Payment Card Industry (PCI)

PIN Transaction Security (PTS) Hardware Security Module (HSM)

Modular Derived Test Requirements

Version 4.0

December 2021

Document Changes

Date	Version	Author	Description
April 2009	1.0	PCI	Initial release
February 2012	2.x	PCI	RFC modifications for consistency with PCI POI requirements
May 2012	2.0	PCI	Public release
February 2016	3.x	PCI	RFC version
June 2016	3.0	PCI	Addition of approval classes for key-loading devices and HSM remote administration platforms. Added Device Management. Significant enhancements to existing DTRs. See *PCI PTS HSM - Summary of Requirements Changes from Version 2.0 to 3.0.*
December 2021	4.0	PCI	Added new module, "Cloud-Based HSMs as a Service – Multi-tenant Usage Security Requirements."

Contents

Hardware Security Module Overview

These requirements cover the manufacturing, distribution, and implementation of hardware security modules, or HSMs. The specifics of user key management and operation are covered separately within documents such as the PCI PIN or PCI P2PE requirements.

As outlined in the testing requirements in this document, an HSM is a solution that provides for the security of cryptographic key storage and operation through a combination of hardware and software security features. An HSM may also be considered a secure cryptographic device (SCD) or cryptographic module under the definitions of ISO13491-1 or FIPS140-2 / FIPS140-3, although a solution meeting the requirements of those standards is not necessarily guaranteed to meet all applicable requirements within this document.

For the purposes of the requirements outlined herein, it is considered that any HSM Solution must, at a minimum, provide for the following fundamental security tenets. An HSM Solution must:

1. Protect the confidentiality of secret and private cryptographic keys.
2. Protect the integrity of public cryptographic keys.
3. Permit and facilitate the use of unique secret and private keys per device.
4. Protect and enforce the properties—such as purpose, algorithm, and length—of cryptographic keys.
5. Provide for the maintenance of cryptographic-key “ownership” and operational functionality, so that only a key owner is able to perform cryptographic operations using that key or define what operations may be performed using those keys.
6. Ensure the confidentiality, integrity, and overall security of cryptographic operations, so that these operations do not leak sensitive data, or permit for the operations themselves to be manipulated.
7. Provide for proof of existence and correct operation, including validation that it is in a state where it continues to maintain these tenets as listed above—i.e., it has not been subject to attack.

The simplest HSM implementation protects a single cryptographic key and requires physical access in secured facilities to protect the loading and operational use of that key. This one key is stored in clear text within the secure boundary of the HSM and is never output from this boundary.

In reality, it is expected that many HSM Solutions will be more complex than this, allowing for the storage and use of more than one cryptographic “working” key, as well as using additional cryptographic controls—with their own associated cryptographic keys and key management controls—to protect the loading, storage, operation, and destruction of these working keys.

In such implementations it is common for some subset—and potentially all—of the working keys to be protected through encryption under another key. This key may be known as the “storage master,” “local master,” “master file,” or “housekeeping” key—or by many other potential names. In this standard we refer to this key as the “tamper key.” However this key is named, its purpose is to allow for the tamper boundary of the HSM to be focused on this key alone, and the operations performed with the data hierarchies upon which that key operates.

