm validation evidence above (official listing status, implementation identity, operating-environment match, and TSS boundary identification) and recording the determination in the Cryptographic Validation Coverage Matrix.

3. The TOE-integration residue remains evaluated: the evaluator shall verify, per the TSS, that the TOE refuses RBG-dependent cryptographic services when the module reports a self-test failure or error state. Where inducing the failure is not practical, the evaluator shall use a method accepted by the responsible Certification Body and record it in the evaluation report.

This transitional coverage does not originate evaluation methodology for the RBG health tests themselves; it reuses the self-test examination already performed in the module validation, pending the Catalogue Evaluation Methods.

3.1.4. Verification of Consumed Catalogue Components

Before conducting SFR-specific integration activities, the evaluator shall verify that the ST includes every Catalogue-derived component triggered by the TOE implementation, as specified in the Consumed Cryptographic Catalogue Components appendix of the PP-Module [[DBMS_MOD_CRYPTO]].

The evaluator shall examine the ST to confirm it includes, at minimum, the Catalogue components applicable to the ST selections:

Purpose	Included Component	CCDB-018 Section
Symmetric-key data encryption (DEK)	FCS_COP.1/SKC and/or FCS_COP.1/AEAD	Section 4.10 and/or Section 4.2
Random Bit Generation (when direct symmetric key generation, ML-KEM KeyGen, or ML-KEM encapsulation is claimed)	FCS_RBG.1	Section 6.2
DEK derivation from non-password inputs (when used)	FCS_CKM.5	Section 3.6
Password-based DEK derivation (when used)	FCS_CKM_EXT.8	Section 3.9 and Annex A.2.3
Hashing (integrity, key wrap verification)	FCS_COP.1/Hash	Section 4.4
Signature verification for an in-scope DBMS function, when used	FCS_COP.1/SigVer plus FCS_COP.1/Hash or FCS_COP.1/XOF (neither for pure ML-DSA)	Section 4.8 and supporting primitive section; ML-DSA row anticipates post-Version 1.0 publication

Purpose	Included Component	CCDB-018 Section
Non-protocol ML-KEM key establishment for a Master Key origin or supplementary end-to-end protection, when selected	FCS_COP.1/KeyEncap for the TOE role and FCS_CKM.5; FCS_CKM.1/KEM and FCS_RBG.1 for TOE KeyGen; FCS_RBG.1 for encapsulation; FCS_COP.1/KeyWrap or FCS_COP.1/AEAD for a transferred key; FCS_CKM.6 for secret values	ML-KEM components anticipate post-Version 1.0 publication; key derivation and transferred-key protection use the applicable published Catalogue components
Key Destruction (MEK/DEK lifecycle)	FCS_CKM.6	Section 3.7

The evaluator shall additionally verify:

1. When FCS_CKM_EXT.1.1 selects "Generated Internally" for MEK origin, the ST includes FCS_CKM.1/SKG, FCS_RBG.1, the selected seeding component, and the required RBG failure/self-test components.
2. When the TOE directly generates DEKs, the ST includes the same key-generation and RBG component set for DEK generation.
3. When FCS_CKM_EXT.1.1 selects "Imported from External Entity", "Externally Managed", or "Established through Key Encapsulation", the ST includes mutual TLS channel components from the Functional Package for TLS v2.1 (FCS_TLSC_EXT.1 or FCS_TLSS_EXT.1, as applicable to the TOE role).
4. When FCS_CKM_EXT.1.1 selects "Established through Key Encapsulation", the ST includes FCS_COP.1/KeyEncap with the TOE's operations selected, FCS_CKM.5 for MEK derivation, and the required supporting primitives. It includes FCS_CKM.1/KEM and FCS_RBG.1 when the TOE performs KeyGen, and FCS_RBG.1 when the TOE performs encapsulation.
5. When FCS_CKM_EXT.1.2 selects key wrapping for DEK protection, the ST includes FCS_COP.1/KeyWrap.
6. When FCS_CKM_EXT.1.2 selects symmetric key encryption or authenticated encryption for DEK protection, the ST includes FCS_COP.1/SKC or FCS_COP.1/AEAD, as applicable.
7. When an included operation requires IVs, nonces, tweaks, or other one-time values, the ST includes FCS_OTV_EXT.1.
8. When FDP_ITC_EXT.1.4 selects direct supplementary key wrapping, the ST includes FCS_COP.1/KeyWrap. When it selects KEM-based supplementary protection, the ST includes FCS_COP.1/KeyEncap for the TOE role, FCS_CKM.5 for KEK derivation, and FCS_COP.1/KeyWrap or FCS_COP.1/AEAD for protection of the transferred key. It includes FCS_CKM.1/KEM and FCS_RBG.1 when the TOE performs KeyGen, FCS_RBG.1 when the TOE performs encapsulation, and lists all secret ML-KEM values under FCS_CKM.6. Protocol KEMs are claimed through the applicable Functional Package instead.
9. When certificate-based mutual TLS authentication requires signature generation or verification support, the ST includes the applicable asymmetric components required by the TLS Functional Package.

10. All Catalogue-derived components include the required assignments and selections for key sizes, algorithm parameters, standards references, and dependencies.

The evaluator shall also verify that the guidance documents identify which Catalogue-defined algorithms are used for each cryptographic operation, and that the TOE's behaviour observed during integration testing is consistent with the included Catalogue components.

3.1.5. Use Case Selection Verification

The evaluator shall verify that the ST identifies `[USE CASE 1] General-Purpose Cryptographic Deployment`, `[USE CASE 2] Enterprise Enhanced`, or both, as applicable.

When the ST claims Enterprise Enhanced, the evaluator shall compare every applicable Catalogue operation in the ST against the Enterprise Enhanced Use Case Selection Template in the PP-Module and verify that no disallowed algorithm, mode, key size, hash, KDF primitive, DRBG primitive, or `never` reseeding choice is used to satisfy the use case. This is a conformance check on the selected Catalogue operations, not a new algorithm Evaluation Activity.

If the ST makes a FIPS 140-3 [[FIPS_140_3]] validation claim, the evaluator shall verify that the claim separately identifies the module certificate, version, cryptographic boundary, and approved mode. Absence of such a validation claim does not change the Catalogue Evaluation Activities, and selection of Enterprise Enhanced alone shall not be reported as FIPS validation, NIAP approval, CNSA 2.0 [[CNSA_2]] conformance for the complete product, or product-list eligibility.

3.2. Handling Selections and Assignments

3.2.1. Evaluating SFRs with Selections

When an SFR includes a selection operation, the evaluator shall:

1. Verify that the ST author selected one or more valid options from the selection list.
2. Verify that selected options are consistent with the TOE's actual capabilities.
3. Verify that the TSS describes how the selected functionality is implemented.
4. Verify that guidance documentation describes how to configure and use the selected functionality.
5. Perform tests that exercise the selected functionality.

Key selections in this PP-Module and their evaluation implications:

- `FCS_CKM.1/SKG`: Direct symmetric key generation includes the consumed `FCS_RBG.1` component when the TOE directly generates DEKs or internally generates MEKs. DEK derivation triggers the consumed `FCS_CKM.5` or `FCS_CKM_EXT.8` component and the supplementary DBMS-specific derivation activities in this SD.
- `FCS_CKM.1/KEM`: TOE-performed ML-KEM.KeyGen includes `FCS_RBG.1` and is evaluated separately from encapsulation and decapsulation. A provisioned or externally generated key pair does not trigger this component.

- FCS_COP.1/KeyEncap: Encapsulation, decapsulation, or both are selected according to the TOE role. Encapsulation triggers FCS_RBG.1; decapsulation alone does not. Protocol-level KEM use is outside this component.
- FCS_CKM_EXT.1.1: The selected internal, imported, externally managed, or KEM-established origin determines the applicable MEK tests. A KEM-established origin also triggers MEK derivation and the role-specific ML-KEM activities.
- FCS_CKM_EXT.1.2: Key wrapping vs. symmetric key encryption vs. authenticated encryption for stored DEK protection - determines whether the ST includes FCS_COP.1/KeyWrap, FCS_COP.1/SKC, or FCS_COP.1/AEAD, and whether FCS_OTV_EXT.1 is required.
- FDP_DAR_EXT.1.1: "Storage-Scope Encryption" vs. "Granular Data Encryption" - determines which storage inspection tests apply. Both selections require testing for the selected path.

3.2.2. Evaluating SFRs with Assignments

When an SFR includes an assignment operation, the evaluator shall:

1. Verify that the ST author provided appropriate values for the assignment.
2. Verify that assigned values are consistent with the TOE's capabilities and the security problem definition.
3. Verify that the TSS describes how the assigned values are implemented.
4. Verify that assigned values are sufficiently specific to enable deterministic testing.

3.2.3. Verifying ST Author Made Valid Selections

The evaluator shall verify that selections are:

- **Valid**: The selected option appears in the SFR selection list.
- **Complete**: Required selections are made (no unresolved selections remain).
- **Consistent**: Selections across SFRs are mutually compatible.
- **Accurate**: Selected functionality is actually implemented by the TOE.

Chapter 4. Assurance Activities Overview

4.1. Mapping to Base PP SARs

The DBMS Cryptographic Functions PP-Module inherits all Security Assurance Requirements (SARs) from the collaborative Protection Profile for Database Management Systems (cPP_DBMS). No additional or modified SARs are introduced by this PP-Module.