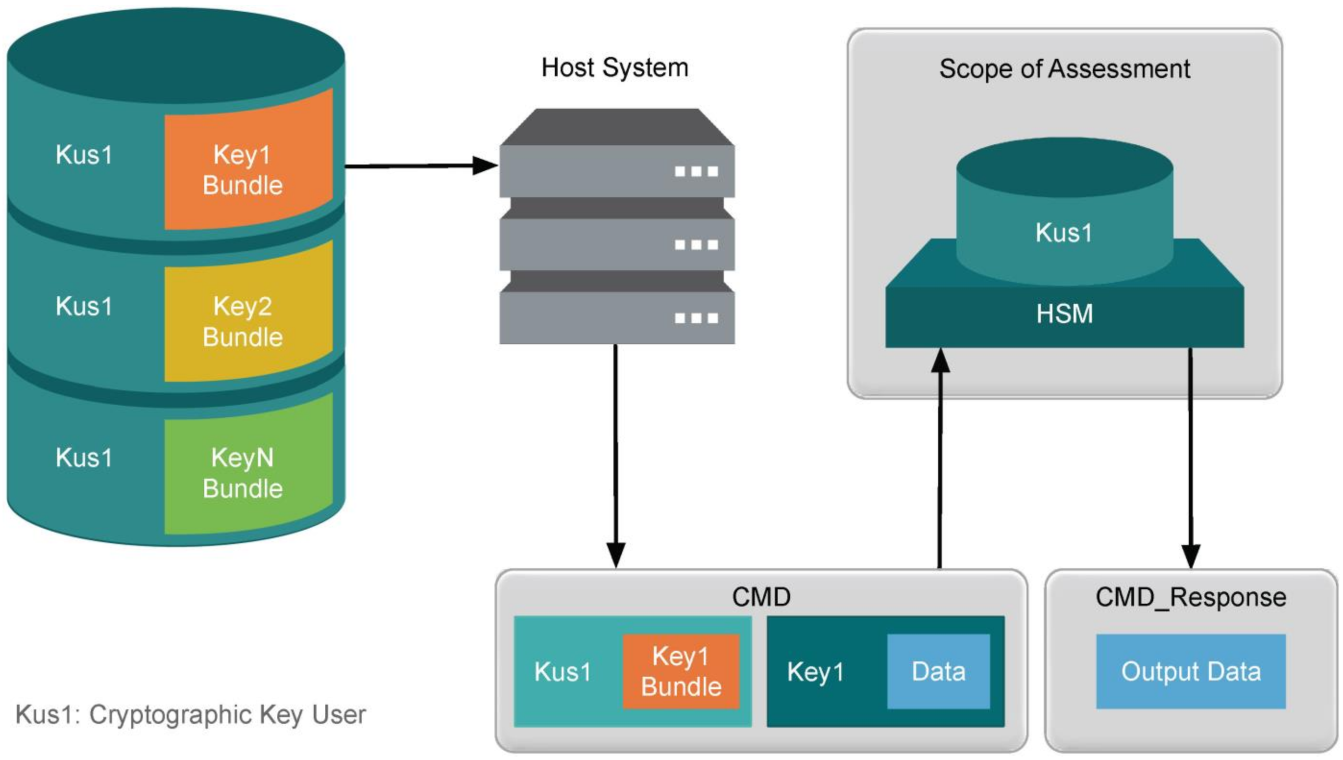
Encryption under this key allows for other keys to be stored external to the tamper boundary of the HSM, with any attack on the HSM resulting in the erasure of the tamper key, thereby rendering all keys encrypted with this unusable and unrecoverable—even if they are stored outside the HSM itself.

Encryption of working key values alone is not sufficient to provide for all of the security tenets required by an HSM, metadata about the key such as the algorithm the key is to be used with, the purpose of the key, and the length of the key, must also be secured along with the key value itself. This is achieved through "key wrapping," which takes the key and key metadata and "wraps" that into a bundle that protects the confidentiality of the key along with the integrity and authenticity of its metadata.

Within these requirements, key wrapping is mandated for all key-storage methods where the key may be exposed outside of the HSM boundary. Keys stored within the HSM that are protected by the physical and logical security properties of the HSM itself may not require direct encryption during storage but will always require that their metadata is securely maintained.

An example of an HSM Solution utilizing the external storage of working keys, which may be considered for assessment under these requirements is provided below, with the scope of assessment noted. This is an example only, and many other implementation methods are possible.