4.1.1. Inherited SAR Families

The following SAR families from the cPP_DBMS apply unchanged:

SAR Family	Components	Primary SD Coverage
ADV (Development)	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1	cPP_DBMS SD
AGD (Guidance Documents)	AGD_OPE.1, AGD_PRE.1	cPP_DBMS SD + Crypto-specific Guidance Activities in this SD
ALC (Life-cycle Support)	ALC_CMC.2, ALC_CMS.2, ALC_DEL.1, ALC_FLR.3	cPP_DBMS SD
ASE (Security Target Evaluation)	ASE_CCL.1, ASE_ECD.1, ASE_INT.1, ASE_OBJ.2, ASE_REQ.2, ASE_SPD.1, ASE_TSS.1	cPP_DBMS SD + PP-Module conformance verification in this SD
ATE (Tests)	ATE_COV.1, ATE_FUN.1, ATE_IND.2	cPP_DBMS SD + Crypto-specific Test Activities in this SD
AVA (Vulnerability Assessment)	AVA_VAN.2	cPP_DBMS SD

4.1.2. Applying Base PP SAR Activities

For each SAR inherited from the cPP_DBMS, the evaluator shall:

1. Perform all Evaluation Activities specified in the cPP_DBMS SD.
2. Apply crypto-specific activities from this SD where applicable.
3. Document findings in the Evaluation Technical Report (ETR).

4.2. Documentation Requirements

4.2.1. AGD_OPE: Operational User Guidance — Cryptographic Requirements

The operational user guidance shall include sections addressing:

Key Management Guide:

- Supported key hierarchy model (DEK/MEK)
- MEK generation or import procedures
- DEK lifecycle management (creation, rotation, destruction)
- Keystore location, format, and access control requirements

Encryption Strategy Configuration:

- Enabling and verifying Storage-Scope or Granular encryption
- Scope of encryption (data files, logs, temp space, backups, exports, and documented exclusions)
- Column encryption mode selection (deterministic vs. randomized)

Trusted Channel Configuration (BYOK):

- Certificate management for the KMS connection
- Supported external KMS types
- Troubleshooting key import failures

4.2.2. AGD_PRE: Preparative Procedures — Cryptographic Deployment

The preparative procedures shall include:

Cryptographic Environment Preparation:

- Prerequisites for keystore initialization
- Required entropy sources and RNG configuration
- File system permissions for key storage

Deployment Verification:

- Verifying encryption is active after deployment
- Testing key import before production use
- Baseline cryptographic configuration validation

The evaluator shall verify that these guidance elements are present and accurate by applying the documented procedures during evaluation.

Chapter 5. FCS Class: DBMS Integration Activities

5.1. FCS_CKM.1/SKG Cryptographic Key Generation (TOE-Generated Symmetric Keys) (Selection-Based)

This consumed SFR applies when the TOE directly generates DEKs or internally generates MEKs. Random bit generation and algorithm correctness are evaluated using the Catalogue methods. The activities below cover only DBMS-specific use of directly generated keys within the key hierarchy and their protection under the Master Encryption Key (MEK). Internally generated MEKs are evaluated through the FCS_CKM_EXT.1 integration activities. When the TOE derives DEKs, the consumed FCS_CKM.5 or FCS_CKM_EXT.8 component is evaluated using the Catalogue methods and the DBMS integration activities in Section 5.3, “FCS_CKM.5 / FCS_CKM_EXT.8 Key Derivation (Consumed Catalogue Components)”.

5.1.1. FCS_CKM.1/SKG Evaluation Activities

5.1.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- If the TOE directly generates DEKs, the mechanism for directly generating Data Encryption Keys (DEKs), consistent with FCS_CKM.1.1/SKG and the claimed FCS_RBG.1 component.
- If the TOE directly generates DEKs, the supported DEK key sizes in bits, consistent with the assignment in FCS_CKM.1.1/SKG and the claimed FCS_COP.1/SKC or FCS_COP.1/AEAD parameters.
- If the TOE directly generates DEKs, the key hierarchy: how generated DEKs are immediately protected (wrapped or encrypted) by the Master Encryption Key (MEK). The TSS shall explicitly state that DEKs are never stored in plaintext on persistent storage.
- If the TOE directly generates DEKs, when DEKs are generated (e.g., at tablespace creation, at column encryption enablement) and the conditions under which they may be regenerated or rotated.
- The source of randomness used for direct key generation, identifying FCS_RBG.1 and its selected seeding component.
- If the TOE derives DEKs instead of, or in addition to, directly generating DEKs, the applicable FCS_CKM.5 or FCS_CKM_EXT.8 component and the derivation path evaluated under Section 5.3, “FCS_CKM.5 / FCS_CKM_EXT.8 Key Derivation (Consumed Catalogue Components)”.

Application Note: A derived-only DEK implementation does not perform the DEK generation tests in this section. The evaluator instead applies the FCS_CKM.5 or FCS_CKM_EXT.8 activities for the derivation path and verifies any internally generated MEK under FCS_CKM_EXT.1. If the TSS states that key generation is performed by the platform or by an external component rather than the TOE itself, the evaluator shall verify this claim is consistent with the TOE boundary as described in the ST and that any platform-provided key generation meets the Catalogue requirements through appropriate environmental assumptions.

5.1.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- If the TOE directly generates DEKs, any administrator-visible steps associated with DEK generation, such as enabling encryption on a tablespace, database, or column set.
- If the TOE directly generates DEKs, how to verify DEK generation has succeeded (e.g., via system views, status commands, or log entries).
- Any configuration parameters that affect claimed direct key generation behaviour (e.g., key size selection, algorithm mode selection where multiple options are supported).
- A statement that DEKs are managed automatically by the TOE and that administrators should not attempt to access or export DEK material directly.

5.1.1.3. Test Activities

Test 1: DEK Generation and Protection Verification

Applicable when the TOE directly generates DEKs.

1. Step: Configure the TOE with a known Master Encryption Key (MEK_1) per the administrator guidance.
2. Step: Enable database encryption on a new tablespace, database, or column set, triggering DEK generation.
3. Step: Stop the DBMS service to flush memory caches.
4. Step: Using a binary editor or appropriate inspection tool, examine the persistent storage location where the key store or encrypted data resides.
5. Step: If the test procedure provides known DEK material or a controlled test key, search for that known plaintext representation in the persistent storage location. If the TOE does not expose or allow controlled DEK material, inspect the key store metadata, key export interfaces, and storage artifacts to verify that DEKs are present only as protected key records and are not exportable in plaintext.

Expected result: The DEK is present only in protected form in persistent storage. No plaintext DEK material is discoverable, and the TOE provides no administrative or user interface that exports plaintext DEKs outside the TOE boundary.

Test 2: DEK Key Size Consistency

Applicable when the TOE directly generates DEKs.

1. Step: Examine system views, status commands, or audit logs that report the encryption algorithm and key size for active encrypted objects.
2. Step: Verify the reported key size matches the assignment made in `FCS_CKM.1.1/SKG` and the corresponding `FCS_COP.1/SKC` or `FCS_COP.1/AEAD` claims.

Expected result: All active DEKs use the key size(s) specified in the ST. No objects use key sizes not claimed in the ST.

5.2. FCS_CKM.1/KEM Cryptographic Key Generation - ML-KEM Key Pair (Selection-Based)

This anticipated component applies only when the TOE performs ML-KEM.KeyGen. Provisioning or generation by an external KMS does not trigger it.

5.2.1. FCS_CKM.1/KEM Evaluation Activities

5.2.1.1. TSS Activities

The evaluator shall verify that the TSS identifies:

- The ML-KEM parameter set and the TOE component performing KeyGen.
- The randomness source and corresponding `FCS_RBG.1` claim.
- The representations and storage locations of the encapsulation key, decapsulation key, and any retained KeyGen seed.
- How the decapsulation key and KeyGen seed are protected and destroyed under `FCS_CKM.6`.
- The DBMS function and `FCS_COP.1/KeyEncap` role for which the key pair is generated.

The evaluator shall verify that the Cryptographic Validation Coverage Matrix maps ML-KEM KeyGen evidence separately from EncapDecap evidence and covers the selected parameter set.

5.2.1.2. Guidance Activities

The evaluator shall verify that the guidance describes how to generate, activate, rotate, back up where permitted, restore where permitted, and destroy the ML-KEM key pair, and how an administrator confirms the active parameter set and key identifier without exposing secret key material.

5.2.1.3. Test Activities

Test 1: ML-KEM Key-Pair Generation and Protection

1. Step: Generate an ML-KEM key pair using the documented TOE procedure.
2. Step: Verify that the reported parameter set and key identifier match the ST and TSS.
3. Step: Exercise the claimed encapsulation or decapsulation function using the generated key pair and verify successful shared-secret establishment with the peer test implementation.
4. Step: Inspect persistent key records and administrator-visible outputs and verify that the decapsulation key and any retained KeyGen seed are not exposed in plaintext.
5. Step: Destroy the key pair, then verify that the TOE no longer permits the claimed operation with that key identifier.

Expected result: The TOE generates the claimed ML-KEM key pair using the claimed RBG, the pair supports the selected operation, secret representations are protected, and destruction prevents

subsequent use.

5.3. FCS_CKM.5 / FCS_CKM_EXT.8 Key Derivation (Consumed Catalogue Components)

These activities apply when the TOE derives DEKs instead of, or in addition to, directly generating DEKs, or when the TOE uses FCS_CKM.5 to derive an MEK or key-encryption key (KEK) from an ML-KEM shared secret. Algorithm-level KDF correctness, approved parameters, and primitive testing are evaluated using the Catalogue component evaluation methods for FCS_CKM.5 or FCS_CKM_EXT.8. This SD verifies the DBMS-specific integration of the derived keys into the database encryption hierarchy.

5.3.1. FCS_CKM.5 / FCS_CKM_EXT.8 Evaluation Activities

5.3.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- Whether DEKs are derived from existing keying material or other non-password derivation inputs (FCS_CKM.5), from a password/passphrase (FCS_CKM_EXT.8), or whether an ML-KEM shared secret is used as the input to FCS_CKM.5 to derive an MEK or KEK.
- The derivation inputs used by the DBMS, including salts, labels, context strings, key identifiers, tenant/database identifiers, counters, iteration counts, memory or work factors, and any other values required by the included Catalogue component. For ML-KEM input, the description shall identify the context that binds the derived key to the TOE and peer identities, the selected key purpose, and the applicable establishment transaction or protected-key record.
- The supporting Catalogue primitive components used by the derivation function (for example, hash, keyed hash, CMAC, SKC, or AEAD components), and how those components match the ST assignments.
- The output key sizes and how derived DEKs are bound to the encrypted database object, tenant, table, column, or storage scope, or how an ML-KEM-derived MEK or KEK is bound to its key-establishment or key-transfer purpose.
- How derived keys are protected after derivation and whether any derived key material is cached, persisted, rotated, or destroyed. Where ML-KEM input is used, the TSS shall explain when the raw shared secret is destroyed and demonstrate that it is not used directly as an MEK, KEK, or data-encryption key.

5.3.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- Any administrator-visible configuration that affects key derivation, including password/passphrase policy when FCS_CKM_EXT.8 is claimed and ML-KEM derivation context or key-purpose configuration when FCS_CKM.5 consumes an ML-KEM shared secret.

- How derivation parameters are generated, stored, backed up, restored, and migrated during upgrade or rotation.
- How administrators verify that derived-key encryption is enabled for the intended database objects.

5.3.1.3. Test Activities

Test 1: Non-Password DEK Derivation Parameter Binding

Applicable when `FCS_CKM.5` is claimed for DEK derivation.

1. Step: Configure two encrypted database objects, tenants, or storage scopes that use distinct derivation context values according to the TSS.
2. Step: Cause the TOE to derive DEKs for both objects and record observable key identifiers, protected key records, metadata, or ciphertext records.
3. Step: Verify that the recorded protected key records or ciphertext records differ when the derivation context differs.
4. Step: Verify that the derivation context values and output key sizes are consistent with the ST assignments and TSS.

Expected result: Distinct derivation contexts produce distinct protected key records or ciphertext effects consistent with the claimed derivation design. The TOE records or derives keys using the parameters described in the TSS.

Test 2: Password-Based DEK Derivation Parameters

Applicable when `FCS_CKM_EXT.8` is claimed for password-based DEK derivation.

1. Step: Configure two independent encrypted objects or keystores using the same passphrase where the TOE generates a distinct salt or equivalent diversification value for each object.
2. Step: Inspect the metadata or protected key records that store the derivation parameters.
3. Step: Verify that salts or diversification values differ where the TSS claims they are unique, and that iteration counts, memory cost, and other work-factor parameters match the ST assignments.
4. Step: Attempt to configure derivation parameters below the minimum values claimed in the ST, if administrator configuration is supported.

Expected result: Password-based derivation uses the claimed parameters, rejects unsupported or below-minimum parameters, and produces distinct protected key records when unique salts or diversification values are required by the TSS.

Test 3: Derived DEK Storage Protection

1. Step: Create encrypted database content protected by a derived DEK.

2. Step: Stop the DBMS service to flush persistent state.
3. Step: Inspect database storage, key store files, backups, and exports that contain the protected key record.
4. Step: Search for any known derivation input that should remain secret (for example, a test passphrase) and for any known derived DEK material if the evaluator used a controlled test key or test interface.
5. Step: Verify that public derivation parameters such as salts, labels, or context values are present only as described in the TSS.

Expected result: Secret derivation inputs and derived DEK material are not stored in plaintext. Public derivation parameters are stored only as documented, and the derived DEK is protected under the MEK or equivalent claimed mechanism.

Test 4: ML-KEM Shared-Secret Derivation, Context Binding, and Destruction

Applicable when `FCS_CKM.5` is used to derive an MEK or KEK from the output of `FCS_COP.1/KeyEncap`.

1. Step: Complete an ML-KEM key-establishment operation using the selected encapsulation or decapsulation role and cause the TOE to derive the claimed MEK or KEK.
2. Step: Repeat the operation after changing a context input identified in the TSS, such as the peer identifier, key purpose, transaction identifier, or protected-key-record identifier.
3. Step: Compare observable derived-key identifiers, protected key records, or ciphertext effects and verify they differ when the bound context differs.
4. Step: Inspect persistent key records, logs, and administrator-visible output and verify that the raw ML-KEM shared secret is not persisted or exposed and is not used directly as the MEK, KEK, or DEK.
5. Step: Complete the destruction or lifecycle transition described in the TSS and verify that the raw shared secret is no longer available for reuse.

Expected result: The TOE derives the claimed key with the selected KDF and context, distinct bound contexts do not produce interchangeable protected artifacts, and the raw ML-KEM shared secret is destroyed without being used directly as a database protection key.

5.4. FCS_CKM_EXT.1 Cryptographic Key Management (Master Keys)

This SFR governs the full lifecycle of the Master Encryption Key (MEK) - the root key in the DBMS key hierarchy. It covers internal generation, external import (BYOK), external management (e.g., a KMIP-conformant key manager), and establishment from an ML-KEM shared secret, as well as DEK protection under the MEK and MEK protection and destruction requirements.

5.4.1. FCS_CKM_EXT.1 Evaluation Activities

5.4.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- Which option is selected in FCS_CKM_EXT.1.1 ("Generated Internally", "Imported from External Entity", "Externally Managed", or "Established through Key Encapsulation") and the rationale for that selection.

If "Generated Internally" is selected, additionally verify the TSS describes:

- The algorithm and randomness source used for MEK generation, referencing FCS_CKM.1/SKG and the applicable Catalogue RBG component (FCS_RBG.1).
- The generated MEK key sizes and how they match the ST assignments and selected DEK protection operation.
- The storage location of the generated MEK (e.g., software keystore/wallet, filesystem-protected keyfile, platform key store integration).
- The FCS_CKM_EXT.1.2 protection model selected for the stored MEK and the evidence that the implementation satisfies that model.
- How the TOE ensures the MEK storage location is accessible only to authorized roles.

If "Imported from External Entity" is selected, additionally verify the TSS describes:

- The type of external entity from which the MEK is imported (e.g., Key Management Service, HSM, cloud key vault).
- The protocol used for the import channel (referencing FDP_ITC_EXT.1).
- The mutual TLS authentication requirements for the import operation, including the TOE certificate or credential used to authenticate to the external entity.
- How the imported MEK is stored and protected within the TOE boundary after import, including the FCS_CKM_EXT.1.2 protection model selected for that storage path.

If "Externally Managed" is selected, additionally verify the TSS describes:

- The external key management service used (e.g., a KMIP-conformant key manager) and whether the TOE retrieves the MEK for transient use or invokes MEK operations by reference.
- The protocol used for the key-management channel (referencing FDP_ITC_EXT.1) and the key-management message protocol carried over it (the message protocol itself is not evaluated).
- How the TOE ensures plaintext MEK material is not persistently stored within the TOE boundary, including the handling and lifetime of any transient in-memory copies.
- Where MEK operations are invoked by reference, which cryptographic operations are performed by the external service rather than the TSF, consistent with the FCS_COP components claimed in the ST.
- The TOE's behavior when the external service is unreachable or denies use of the MEK (e.g., whether protected data becomes inaccessible, whether cached material allows continued

operation and for how long).

If "Established through Key Encapsulation" is selected, additionally verify the TSS describes:

- The external entity with which the TOE establishes the ML-KEM shared secret, the ML-KEM parameter set, and whether the TOE performs encapsulation, decapsulation, or both under FCS_COP.1/KeyEncap.
- Whether an ML-KEM key pair used by the TOE is generated within the TOE under FCS_CKM.1/KEM and FCS_RBG.1, or is provisioned from an external source. Externally performed KeyGen, encapsulation, or decapsulation shall be clearly identified as outside the TOE and shall not be claimed as a TOE cryptographic operation.
- For encapsulation, how the TOE authenticates the recipient encapsulation key; for decapsulation, where and how the TOE protects the decapsulation key and how it is destroyed under FCS_CKM.6.
- How the TOE passes the ML-KEM shared secret to FCS_CKM.5, the context bound by the derivation, the resulting MEK size and purpose, and when the raw shared secret is destroyed.
- How the mutually authenticated channel in FDP_ITC_EXT.1 protects the key-establishment exchange. If the exchange occurs inside a higher-level protocol such as TLS, the applicable protocol Functional Package, rather than FCS_COP.1/KeyEncap, shall own that protocol use.