Figure 1: Example of HSM Solution using External Storage

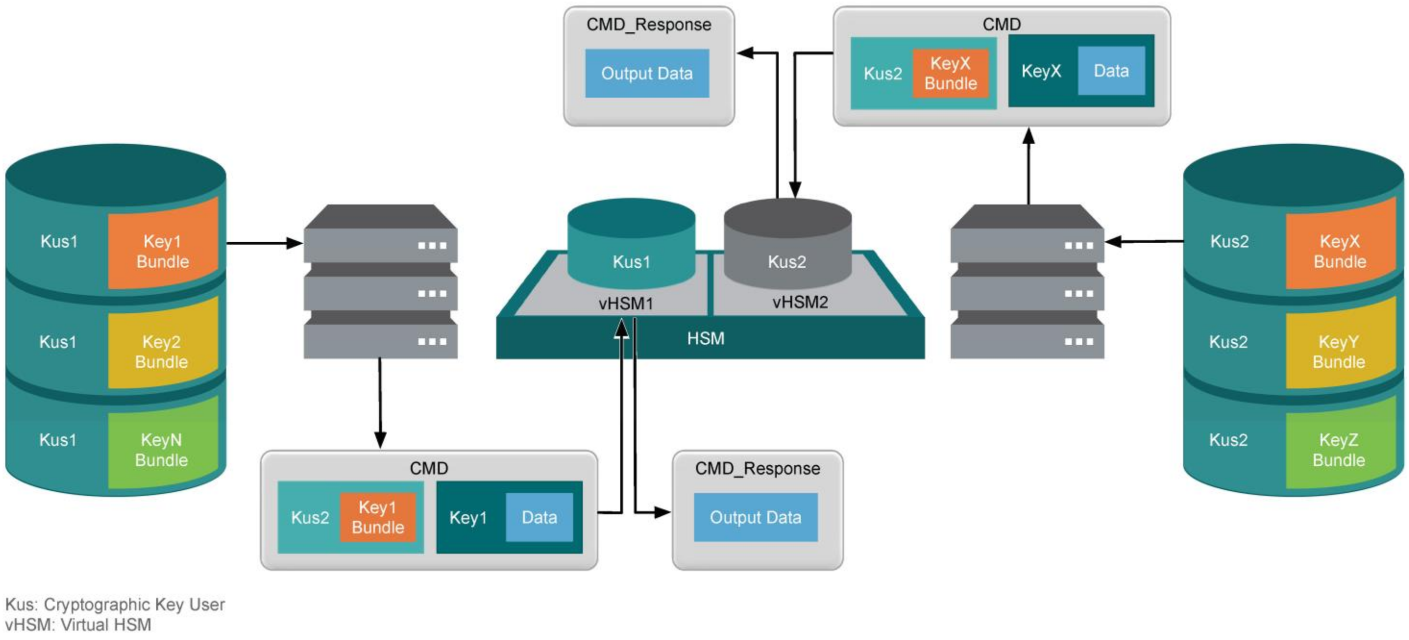


In many traditional HSM implementations, it is common for at least some of the security tenets—such as the requirement for proof of existence, or key ownership—to be provided by the physical possession of the HSM itself. One example of this is through the HSM user having direct control and ownership of the HSM tamper key. However, this is not always ideal or even possible, so there are no mandates within this standard that require that the tamper key used to extend the tamper-responsive envelope of the HSM be directly owned and managed by the HSM user.

This is particularly important for Cloud-based HSM Solutions, which are covered for the first time within this version of the standard. In these implementations, the HSM user is not the same entity that is deploying and managing the HSM Solution being used, and there may be multiple users present and operating on any one Cloud-based HSM Solution.

One way to implement multi-user operation in a Cloud-based HSM Solution is to provide for two or more “virtual” HSMs within a single physical HSM instance, with each virtual HSM having its own tamper key, controlled by the HSM Solution Consumer. An example of such an implementation is illustrated below.

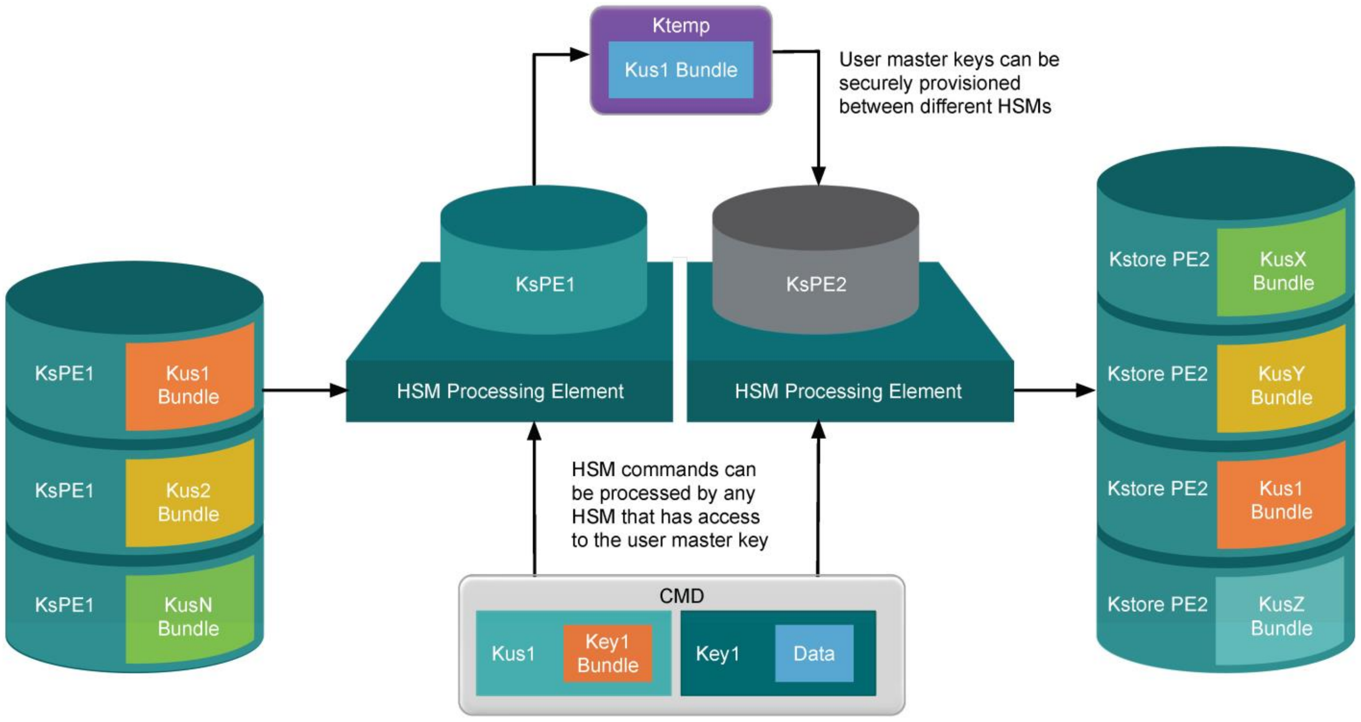
Figure 2: Example of Cloud-Based HSM Solution



Another possible implementation is to disconnect the tamper key from the user master key, so that the HSM Solution stores the master keys of the HSM Solution Consumers in an encrypted form, the same way the working keys are themselves stored encrypted under the HSM Solution Consumer master key. In either implementation, the requirements for protecting the ownership and security of the user keys remains, and it is expected that any HSM tamper key—regardless of whether it is directly managed by the user—is unique per HSM and does not facilitate the exposure or unauthorized use of the user keys it protects.