For all selections, additionally verify the TSS describes:

- Each Master Key storage or use path and the corresponding protection model selected in the first selection of FCS_CKM_EXT.1.2: cryptographic protection under FCS_COP.1/KeyWrap or FCS_COP.1/AEAD, non-exportable hardware-backed key storage within the TOE boundary, or no persistent Master Key material within the TOE boundary. If more than one model is selected, the TSS shall map each model to the applicable key path.
- For cryptographic protection, the protecting key, claimed Catalogue operation, one-time values where applicable, and separation between the protected MEK and the protecting key. For non-exportable hardware-backed storage, the evaluated interface, TOE-boundary allocation, authorization enforcement, and non-exportability property. For no-persistent-material handling, the lifetime, clearing, and restart behavior of transient copies and the locations examined to establish that no plaintext or reversibly encoded MEK persists.
- The DEK protection method selected in FCS_CKM_EXT.1.2 (FCS_COP.1/KeyWrap, FCS_COP.1/SKC, or FCS_COP.1/AEAD) and the corresponding Catalogue component claims. Where FCS_COP.1/SKC is selected, the evaluator shall examine any compensating integrity mechanisms for stored DEKs described in the TSS (for example, keystore-container integrity protection); the module does not require such mechanisms under the General-Purpose use case.
- Any IVs, nonces, tweaks, or other one-time values used for DEK protection and whether FCS_OTV_EXT.1 is included.
- How the MEK is protected against disclosure and modification while in use (in-memory protection).
- The MEK destruction procedure, including the method by which MEK material is overwritten or otherwise rendered unrecoverable (consistent with the consumed FCS_CKM.6 component) and the effect of MEK destruction on data encrypted under DEKs that were protected by the

destroyed MEK.

5.4.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- Step-by-step instructions for MEK generation (if "Generated Internally" is selected), including prerequisite configuration (e.g., keystore location), the command or procedure to generate the MEK, and how to verify generation succeeded.
- Step-by-step instructions for MEK import (if "Imported from External Entity" is selected), including prerequisites for establishing the mutual TLS connection to the external KMS (network access, trust anchor configuration, TOE/client certificate configuration), the command or procedure to initiate and complete the import, and how to verify the import succeeded and the MEK is active.
- Step-by-step instructions for external key-management integration (if "Externally Managed" is selected), including prerequisites for establishing the mutual TLS connection to the key management service, endpoint and credential configuration, and how to verify the integration is active and the MEK is usable.
- Step-by-step instructions for ML-KEM establishment (if "Established through Key Encapsulation" is selected), including peer and trusted-key configuration, the TOE's selected role, local ML-KEM key-pair generation or provisioning where applicable, derivation-context configuration, rotation or replacement, and how to verify that the derived MEK is active.
- The DEK protection method used by the TOE and any administrator-visible controls that affect whether DEKs are key-wrapped, symmetrically encrypted, or authenticated-encrypted under the MEK.
- How to configure each selected Master Key protection model, restrict Master Key use to authorized TSF functions and roles, and verify that cryptographic protection, hardware-backed non-exportability, or transient-only handling is active as claimed.
- The MEK rotation procedure, including the steps to generate or import a new MEK, how DEK re-protection occurs (automatically or requiring an administrator action), how to verify data remains accessible after rotation, and any operational impact or downtime considerations.
- The MEK destruction procedure, including the command or procedure to destroy the MEK, a clear warning that MEK destruction renders all data encrypted under dependent DEKs permanently inaccessible, and the method used for key material zeroization (consistent with the `FCS_CKM.6` claims).
- How to restrict access to the MEK storage location to authorized roles only.

5.4.1.3. Test Activities

Test 1: Master Key Generation (Internal)

Applicable when "Generated Internally" is selected in `FCS_CKM_EXT.1.1`.

1. Step: Following the administrator guidance, configure the TOE keystore and generate a new MEK.
2. Step: Verify the MEK artifact (keystore file or equivalent) is created at the location specified in

the guidance and ST.

3. Step: Verify, using system views, status commands, logs, or developer evidence, that the MEK generation method and key size correspond to the FCS_CKM.1/SKG and FCS_RBG.1 claims in the ST.
4. Step: Using a standard unprivileged operating system account (not the database administrator or keystore owner), attempt to read the MEK artifact directly.
5. Step: Inspect the structure of the MEK artifact using a binary or hex editor to verify it is not stored in plaintext.

Expected result: The unprivileged account is denied access to the MEK artifact. The artifact does not contain any recognizable plaintext key material.

Test 2: Master Key Import (BYOK)

Applicable when "Imported from External Entity" is selected in FCS_CKM_EXT.1.1.

1. Step: Configure the TOE and an external KMS test instance per the administrator guidance.
2. Step: Using a network traffic capture tool, monitor the connection between the TOE and KMS.
3. Step: Initiate the MEK import operation from the TOE.
4. Step: Examine the captured traffic and verify it uses the protocol claimed in FDP_ITC_EXT.1 (e.g., TLS 1.3).
5. Step: Verify that the KMS authenticates the TOE using the configured TOE/client certificate or credential during mutual TLS establishment.
6. Step: Verify the TOE reports a successful import (via system view, log, or status command).
7. Step: Encrypt a small test dataset using the imported MEK.
8. Step: Verify the test dataset is accessible (decrypt and read the data).

Expected result: The import channel uses the claimed mutual TLS protocol. The import succeeds and the imported MEK is functional for data encryption and decryption.

Test 2a: Externally Managed Master Key

Applicable when "Externally Managed" is selected in FCS_CKM_EXT.1.1.

1. Step: Configure the TOE and the external key management service test instance (e.g., a KMIP-conformant key manager) per the administrator guidance, and verify the integration is active.
2. Step: Encrypt a small test dataset so DEKs are protected under the externally managed MEK, and verify the data is accessible.
3. Step: Following the TSS description of MEK handling, inspect the TOE's persistent keystore locations and configuration artifacts (using a binary or hex editor where applicable) and verify no plaintext MEK material is persistently stored within the TOE boundary.
4. Step: Make the external key management service unreachable (or revoke the TOE's use of the

MEK at the service).

5. Step: Restart the TOE, or otherwise force it past any transient cached-key lifetime described in the TSS, and attempt to access the protected test dataset.
6. Step: Verify the TOE's behavior matches the TSS description (protected data is inaccessible without the external service, or accessible only within the documented cached lifetime).
7. Step: Restore the service connection and verify protected data is accessible again.

Expected result: The externally managed MEK is functional for data protection; no plaintext MEK material persists within the TOE boundary; and loss of the external service denies access to protected data consistent with the TSS.

Test 2b: Master Key Established through ML-KEM

Applicable when "Established through Key Encapsulation" is selected in `FCS_CKM_EXT.1.1`.

1. Step: Configure the TOE and external peer per the administrator guidance, including the claimed ML-KEM parameter set, selected TOE role, peer-key authentication, and mutual TLS channel.
2. Step: Complete the ML-KEM operation and `FCS_CKM.5` derivation, then verify the TOE reports that the resulting MEK is active.
3. Step: Encrypt a small test dataset and verify that it can be decrypted and read using the established MEK hierarchy.
4. Step: Repeat establishment after rotating the applicable ML-KEM key or derivation context and verify that the TOE obtains a distinct active MEK while retaining access according to the documented rotation procedure.
5. Step: Present an invalid ML-KEM ciphertext or an unauthorized/mismatched encapsulation key, as applicable to the selected TOE role, and verify that establishment fails without activating a new MEK.
6. Step: Inspect persistent records, administrator-visible output, and logs and verify that neither the raw ML-KEM shared secret nor plaintext MEK material is exposed or persistently stored.

Expected result: The TOE establishes and derives the MEK only with an authenticated peer and valid ML-KEM input, uses it successfully in the database key hierarchy, handles rotation as documented, and does not expose the raw shared secret or plaintext MEK.

Test 3: Master Key Rotation

1. Step: Configure the TOE with an initial MEK (MEK_1) and encrypt a test dataset.
2. Step: Record the protected form of a DEK protected by MEK_1 (if observable via a system view or key store inspection tool).
3. Step: Following the administrator guidance, perform a MEK rotation to MEK_2.
4. Step: If the protected DEK form is observable, verify it has changed (now protected under

MEK_2).

5. Step: Verify the test dataset remains readable after rotation.
6. Step: If supported by the TOE, attempt to revert to MEK_1 (which should no longer be active) and verify data remains accessible via MEK_2.

Expected result: DEKs are re-protected under MEK_2 after rotation. Data remains accessible. MEK_1 is no longer the active protection key.

Test 4: Master Key Destruction

1. Step: Configure the TOE with a MEK and encrypt a test dataset. Note that the dataset is accessible (reads succeed).
2. Step: Initiate MEK destruction per the administrator guidance.
3. Step: Immediately after destruction, attempt to read the test dataset.
4. Step: Inspect the persistent MEK storage location (keystore file or equivalent) to verify the MEK material has been removed or overwritten.

Expected result: The test dataset is no longer accessible after MEK destruction. The persistent storage location no longer contains the MEK material in any recoverable form.

Test 5: DEK Protection Method Consistency

1. Step: Identify the DEK protection method selected in `FCS_CKM_EXT.1.2` and the corresponding consumed component (`FCS_COP.1/KeyWrap`, `FCS_COP.1/SKC`, or `FCS_COP.1/AEAD`).
2. Step: Create encrypted database content that causes a DEK to be generated or derived and protected under the active MEK.
3. Step: Inspect available key store metadata, protected key records, system views, audit logs, or developer evidence to determine the operation used to protect the DEK.
4. Step: If the selected operation uses IVs, nonces, tweaks, tags, or other one-time values, verify that the associated metadata is present or derived as described in the TSS and that `FCS_OTV_EXT.1` is included when required.
5. Step: If the evaluator has controlled DEK material or a test interface exposing a known test key, search persistent key records and storage artifacts for that known plaintext key material.
6. Step: If the TOE does not expose controlled DEK material, verify that no TOE interface exports plaintext DEKs and that the protected key records are not documented as plaintext key storage.

Expected result: The observed DEK protection method is consistent with the ST selections and included Catalogue components. DEKs are stored only in protected form, one-time-value handling is consistent with the TSS, and the TOE does not expose plaintext DEK material.