An illustration of an HSM Solution that implements separate storage and user master keys is provided below.

Figure 3: Example of HSM Solution using Separate Storage and User Master Keys



Kus: Cryptographic Key User
KsPE: Cryptographic Key Storage Processing Element
Ktemp: Ephemeral Key
vHSM: Virtual HSM

It is not the intent of these security requirements to dictate or favor any particular HSM design or implementation. However, it is important when reading this document to be aware that while traditional implementations are not prevented, new implementations where common usage methods are altered may also be possible.

Reporting Requirements for PTS Laboratories

To be acceptable for reviewing, evaluation reports must present evidence of a device's compliance to the Security Requirements. Before releasing a new test report, a delta report, or any updated report version, the lab must perform a thorough technical and quality assurance review, to ensure that the report:

- Accurately provides information specified in this document, without ambiguous or inconsistent information;
- Provides complete and accurate device details;
- Includes information relevant to any applicable FAQs; and
- Conforms to Laboratory General Requirements and any other related documentation in the PTS Program, such as (but not restricted to) Reporting Guidance and Template.

Minimal Contents of Reports and Minimal Test Activities

All reports shall include a device summary section at the beginning of the document. This summary shall include the following:

- A device overview that summarizes the device's design, hardware and software architectures, functionalities, and any other security-relevant attributes, features, or functions including (but not restricted to):
    - Security processor(s) and other processors and memory, operating system(s), boot-up sequences, firmware modules, software applications, crypto functions, data-loading mechanisms, hardware versions and software versions with explanations of versioning, features and functions associated with the device's approval, etc.
    - This overview must present all security-relevant features necessary to derive assets, threats, and attacks relevant to the Security Requirements, as follows:
        - Photographs of the device from different perspectives, sufficient for all sides of the evaluated device to be clearly seen.
        - A list or table of the device's features and functions.
        - A list or table of the device's security-related assets relevant to applicable DTRs.
        - A list or table of the device's principal physical components along with a photograph of the components, disassembled, indicating each of the components.
        - Illustrations/descriptions of the device's logical architecture, interfaces, and key management, to indicate the hierarchy/relationships of firmware and other logical attributes of the device.
        - Block diagram(s) indicating the hardware evaluation boundary with regard to the device's overall architecture and peripheral interfaces.
        - If applicable, a diagram or description the device's integration into other architectures and/or integrating components.

  - Clear definitions of which hardware and firmware features of the overall device solution are within or without the scope of evaluation.
- A summary list of DTRs with verdicts on whether tested and whether compliant; and
- A summary of any assistance provided by the vendor to the lab.

In support of some test steps, as directed by the test laboratory, the vendor must support the laboratory in various tasks (such as, but not restricted to code review, fuzzing interfacing, DPA, etc.) to avoid prohibitively lengthy test activities.

The vendor shall make all source code available to the lab and provide assistance to make a systematic review of relevant security functions.

Evidence-based reporting, demonstrating device compliance through robust testing, is the fundamental basis for achieving device approval. For all DTRs, the tester shall provide the following in writing:

- At the beginning of the report, a list of all documentation, including DTR sections and Technical FAQs version used during the evaluation
- Throughout the report, inline references to specific documents when addressing other sub-requirements
- For each DTR, the DTR definition and guidance (if any), followed by the sequence of all DTR work items (e.g., TA1.1, TA1.2, TA1.3, etc.), directly preceding the report's explanations on each item. Cited DTR text shall match the current DTRs version and shall be clearly distinguishable from Laboratory-generated report text. "N/A" verdicts shall be clearly identifiable as such.

The evaluation report document shall demonstrate compliance to Security Requirements and shall present all test evidence as requested within individual DTRs. For all DTRs, the tester shall present sufficient information on direct tests and theoretical claims to validate conclusions by demonstrating how any conclusions are derived. The tester should use his or her own judgment in determining the appropriate tests and shall document why the test evidence and methods used are valid against PTS Program—i.e., considering DTRs, FAQs, Program Guide, and any other related documents. Every DTR should be supported by sufficient evidence for the evaluation conclusions placed in the report to be understood and confirmed. This includes (but is not limited to):

- References to relevant information in the overview sections of the evaluation report, and to other DTRs where appropriate;
- Descriptions of the vendor's attestations of compliance to security requirements, with:
  - Descriptions of information and assistance provided by the vendor to support the evaluation;
  - Accurate descriptions of relevant device attributes, for example (but not restricted to): physical and logical protections, chip architecture, OS, etc.
  - Detailed explanations of the scope and focus of test activities and attack hypotheses, including explanations of white-box or black-box approaches used, and why;
  - Details of decisions made for performing penetration testing, the methodologies used, and the results of penetration testing.
- Proofs of the reliability of testing reused between devices having similar characteristics;
- Justifications for any reliance on test evidence not derived directly from the evaluation activities;

- Justifications for any reliance on test evidence derived from devices other than the device under test;
- Explanations for any conclusions based on theoretical analysis instead of applied testing.

The tester shall detail where costing information is based on testing or assumptions and provide justification for any use of assumptions rather than actual testing.

The tester shall justify any deviation from the prescribed routines. The tester is not limited to only presenting information specified by DTR text/guidance/FAQs/Program Guide. Where necessary to support a conclusion, the tester shall expand upon that information.

In most cases the DTR text is insufficient without combining photographs and/or other graphic illustrations that explain the evaluation. Images shall be of sufficient quality for relevant details to be viewed—for example, clear identification of a hardware component, relevant information clearly discernible in a graph, images capturing displays and other outputs, source-code fragments, etc.

All DTRs must include references to documents and any other relevant sources of information upon which the evaluation relies. References must indicate information sources sufficiently to enable PCI to identify test evidence following device approval.

Where test evidence from prior evaluations is being reused, the similarities and differences between the prior and current devices must be detailed. Evidence of how the similarities were confirmed must also be presented.

Clear indications to the reader must be provided in any DTR where results have been reproduced from a prior report.

Re-use of test results for PTS HSM v4.0 evaluations is only allowed where the evidence is no older than two years—other v4.0 evaluation work may supplement work done in the current review.

Asset Flow Analysis

Guidance

The purpose of the Asset Flow Analysis is to describe in block-diagram form how assets travel within the device (both logically and physically) and are protected as they are processed and manipulated by it. The Asset Flow Analysis does not have to be provided as a single flow diagram. It can be made up from several flow diagrams so long as it is clear how each flow diagram is interconnected and/or interrelated.

Within the Asset Flow Analysis, appropriate domains are assigned for each logical and physical component in the device including software modules, hardware components, and PCB tracks (which may be grouped if several tracks combine a bus). The tester then uses this Asset Flow Analysis to scope the device and apply the appropriate DTRs for the specific domain.

Requirement

The vendor shall provide an Asset Flow Analysis highlighting each physical and logical component in the device and indicating how each asset flows between each physical component, showing the logical modules used to protect it. The general idea is to indicate the status of an asset as it travels through the device—for example, whether the asset is clear text or encrypted at a point in the data flow. Any hardware component or software module interfacing with the asset will be virtually marked (or tagged) with the domain that the asset belongs to as defined in Appendix F, "Domain-Based Asset Flow Analysis."

The Asset Flow Analysis shall be included in the PCI HSM report and referenced in the appropriate DTR evaluations throughout. It is therefore expected that the vendor will perform this domain-based Asset Flow Analysis and provide the results and a complete explanation to the testers. The test lab will verify the analysis and use the effective domain rating as direction for the device evaluation.

It is important that asset flows and domains are correct and complete for each asset. The lab will validate and notify the vendor if discrepancies are discovered in the asset flows for corrective action. The lab will also validate whether any asset containers correctly protect the asset and that all required cryptographic keys are listed in the asset flow analysis.

DTR Module 1: Core Requirements

A – Physical Derived Test Requirements

DTR A1 Tamper-Detection Mechanisms

The device uses tamper-detection and response mechanisms that cause it to become immediately inoperable and result in the automatic and immediate erasure of any sensitive data that may be stored in the device, such that it becomes infeasible to recover the sensitive data. These mechanisms protect against physical penetration of the device. There is no demonstrable way to disable or defeat the mechanisms and access internal areas containing sensitive information without requiring an attack potential of at least 26 per device for identification and initial exploitation, with a minimum of 13 for initial exploitation, as defined in Appendix A.

Guidance

Areas of the device that contain sensitive information persistently or during the normal operation of the device must be protected via tamper-detection mechanisms.

"Immediately inoperable" is defined as immediately ceasing all processing activities involving secure key materials and critical security parameters. Diagnostic functionality such as status information and audit trails may continue to be available following the stoppage of processing activities. The user may have access to utilities to recover the device to an operative state. Additionally, the tamper-detection and response mechanisms must result in the automatic and immediate erasure of any sensitive data that may be stored in the device, such that it becomes infeasible to recover the sensitive data.

"Immediate" is defined as fast enough to ensure that erasure occurs before the tamper-detection mechanisms can be disabled using attack methods described in A1.

For those devices that do not contain secret information, device disablement may be used in lieu of "immediate erasure of all secret information."

"Secret information" is any private or secret cryptographic keys or passwords/authentication codes that the device relies on to maintain security characteristics governed by PCI requirements. If any of this secret information is not zeroized, other mechanisms must exist to disable the device.