Test 6: Master Key Protection Model Consistency

1. Step: Identify every Master Key storage or use path and the protection model selected for it in the first selection of FCS_CKM_EXT.1.2.
2. Step: Using an unauthorized database role and, where applicable, a standard unprivileged platform account, attempt to invoke Master Key operations, read a persistent Master Key artifact, or export the key through the documented key-store interface.
3. Step: For a cryptographically protected persistent copy, inspect the protected record and the TSS/developer evidence to verify that the claimed FCS_COP.1/KeyWrap or FCS_COP.1/AEAD operation protects the MEK and that the protecting key is not stored alongside it in an equivalent plaintext form.
4. Step: For non-exportable hardware-backed storage, exercise the documented export or read interface, or the closest negative operation the interface permits, and verify that plaintext export is refused and only authorized TOE operations can use the key.
5. Step: For a no-persistent-material path, stop and restart the TOE or force it past the documented transient-key lifetime, then inspect the persistent locations and artifacts identified in the TSS and verify that no plaintext or reversibly encoded Master Key remains within the TOE boundary.
6. Step: Verify that the selected mechanism is TSF behavior within the evaluated TOE boundary and is not satisfied solely by the operational-environment protections assigned to OE.KEY_STORE_PROTECTION.

Expected result: Each Master Key path implements the protection model selected in the ST; unauthorized use or export is denied; persistent material is cryptographically protected or non-exportable as claimed; transient-only paths leave no persistent Master Key; and environmental storage protection is treated only as supporting protection.

5.5. FCS_COP.1/KeyEncap ML-KEM Encapsulation and Decapsulation (Selection-Based)

Algorithm-level ML-KEM validation is performed using the Catalogue evaluation methods and applicable scheme-recognized validation program. The following activities verify the DBMS integration and the operation roles selected in the ST.

5.5.1. FCS_COP.1/KeyEncap Evaluation Activities

5.5.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- The selected ML-KEM parameter set and whether the TOE performs encapsulation, decapsulation, or both.
- The source, storage location, authentication, protection, rotation, and destruction of each ML-KEM encapsulation or decapsulation key used by the TOE.
- Whether the TOE generates an ML-KEM key pair under FCS_CKM.1/KEM; if KeyGen is external, the

TOE boundary and provisioning interface shall be identified without claiming external generation as a TOE operation.

- For encapsulation, the FCS_RBG.1 source used by the operation. For a decapsulation-only implementation, the evaluator shall verify that no RBG dependency is asserted merely for decapsulation.
- How the resulting shared secret is passed to FCS_CKM.5, the derived MEK or KEK purpose, and how the raw shared secret is destroyed under FCS_CKM.6.
- When a KEM-derived KEK protects transferred keying material, the FCS_COP.1/KeyWrap or FCS_COP.1/AEAD operation that protects the existing MEK or DEK; ML-KEM shall not be described as directly encapsulating that existing key.
- Any externally performed ML-KEM operation and any ML-KEM use owned by a protocol Functional Package, such as TLS, so those operations are not also claimed under this component.

5.5.1.2. Guidance Activities

The evaluator shall verify the guidance describes how to configure the selected TOE role, establish and authenticate peer encapsulation keys, provision or generate TOE key pairs where applicable, rotate and destroy ML-KEM keys, diagnose rejected ciphertexts or keys, and identify any operation allocated to an external KMS or other environmental component.

5.5.1.3. Test Activities

Test 1: ML-KEM Encapsulation Integration

Applicable when encapsulation is selected.

1. Step: Configure an authenticated recipient encapsulation key and cause the TOE to perform encapsulation.
2. Step: Verify the recipient can decapsulate the resulting ciphertext to the same shared secret used as input to the TOE's claimed derivation operation, using evaluator or developer instrumentation that does not expose secret material in operational interfaces.
3. Step: Replace the recipient key with an unauthorized or mismatched key and verify the TOE rejects it or otherwise prevents establishment with an unauthenticated peer.
4. Step: Verify evidence identifies the claimed FCS_RBG.1 source for encapsulation.

Expected result: Encapsulation succeeds only for an authenticated recipient key, interoperates at the selected parameter set, and uses the claimed randomness source.

Test 2: ML-KEM Decapsulation Integration and Invalid Ciphertext Handling

Applicable when decapsulation is selected.

1. Step: Configure the TOE decapsulation key and provide a valid ML-KEM ciphertext generated for the corresponding encapsulation key.

2. Step: Verify the TOE completes the claimed derivation or key-establishment workflow without exposing the raw shared secret.
3. Step: Modify the ciphertext and repeat the operation, then verify that the resulting value cannot activate, unwrap, or decrypt the protected database key material associated with the valid exchange.
4. Step: Destroy or deactivate the decapsulation key and verify subsequent operations using that key identifier are refused.

Expected result: Valid ciphertext supports the claimed workflow, malformed or mismatched ciphertext cannot recover protected database key material, and destroyed decapsulation keys cannot be reused.

5.6. FCS_CKM.6 Cryptographic Key Destruction (Catalogue Reference)

This consumed SFR is reproduced in the PP-Module from `FCS_CKM.6` (CCDB-018 Section 3.7). The primary evaluation methods for key destruction are defined by the Catalogue and associated evaluation-methods guidance. This SD provides only supplementary activities specific to the DBMS key hierarchy and destruction consequences.

5.6.1. FCS_CKM.6 Evaluation Activities

5.6.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- That the destruction method is consistent with the included `FCS_CKM.6` operations.
- The specific destruction method selected (e.g., single overwrite with zeros, overwrite with a pseudo-random pattern) and confirmation that it applies to both MEK and DEK material.
- When key destruction occurs (e.g., explicit administrator command, expiry-triggered destruction, key rotation completion) and whether any keys are excluded from destruction requirements.

5.6.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- The key destruction commands or procedures available to authorized administrators.
- A clear statement of the irreversibility of key destruction and the impact on encrypted data.

5.6.1.3. Test Activities

Note: Detailed zeroization testing is performed as part of the Crypto Catalogue evaluation methods for `FCS_CKM.6`. The following supplementary test verifies DBMS-specific destruction behaviour.

Test 1: Key Material Absence Post-Destruction

1. Step: Create and activate a MEK and record any observable identifier (key ID, fingerprint).
2. Step: Initiate key destruction per the administrator guidance.
3. Step: After destruction, query any system views or key management interfaces that list active or archived keys.
4. Step: Inspect the persistent key store location for the destroyed key's identifier or material.

Expected result: The destroyed key no longer appears in system views or key management interfaces. The key store location does not contain the key material in any recognizable form.

Chapter 6. FDP Class: User Data Protection

6.1. FDP_DAR_EXT.1 Data-at-Rest Encryption Strategy

This SFR governs how the TOE applies encryption to persistent database storage. The ST author must select either Storage-Scope Encryption or Granular Data Encryption. The algorithms used are defined by the consumed FCS_COP.1/SKC and/or FCS_COP.1/AEAD component, with FCS_OTV_EXT.1 included when required by the selected mode.

Note: When this PP-Module is used as a mandatory component of the DBMS Cloud Module PP-Configuration, data-at-rest cryptographic testing remains governed by this SD. The Cloud Module SD may add cloud-deployment integration checks, but it does not provide an alternate environmental path that replaces the mandatory Crypto Module claim.

6.1.1. FDP_DAR_EXT.1 Evaluation Activities

6.1.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- Which strategy is selected in FDP_DAR_EXT.1.1: Storage-Scope Encryption or Granular Data Encryption.

If "Storage-Scope Encryption" is selected, additionally verify the TSS:

- Defines the scope of encryption - which persistent database storage objects are encrypted. At minimum, the TSS shall address: data files, transaction/redo log files, undo/rollback segment files, temporary tablespace or temp files, backup artifacts generated by the TOE, export artifacts generated by the TOE, snapshots under TOE control, and database-managed diagnostic or spill files that can contain user data.
- Identifies any excluded storage-object categories and provides a rationale showing that each exclusion either does not contain plaintext user data or is protected by an equivalent claimed mechanism.
- Identifies the encryption algorithm and mode, including any IV, nonce, tweak, tag, or other one-time-value handling, consistent with the included FCS_COP.1/SKC, FCS_COP.1/AEAD, and FCS_OTV_EXT.1 components.
- Describes how encryption is applied transparently to the DBMS (i.e., applications do not need modification to benefit from encryption).

If "Granular Data Encryption" is selected, additionally verify the TSS:

- Defines the granularity level (column-level, row-level, cell-level, or a combination).
- Identifies which encryption modes are supported (e.g., deterministic, randomized/probabilistic) and the security implications of each:
  - **Deterministic** encryption produces the same ciphertext for the same plaintext and key, enabling equality searches and indexing but leaking frequency information.

    - **Randomized (probabilistic)** encryption produces different ciphertext for the same plaintext, providing stronger confidentiality but preventing sorting, range queries, and indexing on encrypted values.
- Describes how encrypted columns or data elements interact with database indexing, query processing, and backup/restore operations.
- Identifies any data types, storage formats, indexes, generated columns, logs, exports, or backups that are excluded from granular encryption and provides rationale.

For all selections, additionally verify the TSS:

- Identifies the encryption algorithm, key size, mode of operation, and one-time-value handling, confirming consistency with the included `FCS_COP.1/SKC`, `FCS_COP.1/AEAD`, and `FCS_OTV_EXT.1` Catalogue components. The evaluator shall verify that the components selected in `FDP_DAR_EXT.1.2` are included in the ST and are the cryptographic operations actually performing the FDP_DAR_EXT.1.1 encryption.
- Explains how compression, deduplication, encoding, or page formatting affect storage inspection tests, and how the evaluator can obtain representative raw artifacts for inspection.