The device is protected against penetration by including features that detect: (i) any feasible attempts to tamper with the device; and (ii) cause immediate erasure of all cryptographic keys and sensitive data when such an attempt is detected.

Removal of the case or the opening, whether authorized or unauthorized, of any access entry to the device's internal components causes the automatic and immediate erasure of the cryptographic keys stored within the device.

Any tamper-detection/key-erasure mechanisms function even in the absence of applied power.

(continued on next page)

If any of these keys is not zeroized, other mechanisms must exist to disable the device, and these keys must be protected in accordance with Requirement A4.

Secret or private cryptographic keys that are never used to encrypt or decrypt data, or are not used for authentication, do not need to be considered secret data and therefore do not need to be erased—for example, where the device uses a chip set that automatically generates keys at initialization but the keys are not subsequently used by the device.

The vendor shall provide documentation of test results for inspections of internal buffers.

If tamper grids are used as a primary mechanism, they meet the following:

- *Use a minimum of two layers of internal grids for protection.*
- *Vias of "upper grid" must be protected separately to vias of "lower grid"—for example, the two tamper grids must not be connected by vias that are accessible on both grid layers, or vias must be protected by other tamper mechanisms, such as switches.*
- *Maximum width/separation (of active traces) must be 6 mil.*
- *Use "opposing" tamper-responsive traces routed side-by-side on each layer.*

TA1.1 The tester shall examine the Asset Flow Analysis to verify that it is complete and accurate and that it allows to unambiguously map all hardware components—including PCB tracks, passive components, plastics, etc.—to a security domain.

TA1.2 The tester shall provide an overview of the device and how it is constructed. The tester shall include an exploded diagram of the device showing how all sub-components are assembled and connected internally—for example, an explanation of processor (secure and unsecure) architectures and where these are located with regard to the internal areas of the device.

TA1.3 The tester shall attempt to penetrate or open the device to activate the tamper-detection mechanisms and then perform tests to support evidence that the mechanisms cause the immediate and automatic erasure or non-recoverability of any secret data contained in the device. Tests that may be performed could include attempting a transaction to determine whether the transaction fails, using a special function of the device that allows a user to determine the status of secret data, or using special software to determine whether secret data has been erased.

TA1.4 The tester shall describe how the device responds to a tamper-detection event, including any mechanisms for how a tamper event is indicated.

TA1.5 Deriving from the previous descriptions, the tester shall explain how the immediate and complete erasure of all sensitive information from the device results from tamper-detection events.

TA1.6 The tester shall verify that attempts to probe critical security circuitry through any means other than opening the case (such as through vents, fans, or holes) do not allow probing access without triggering the tamper-detection mechanisms.

TA1.7 If a cover is part of the tamper-detection mechanism, the tester shall verify that attempts to remove the cover by removing fasteners, plates, connectors, etc., or by creating a gap between the covers or cover and housing do not allow access to probe critical security circuitry without triggering the tamper-detection mechanisms.

TA1.8 If a cover is part of the tamper-detection mechanism, the tester shall attempt to remove the cover by disabling or defeating the tamper-detection mechanisms. To remove the cover, the tester may use any attack method. The tester may perform any test needed to validate the attack scenario. The tester will use his or her own judgment in determining the appropriate tests and whether the attack will be performed in its entirety or in part to verify the theory. If an attack scenario can be developed that yields an attack potential of less than 26 per device for identification and initial exploitation, or an initial exploitation value of less than 13, as defined in Appendix A, the vendor assertion cannot be verified.

If the device is restricted to deployment in an environment meeting at least the security of a controlled environment, the following applies: To remove the cover, the tester may open, pry, or disassemble the device at cover seams and remove other plates, connectors, etc., to gain access to the tamper-detection mechanisms. Removing shall not consist of drilling, milling, burning, melting, grinding, or dissolving the cover or enclosure. The tester may drill out visible fasteners (e.g., screws, rivets, press-fittings, etc.) to remove the cover.

TA1.9 The tester shall examine vendor documentation relating to the response of the device to tamper detection for consistency with device functionality and documentation, including the key table in the vendor security policy. Any sensitive information that is not erased must be encrypted using accepted algorithms and key sizes according to the table presented in Appendix D.

TA1.10 The tester shall examine vendor-supplied documentation to determine whether the device employs active or passive (i.e., removal of power) erasure. If the device employs passive erasure, the tester shall verify that erasure occurs rapidly enough to prevent an attacker from opening the device and stopping erasure before it is effective. The tester may create an attack scenario, which may be performed in its entirety or in part to verify the theory.

TA1.11 The tester shall examine any relevant documentation submitted by the vendor, including vendor test results for inspections of internal buffers, to verify that it supports the vendor responses.

TA1.12 The tester shall enumerate each of the circuit boards indicated in the device in the table below, providing, at a minimum:

a) The PCB designator (name) used;

b) The version of the PCB;

c) The main purpose of the PCB;

d) Pictures of the front and back of each PCB and references thereto;

e) Note as to whether the PCB contains any sensitive signals (such as clear-text PINs or keying material—but not tamper signals); and, finally,

f) Outline of the tamper-detection mechanisms used on that board—such as “4 tamper switches on lower face,” “6 tamper switches on upper face,” “two internal tamper grids,” etc.

The tester shall adapt the table (for example, by adding columns or additional notes) as necessary, to present any additional information.

PCB Designator	PCB Version	PCB Purpose	Picture Reference	Sensitive Signals	Tamper-Detection Mechanisms

TA1.13 For each PCB that carries sensitive data, the tester shall describe what tamper-detection mechanisms protect these signals from being accessed (such as tamper grids). The tester shall confirm that these mechanisms protect the entire path taken by the signals, as described above.

TA1.14 If the HSM has a keypad, the tester shall describe whether any of the items on the path of the keypad signals are not protected by all tamper-detection mechanism. For example, the tester shall note whether a signal via terminates on the same layer as a tamper grid and if any passive components are located outside of the protected area or are connected to vias (including power vias) that terminate outside of the protected area(s).

TA1.15 Using vendor documentation for each tamper grid that is implemented, the tester shall complete the details indicated in the table below, describing, at a minimum:

a) The location of the tamper grid (PCB designation and layer);

b) Its physical implementation—that is, its composition—for example, copper on a rigid PCB, conductive ink on plastic, etc.;

c) The size of the conductive traces and the distance between each tamper-detecting trace (not necessarily between each trace) as well as the distance between layers for tamper grids that provide protection against penetration through the side of the PCB;

d) The method used to connect to the tamper grid—such as through hole via, buried via, soldered connection, or elastomeric strip;

e) Whether traces from the same tamper signal are routed immediately adjacent to one-another without another tamper signal or passive signal interspacing them.

The tester shall adapt the table (for example, by adding columns or additional notes) as necessary, to present any additional information.

Tamper Grid Location	Physical Implementation	Size of Traces and Distance between Traces, Signals, or Layers	Number of Tamper-detecting Signals	Method of Connection	Adjacent Signals?

TA1.16 The tester shall describe what testing was performed to validate the protection provided by each of the tamper grids enumerated above.

TA1.17 For each tamper switch used in the device, the tester shall complete the details indicated in the table below, at a minimum.

The tester shall use the ”Additional Comments” column to note any unusual features the tamper switch may possess that make it easier or harder to attack—such as being covered by a flexible tamper grid or having a unique construction.

Switch Location	Number Used in that Location	Physical Implementation	Size of Switch Contacts	Conductive Ink Protections	Additional Comments

TA1.18 The tester shall describe what testing was performed to validate the protection provided by each of the tamper switches enumerated above.

TA1.19 The tester shall note which tamper-detection mechanisms use active high, active low, dynamic, resistive, or other types of sensors. The tester shall confirm that any guard rings or interspersed traces in tamper grids are at opposing voltages that will activate tamper detection if electronically shorted. The tester shall note what testing has been performed to confirm this.

TA1.20 The tester shall describe any volume-encapsulation methods used in the device—for example, epoxy filling—that are designed to make penetration or reverse engineering more difficult.

TA1.21 The tester shall describe what testing was performed to validate any volume encapsulation. Testing must include chemical and/or abrasive, and heating methods to bypass this protection.

TA1.22 The tester shall describe any attachment or “forming” methods (such as soldering, elastomeric strips, or adhesive for attachment, and plastic/metal walls for forming the shape of flexible circuits) used as part of the security features of the device. For example, the tester shall detail the methods used to secure any flexible tamper grids or “cover PCBs” so they cannot be bent or lifted out of the way.

TA1.23 The tester shall describe what testing was performed to validate any attachment and forming methods. Methods of testing must include use of localized heating, solvents, and abrasion. The tester shall justify why the testing performed was sufficient and why the security measures cannot be bent, melted, or otherwise bypassed, to gain access to sensitive signals.

TA1.24 The tester shall provide details on the security processor used in the device, including how it drives tamper-detection features. The tester shall provide and reference picture(s) of the location(s) and area(s) surrounding processor(s) used by the device, specifying those that are security-relevant.

TA1.25 The tester shall explain how the device is protected against attacks from all sides of the device, including the front, rear, top, and bottom of the device, and any other device sides.

TA1.26 The tester shall describe any device security features used to protect sensitive data that have not already been covered in the previous descriptions (for example, special processor packaging). The tester shall detail what testing has been performed to validate each of these features.

TA1.27 The tester shall provide and make reference to a schematic diagram of the tamper circuits of the device, showing connections to all tamper-detection features including switches and tamper grids.

TA1.28 The tester shall state how any passive components, connectors, or other items that carry tamper signals are protected against being accessed. The tester shall include any connections to power planes through hole vias that may be exposed outside of the tamper-detecting areas of the device.