6.1.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- Step-by-step instructions for enabling encryption (Storage-Scope or Granular) appropriate to the selection made in the ST.
- **For Storage-Scope Encryption:**
    - Instructions for enabling encryption on a new database and on an existing unencrypted database.
    - Guidance on how to verify the encryption status of individual storage objects (e.g., via system views, status queries, or utility commands).
    - Backup, export, snapshot, and restore procedures for encrypted databases, including key availability requirements during restore.
    - Guidance on the encryption status of temporary storage, redo logs, archived log files, undo/rollback files, backups, exports, and any documented exclusions.
- **For Granular Data Encryption:**
    - Instructions for encrypting individual columns or data elements.
    - Guidance on selecting the appropriate encryption mode (deterministic vs. randomized) for different use cases.
    - Instructions for querying and indexing encrypted data, noting any limitations.
    - Guidance on key management for column-level encryption (e.g., whether separate keys are used per column or a shared key is used).
- Performance considerations and tuning recommendations related to encryption.
- How to disable or re-encrypt data and the impact on key management.
- Any steps needed to disable or account for compression, deduplication, external encoding, or

application-layer transformations during evaluator storage-inspection tests.

6.1.1.3. Test Activities

Test 1: Storage-Scope Encryption - Storage Inspection

Applicable when "Storage-Scope Encryption" is selected. The evaluator requires low-level access to storage files (binary editor, raw disk access, or equivalent).

1. Step: If supported by the TOE and consistent with the evaluation configuration, create a positive-control unencrypted object or pre-encryption artifact containing distinctive plaintext strings to confirm the inspection method can locate plaintext when it is present.
2. Step: Create a test database or tablespace and insert rows containing multiple distinctive, easily searchable plaintext strings across representative data types (e.g., fixed text, variable text, large object data, indexed values, and values likely to appear in logs or temporary files).
3. Step: Ensure Storage-Scope Encryption is enabled for the database or storage scope before inserting the protected test data.
4. Step: Disable compression, deduplication, or external encoding if configurable for evaluation; otherwise document how those transformations are accounted for in the search method.
5. Step: Force checkpoints, log flushes, and temporary operations as described in guidance, then stop the DBMS service to flush all in-memory buffers and caches to disk.
6. Step: Map the raw artifacts to the storage-object categories identified in the TSS and scan each covered artifact for the known plaintext strings and expected encoded representations.

Expected result: The inspection method detects plaintext in the positive-control artifact when one is used. The plaintext strings are not found in storage objects that the TSS claims are encrypted. The observed artifacts correspond to the storage-object categories identified in the TSS.

Test 2: Storage-Scope Encryption - Coverage and Documented Exclusions

Applicable when "Storage-Scope Encryption" is selected.

1. Step: Execute queries that generate redo/transaction log entries, undo/rollback entries, temporary sort or spill operations, index maintenance, backup artifacts, and export artifacts containing the same distinctive plaintext strings used in Test 1.
2. Step: Stop the DBMS service or otherwise obtain quiesced raw artifacts according to the guidance.
3. Step: Inspect each covered artifact category identified in the TSS for the distinctive plaintext strings.
4. Step: For each documented exclusion, inspect a representative artifact or review developer evidence sufficient to confirm that the exclusion rationale is accurate.

Expected result: The scope of encryption coverage in practice is consistent with the TSS. Covered artifacts do not contain the distinctive plaintext strings. Documented exclusions are accurate, bounded, and do not expose plaintext user data that the selected strategy claims to protect, unless

the artifact is protected by an equivalent claimed mechanism.

Test 3: Granular Encryption - Column Encryption Inspection

Applicable when "Granular Data Encryption" is selected.

1. Step: Create a test table containing at least one encrypted column and at least one intentionally unencrypted column as a positive control.
2. Step: Insert rows with distinctive known plaintext values in both column types, including values likely to appear in indexes, logs, exports, and backups.
3. Step: Disable or account for compression, deduplication, and encoding as described in the guidance.
4. Step: Stop the DBMS service or otherwise obtain quiesced raw artifacts.
5. Step: Inspect the raw storage files, relevant index files, logs, exports, and backups using a binary editor or equivalent tool.

Expected result: Known plaintext values are present for intentionally unencrypted positive-control data when the inspection method can observe raw plaintext. Known plaintext values are not present for encrypted columns in artifacts that the TSS claims are protected. Any artifacts excluded from granular protection match the TSS rationale.

Test 4: Granular Encryption - Deterministic vs. Randomized Mode Behaviour

Applicable when "Granular Data Encryption" is selected and the TOE supports both deterministic and randomized encryption modes.

Procedure for Deterministic Encryption:

1. Step: Create a column with deterministic encryption enabled. Insert the same plaintext value in two separate rows.
2. Step: Inspect the stored ciphertext values for the two rows (via a system view, binary inspection, or a raw export).
3. Step: Attempt to create an index on the deterministically encrypted column.
4. Step: Perform an equality search (`WHERE col = 'known_value'`) on the column.

Expected result: Both rows storing the same plaintext produce the same ciphertext. Index creation and equality searches succeed.

Procedure for Randomized Encryption:

1. Step: Create a column with randomized encryption enabled. Insert the same plaintext value in two separate rows.
2. Step: Inspect the stored ciphertext values.
3. Step: Attempt to create a B-tree or standard index on the randomized column.

Expected result: The two rows storing the same plaintext produce different ciphertext values. Index creation fails or is rejected per the TSS description.

6.2. FDP_DIT_EXT.1 Data-in-Transit Protection

This SFR governs the protection of data transmitted between the TOE and external entities. TLS and certificate validation are evaluated using the Functional Package for TLS v2.1 [[TLS_FP]] and the Functional Package for X.509 Certificates [[X509_FP]], as applicable. CCDB-018 specifies cryptographic primitives and does not specify TLS or certificate-validation components. This SD covers the DBMS-specific application of those channels: which channels are protected, the protocol selection, and endpoint authentication.

Application Note: FDP_DIT_EXT.1 is mandatory. The evaluator shall also apply the TLS Evaluation Activities from the Functional Package for TLS v2.1 for the `FCS_TLSC_EXT.1` or `FCS_TLSS_EXT.1` components, and the X.509 Evaluation Activities for the `FIA_X509_EXT` components, claimed in the ST. When this module is claimed with another module (for example, the DBMS Cloud Module), that module identifies the deployment-specific channels (for example, cloud IAM, audit export, or tenant-facing connections) to which these activities apply.

6.2.1. FDP_DIT_EXT.1 Evaluation Activities

6.2.1.1. TSS Activities

The evaluator shall verify the TSS:

- identifies each channel over which the TOE transmits data to or from external entities, including client-to-database connections, database-to-database connections, management and administrative channels, audit export channels, and connections to external services;
- identifies the protocol protecting each channel: TLS (including HTTPS over TLS where applicable), which is mandatory for every conforming TOE, or a supplementary protocol claimed in the `FDP_DIT_EXT.1.1` selection — the evaluator shall verify the TLS Functional Package claim is present regardless of any supplementary protocol, and that any supplementary protocol is claimed from an applicable Functional Package and implemented within the TOE boundary (not by the underlying operating system or kernel);
- identifies the TLS and X.509 components included from the Functional Package for TLS v2.1 and the Functional Package for X.509 Certificates that implement each protected channel; and
- describes how channel endpoints are authenticated, including certificate validation, where required by the selected protocol.

6.2.1.2. Guidance Activities

The evaluator shall verify the operational guidance describes how to configure each protected channel, including protocol selection, certificate and trust-anchor configuration, and any options that enable or disable data-in-transit protection for a channel.

6.2.1.3. Test Activities

Test 1: Protected Channel Establishment

1. Step: Configure each claimed channel according to the operational guidance.
2. Step: Establish a connection over each channel and capture the traffic.
3. Step: Verify that the channel uses the protocol claimed in FDP_DIT_EXT.1.1 (e.g., TLS 1.2 or TLS 1.3) and that application data is not transmitted in plaintext.

Test 2: Endpoint Authentication

1. Step: For a channel that uses certificate authentication, present an untrusted, expired, or otherwise invalid certificate to the TOE endpoint.
2. Step: Verify that the TOE rejects the connection consistent with the claimed X.509 certificate-validation behaviour.

Test 3: Negotiation Failure Handling

1. Step: Attempt to negotiate a non-claimed or downgraded protocol or cipher suite with the TOE endpoint.
2. Step: Verify that the TOE rejects the connection rather than falling back to an unprotected channel.

6.3. FDP_ITC_EXT.1 Trusted Channel for Key Import, External Key Management, and Key Establishment (Selection-Based)

This SFR governs the mutually authenticated trusted channel used to import the Master Encryption Key from an external Key Management System (KMS) in BYOK scenarios, communicate with an external key management service that remains the authoritative store for the MEK (e.g., a KMIP-conformant key manager), or exchange the values needed to establish keying material through ML-KEM. The channel protocol is evaluated using the Functional Package for TLS v2.1 [[TLS_FP]]. This SD covers the DBMS-specific channel properties: authentication behaviour, key material protection in transit, and failure handling. The activities below apply to whichever external Key Origin is selected in FCS_CKM_EXT.1.1; "key operation" is read as the applicable import, external-management, or key-establishment operation.

Application Note: The evaluator shall also apply TLS Evaluation Activities from the Functional Package for TLS v2.1 for the FCS_TLSC_EXT.1 or FCS_TLSS_EXT.1 components claimed in the ST. The activities below are supplementary and focus on the DBMS-to-external-entity channel specifically; any key-management message protocol carried over the channel (e.g., KMIP) is not itself evaluated. Mutual TLS is mandatory for every external Key Origin path. ML-KEM inside TLS is owned by the TLS Functional Package and is not claimed again as FCS_COP.1/KeyEncap.

6.3.1. FDP_ITC_EXT.1 Evaluation Activities

6.3.1.1. TSS Activities

The evaluator shall verify the TSS describes:

- The specific mutual TLS protocol used for the trusted channel and consistency with the components included from the Functional Package for TLS v2.1.
- The certificate infrastructure used for mutual TLS, including TOE/client certificate configuration, KMS server certificate validation, trust anchors, certificate revocation handling if claimed, and where TOE/client certificates and private keys are stored within the TOE.
- How the TOE handles channel establishment failures, including rejection of KMS connections presenting untrusted or expired certificates, handling of TLS negotiation failures (e.g., protocol version mismatch, cipher suite mismatch), and behavior when the KMS is unreachable (timeout, fail-closed vs. fail-open consideration).
- That key material transmitted over the channel is protected by the TLS layer and is never transmitted in plaintext.
- The FDP_ITC_EXT.1.4 selection: where supplementary end-to-end protection is selected, whether transferred keying material is directly protected by FCS_COP.1/KeyWrap, or whether ML-KEM establishes a shared secret from which FCS_CKM.5 derives a KEK and FCS_COP.1/KeyWrap or FCS_COP.1/AEAD protects the existing MEK or DEK. The TSS shall identify the TOE's ML-KEM role, the locations of the ML-KEM keys, derivation inputs, KEK, and transferred key, and where unwrapping, authenticated decryption, or decapsulation occurs.

6.3.1.2. Guidance Activities

The evaluator shall verify the guidance describes:

- Prerequisites for establishing the trusted channel, including certificate requirements (CA certificates to be trusted, TOE/client certificate generation and installation, KMS server certificate requirements) and network access requirements (ports, firewall rules).
- Instructions for configuring the TOE to connect to the external KMS, including KMS endpoint configuration (hostname/IP, port) and certificate or credential configuration for the channel.
- The procedure for rotating or updating certificates used in the channel.
- How to test channel connectivity before performing a key import, and how to diagnose channel establishment failures.
- The TOE's behavior when the KMS is unavailable (e.g., whether cached key material allows continued operation or whether the TOE fails to start or denies new connections).
- When supplementary ML-KEM protection is selected, how to configure the TOE role and peer keys, establish the KEM-derived KEK, select the KeyWrap or AEAD protection operation, rotate the relevant keys, and diagnose failed decapsulation, derivation, unwrap, or authenticated-decryption operations.

6.3.1.3. Test Activities

Test 1: Successful Channel Establishment and Key Import

1. Step: Configure the TOE and an external KMS test instance with valid, mutually trusted certificates.
2. Step: Initiate a key import operation.
3. Step: Using a network capture tool (e.g., Wireshark), capture traffic between the TOE and KMS.
4. Step: Verify the channel uses the TLS protocol version claimed in the ST.
5. Step: Verify, using KMS logs, TLS handshake evidence, or equivalent evidence, that the KMS authenticated the TOE/client certificate and the TOE authenticated the KMS server certificate.

Expected result: The mutually authenticated TLS channel is established successfully. The key import completes. Network capture confirms TLS is in use and no plaintext key material is visible in the packet payloads.

Test 2: Rejection of Untrusted KMS Certificate

1. Step: Configure a test KMS instance with a self-signed certificate not present in the TOE's trusted CA store.
2. Step: Attempt a key import from the TOE to this untrusted KMS.
3. Step: Observe the TOE's behavior and any error messages or audit log entries generated.

Expected result: The TOE rejects the channel establishment. The key import does not proceed. No key material is transmitted. An audit event or error is generated consistent with the TSS description.

Test 3: Mutual TLS - Client Certificate Required

1. Step: Remove or invalidate the TOE's client certificate (or configure the KMS to require client authentication with a different trusted CA than the one that signed the TOE certificate).
2. Step: Attempt a key import operation.
3. Step: Observe the result and any error messages or audit log entries.

Expected result: The KMS rejects the connection. The key import does not proceed. The TOE logs an appropriate error consistent with the TSS description.

Test 4: Key Material Confidentiality in Transit

1. Step: Using a network capture tool, capture all traffic between the TOE and the KMS during a complete key import operation.
2. Step: Examine the captured packets for any plaintext representation of the imported key

material.

Expected result: Key material is not found in plaintext in any captured packets. The payload of all post-handshake TLS records is encrypted and not interpretable without the session keys.

Test 5: KEM-Derived Protection for Transferred Keying Material

Applicable when FDP_ITC_EXT.1.4 selects supplementary protection using ML-KEM.

1. Step: Configure the TOE and external entity for the selected ML-KEM role and establish a shared secret over the mutually authenticated trusted channel.
2. Step: Cause FCS_CKM.5 to derive the KEK and transfer a known test MEK or DEK protected by the selected FCS_COP.1/KeyWrap or FCS_COP.1/AEAD operation.
3. Step: Verify the TOE receives and can use the same test key while network capture shows only the mutually authenticated TLS exchange and protected key artifact.
4. Step: Repeat using an incorrect ML-KEM key, modified ML-KEM ciphertext, or mismatched derivation context and verify that unwrap or authenticated decryption fails and the transferred key is not activated.
5. Step: Inspect persistent records and operational interfaces and verify that the raw ML-KEM shared secret and derived KEK are not exposed or persistently stored except where protected storage is explicitly claimed and evaluated.

Expected result: ML-KEM establishes the input to KEK derivation, KeyWrap or AEAD protects the transferred key, invalid or mismatched establishment inputs prevent key recovery, and the mutually authenticated channel remains in use.

Appendix A: Evaluation Activity Summary Table

This table lists only the DBMS-specific and DBMS-integration activities defined by this SD. It does not restate the algorithm-level Evaluation Activities associated with the consumed Catalogue components. Any validation evidence used for algorithm correctness is recorded in the Cryptographic Validation Coverage Matrix required by Section 3.1.2, “Use of Scheme-Recognized Cryptographic Validation Results”.

Table 2. FCS Class Evaluation Activities Summary

SFR	TSS Activities	Guidance Activities	Test Activities
FCS_CKM.1/SKG Cryptographic Key Generation (TOE-Generated Symmetric Keys) (Selection-Based)	Verify direct DEK generation mechanism when claimed, internally generated MEK mapping when selected, key sizes, key hierarchy, lifecycle events, randomness source	Configure claimed direct generation functions, verify generation status, configuration parameters	Up to 2 DEK generation tests when direct DEK generation is claimed; internally generated MEK generation is tested under FCS_CKM_EXT.1
FCS_CKM.1/KEM Cryptographic Key Generation (ML-KEM Key Pair) (Selection-Based)	Verify parameter set, randomness source, key use, secret-key protection, lifecycle, and boundary when TOE KeyGen is claimed	Configure local KeyGen, key use, protection, rotation, and destruction	1 test: ML-KEM key-pair generation and lifecycle
FCS_CKM.5 / FCS_CKM_EXT.8 Key Derivation (Conditional)	Verify derivation input type, parameters and context, supporting primitive claims, output sizes, derived-key protection, and ML-KEM shared-secret destruction where applicable	Configure derivation parameters, passphrase policy where applicable, ML-KEM key-purpose context, and parameter backup/restore	Up to 4 tests: Non-password parameter binding, password-based parameters, derived DEK storage protection, ML-KEM context binding and destruction

SFR	TSS Activities	Guidance Activities	Test Activities
FCS_CKM_EXT.1 Cryptographic Key Management (Master Keys)	Verify key origin selection, MEK generation, import, external management, or ML-KEM establishment, explicit MEK protection model, DEK protection method, storage and protection, in-memory protection, and destruction procedure	Configure MEK generation, import, external key-management integration, or ML-KEM establishment, MEK and DEK protection methods, rotation procedure, destruction procedure, access restrictions	Up to 8 tests: one of four key-origin paths, rotation, destruction, DEK protection method consistency, and MEK protection model consistency
FCS_COP.1/KeyEncap ML-KEM Encapsulation and Decapsulation (Selection-Based)	Verify selected operation roles, parameter set, key source and protection, randomness dependencies, shared-secret handling, KDF integration, and external/protocol boundary	Configure roles, peer-key authentication, local keys, rotation, destruction, and external allocation	Up to 2 tests: encapsulation integration; decapsulation integration and invalid-ciphertext handling
FCS_CKM.6 Key Destruction (Catalogue Reference)	Verify included `FCS_CKM.6` operations, destruction method, and destruction trigger events	Configure destruction commands, irreversibility warning	1 test: Key material absence post-destruction

SFR	TSS Activities	Guidance Activities	Test Activities
FDP_DAR_EXT.1 Data-at-Rest Encryption Strategy	Verify strategy selection, encryption scope and documented exclusions (Storage-Scope) or granularity and modes (Granular), algorithm and one-time-value consistency	Configure encryption enablement, status verification, backup/export/restore, storage-inspection prerequisites, performance guidance	4 tests: Storage inspection, coverage and exclusions (Storage-Scope); column inspection, deterministic vs. randomized (Granular)
FDP_DIT_EXT.1 Data-in-Transit Protection	Verify protected channels enumerated, protocol selection per channel, claimed TLS/X.509 components, endpoint authentication	Configure protected channels, protocol and certificate settings, enable/disable options	3 tests: protected channel establishment, endpoint authentication, negotiation failure handling
FDP_ITC_EXT.1 Trusted Channel for Key Import, External Key Management, and Key Establishment (Selection-Based)	Verify mutual TLS protocol, certificate infrastructure, failure handling, plaintext protection, and compound KEM-derived KEK plus KeyWrap/AEAD construction where selected	Configure mutual TLS prerequisites, external endpoint, certificate rotation, failure diagnostics, and supplementary KEM-derived transfer protection	Up to 5 tests when an external Key Origin is selected: mutual TLS operation, certificate failures, in-transit confidentiality, and KEM-derived transferred-key protection

SFR	TSS Activities	Guidance Activities	Test Activities
FPT_TST.1/RBG TSF Self-Testing (Catalogue Dependency)	Verify the TSS identifies the DRBG self-tests and when they run	Documented per the TSS; no distinct guidance activity	Per Section 3.1.2, "Use of Scheme-Recognized Cryptographic Validation Results": CMVP module-validation evidence for self-test execution, subject to Certification Body approval, pending the Catalogue Evaluation Methods
FPT_FLS.1/RBG Failure with Preservation of Secure State (Catalogue Dependency)	Verify the TSS identifies the failure behavior preserving a secure state	Documented per the TSS; no distinct guidance activity	Per Section 3.1.2, "Use of Scheme-Recognized Cryptographic Validation Results": module error-state evidence plus the TOE-integration check that RBG-dependent services are refused on self-test failure

Appendix B: Test Environment Requirements

B.1. General Test Environment

The evaluator shall deploy the TOE in a representative test environment that includes:

1. A fully configured DBMS installation per the developer's preparative guidance.
2. Access to low-level storage inspection tools (binary editor, hex dump utility, or raw disk access) for data-at-rest tests.
3. A network traffic capture capability (e.g., Wireshark, tcpdump) for key import, external key-management, and data-in-transit channel tests.
4. Tooling to present invalid, untrusted, or expired certificates to the TOE endpoints and to attempt protocol or cipher-suite downgrade, for the FDP_DIT_EXT.1 and FDP_ITC_EXT.1 tests.
5. An external KMS test instance (hardware or software) for BYOK, external key-management, and trusted channel tests.
6. An ML-KEM-capable peer or test harness supporting the selected parameter set and TOE operation role when `FCS_CKM.1/KEM`, `FCS_COP.1/KeyEncap`, or KEM-derived transfer protection is claimed.
7. Separate operating system accounts with differing privilege levels for access control tests.

B.2. KMS Test Environment

For evaluations involving `FDP_ITC_EXT.1` and any external `FCS_CKM_EXT.1` origin path:

1. Deploy a KMS test instance (e.g., HashiCorp Vault development instance, or an equivalent test KMS) accessible from the TOE test environment.
2. Provision a valid PKI test hierarchy: a test CA, a KMS server certificate signed by the test CA, and a TOE client certificate signed by the test CA.
3. Configure the KMS with a test MEK or key protection key suitable for the import test.
4. Configure the TOE's trusted CA store with the test CA certificate.
5. Confirm network capture is possible between the TOE and KMS prior to beginning tests.
6. When ML-KEM is claimed, configure the external entity or test harness with the complementary ML-KEM operation, authenticated encapsulation keys, known invalid or mismatched test inputs, and the ability to confirm KDF and KeyWrap/AEAD interoperability without exposing secret material through operational interfaces.

B.3. Storage Inspection Prerequisites

For data-at-rest tests (FDP_DAR_EXT.1):

1. Identify the physical or logical storage files used by the TOE for data files, redo/transaction logs, undo files, temporary tablespace, backups, exports, snapshots under TOE control, and database-managed diagnostic or spill files that can contain user data.
2. Confirm the ability to stop the DBMS service and access raw storage files while the service is stopped.
3. Identify and document the location of any keystore files for DEK/MEK storage inspection.
4. Identify any documented exclusions from Storage-Scope or Granular encryption and the evidence needed to confirm the exclusion rationale.
5. Configure a positive-control plaintext artifact when supported, and disable or account for compression, deduplication, encoding, or page-format transformations that could affect storage inspection.

Appendix C: Test Evidence Requirements

For each test activity, the evaluator shall record:

1. Test date and environment configuration (DBMS version, OS version, KMS type/version if applicable).
2. Test steps executed.
3. Actual results observed.
4. Pass/fail determination with rationale.
5. Any deviations from documented procedures and justification.

Evidence should include:

1. Command output or screenshots demonstrating test execution and results.
2. Audit log excerpts showing generated records where applicable.
3. Network captures (for FDP_ITC_EXT.1 tests), with session keys excluded.
4. Binary/hex excerpts from storage inspection (for FDP_DAR_EXT.1 tests), including positive-control evidence when used and confirming presence or absence of plaintext across covered and excluded artifacts.
5. Configuration files or system view output used during testing.

Appendix D: Acronyms and Abbreviations

Table 3. Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
AGD	Assurance Class: Guidance Documents
ATE	Assurance Class: Tests
AVA	Assurance Class: Vulnerability Assessment
BYOK	Bring Your Own Key
CA	Certificate Authority
CAVP	Cryptographic Algorithm Validation Program
CB	Certification Body
CC	Common Criteria
CCDB	Common Criteria Development Board
CEM	Common Evaluation Methodology
CMVP	Cryptographic Module Validation Program
cPP	collaborative Protection Profile
DBMS	Database Management System
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
EA	Evaluation Activity
ETR	Evaluation Technical Report
FCS	Functional class: Cryptographic Support
FDP	Functional class: User Data Protection
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HSM	Hardware Security Module
iTC	international Technical Community
KMS	Key Management Service / Key Management System
MEK	Master Encryption Key
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
PP	Protection Profile
RBG	Random Bit Generator
RNG	Random Number Generator

Acronym	Meaning
SAR	Security Assurance Requirement
SD	Supporting Document
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TDE	Transparent Data Encryption
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
XTS	XEX-based Tweaked CodeBook mode with ciphertext Stealing

Appendix E: Document References

- [CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, CCMB-2022-11-001, CC:2022 Revision 1, November 2022.
- [CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, CCMB-2022-11-002, CC:2022 Revision 1, November 2022.
- [CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements, CCMB-2022-11-003, CC:2022 Revision 1, November 2022.
- [CC4] Common Criteria for Information Technology Security Evaluation, Part 4: Framework for the specification of evaluation methods and activities, CCMB-2022-11-004, CC:2022 Revision 1, November 2022.
- [CC5] Common Criteria for Information Technology Security Evaluation, Part 5: Pre-defined packages of security requirements, CCMB-2022-11-005, CC:2022 Revision 1, November 2022.
- [CCE] Common Criteria for Information Technology Security Evaluation, Errata and interpretation for CC:2022 (Release 1) and CEM:2022 (Release 1), CCMB-002, Version 1.1, July 22, 2024.
- [CEM] Common Methodology for Information Technology Security Evaluation, Evaluation methodology, CCMB-2022-11-006, CEM:2022 Revision 1, November 2022.
- [Crypto_Catalog] Specification of Functional Requirements for Cryptography (CCDB-018), Version 1.0, January 2025. Available at commoncriteria/crypto-catalog and commoncriteriaportal.org.
- [Crypto_Eval_Methods] Evaluation Methods for Cryptographic Security Functional Requirements, companion evaluation-methods document identified by CCDB-018. CCDB-018 states that this document will specify the Catalogue Evaluation Activities; use the version recognized by the responsible Certification Body when available.
- [NIST_CAVP] National Institute of Standards and Technology, Cryptographic Algorithm Validation Program. Available at csrc.nist.gov.
- [NIAP_CAVP_POLICY] National Information Assurance Partnership, Frequently Asked Questions for NIAP Policy #5, 28 May 2021. Available at niap-ccevs.org.
- [CAN_CC_INSTRUCTIONS] Canadian Centre for Cyber Security, Canadian Common Criteria Program Instructions, Version 2.3, 8 May 2025. Available at cyber.gc.ca.
- [TLS_FP] Functional Package for Transport Layer Security (TLS), Version 2.1, 2025-08-25. Available at commoncriteria.github.io/tls.
- [X509_FP] Functional Package for X.509 Certificates, Version 1.0. Available at commoncriteria.github.io/X509.
- [FIPS_140_3] Security Requirements for Cryptographic Modules, FIPS PUB 140-3, March 2019. Available at nist.gov.
- [NIAP_APP_PP] Protection Profile for Application Software, Version 2.0, 16 June 2025. Available at niap-ccevs.org.
- [CNSA_2] Commercial National Security Algorithm Suite 2.0 Cybersecurity Advisory, PP-22-1338, Version 1.0, September 2022, and CNSA 2.0 FAQ, PP-24-4014, Version 2.1, December 2024.

Available from nsa.gov and nsa.gov.

- [DBMS_MOD_CRYPTO] collaborative PP-Module for DBMS Cryptographic Functions (DBMS_MOD_CRYPTO), Version 0.5, 2026-07-20.
- [cPP_DBMS] collaborative Protection Profile for Database Management Systems, Version 2.0, 27 April 2026.
- [cPP_DBMS_SD] Supporting Document Mandatory Technical Document Evaluation Activities for the collaborative Protection Profile for Database Management Systems, Version 2.0, 27 April 2026.
- [DBMS_Cloud_MOD] collaborative PP-Module for DBMS in the Cloud (DBMS_Cloud_MOD), Version 0.5, 2026-07-20.
- [DBMS_Cloud_SD] Supporting Document for collaborative PP-Module for DBMS in the Cloud, Version 0.5.