TA1.29 From the above descriptions the tester shall explain how the device is rendered inoperative after any tamper event.

TA1.30 Describe the different attack paths considered. Using the format shown in Appendix A the tester shall generate attack costings, using different attack techniques on the device, and present the two most feasible. The attacks should be dissimilar in approach unless the lab can fully justify the infeasibility of any second divergent approach. The tester shall state explicitly where testing has verified any specific stage of the attack (including the time, equipment, and skill required, and number of mechanisms to bypass), and where assumptions are used in place of testing. The tester shall justify why any assumptions have been used instead of than actual testing. Calculations shall include evidence justifying particular rating levels as being appropriate.

If the device is restricted to deployment in and environment meeting at least the security of a controlled environment, the following applies: The tester may drill out visible fasteners (e.g., screws, rivets, press-fittings, etc.) to remove the cover or to create a gap between the covers or cover and housing to insert probes. However removal shall not consist of drilling, milling, burning, melting, grinding, or dissolving the cover or enclosure.

TA1.31 The tester shall verify that no attack was able to be determined, including those outlined above, which could be performed for less than 26 points in total, less than 13 points during initial exploitation.

DTR A2 Robustness Under Changing Environmental and Operational Conditions

The security of the device is not compromised by altering environmental conditions or operational conditions (for example, subjecting the device to temperatures or operating voltages outside the stated operating ranges).

Guidance

In order to minimize the destructive testing required, the tester may use the following to generate the assessment: review test results provided by the vendor, utilize subsections of the module for destructive testing (e.g., an epoxy sample), or analyze individual components apart from the device.

The vendor must either provide substantive data to support the security of the product outside normal operating conditions (if test results and/or testing on subsections/components are used to verify the security) or show that the product uses sensors that will trigger a tamper response.

The objective is not to replicate the vendor testing, but instead it is to account for shortcomings within the vendor's testing of the implementation.

The tester may rely upon vendor testing as appropriate to fulfill the following test steps.

TA2.1 The tester shall provide the operational temperature and voltage range for the device as detailed in vendor documentation.

TA2.2 Using the schematics and descriptions from TA1.2, TA1.12, TA1.15 and TA1.17, the tester shall list the temperature ranges for all components included in the tamper-detection circuit. This shall include mechanical switches and active elements (but not passive elements such as resistors and capacitors).

TA2.3 The tester shall use the table below to detail the environmental protection features implemented by the device. For voltage, the tester shall outline which voltage is being monitored—for example, external, main processor core voltage, battery voltage, etc. If more than one voltage monitoring is provided, the tester shall detail all of these. For each environmental factor monitored, the tester shall detail what circuitry is performing this monitoring and what occurs when during an out-of-range detection.

	Maximum Value	**Minimum Value**	**Detecting Circuitry**	**Response**
Voltage (Specify type)	Configured Value	Configured Value		
	Tested Value	Tested Value		
Temperature	Configured Value	Configured Value		
	Tested Value	Tested Value		

TA2.4 In the maximum/minimum values for each item, the tester shall note what the vendor attests the value is set to in the "Configured Value" cells of the above table.

TA2.5 Using a device that has been configured—using special test code from the vendor, which shall be removed from production units—to operate self-tests such that the correct operation of the device can be confirmed, the tester shall test each of the environmental features listed above and enter the value at which the detection circuitry activates into the "Tested Value" cells of the above table.

TA2.6 The tester shall detail whether the self-test program used above is executed correctly at all times during each of the tests above—within the vendor's stated minimum and maximum voltage and temperature ranges for the device under evaluation—before activation of the environmental detection circuitry.

TA2.7 Given the details and results above, the tester shall justify why the tamper-detection mechanisms will remain functional and the device secure at all extremes within the range of environmental monitoring.

TA2.8 The tester shall develop attack scenarios to compromise the device by altering operational conditions and consider whether altering environmental conditions may be used to develop attacks.

TA2.9 The tester may perform any test needed to validate the attack scenario**.** The tester shall present sufficient evidence and/or references for the above, for demonstrating compliance to this DTR.

DTR A3 Protection of Sensitive Functions or Information

Sensitive functions or information are only used in the protected area(s) of the device. Sensitive information and functions dealing with sensitive information are protected from unauthorized modification or substitution, without requiring an attack potential of at least 26 per device for identification and initial exploitation, with a minimum of 13 for initial exploitation, as defined in Appendix A.

Guidance

Public keys used for functions that impact security requirements, such as firmware updates or remote key-distribution schemes, must be protected against modification or substitution. Secret and private keys used for functions that impact security requirements must be protected against modification, substitution, or disclosure.

The protected area of the device is that area(s) within the boundaries of the tamper-detection and response mechanisms.

The lab shall consider glitch attacks including (but not restricted to): voltage and EM glitching. At a minimum, these should consider the core and battery input for the security processor.

TA3.1 The tester shall verify the completeness of the information regarding sensitive information and functions presented by the vendor.

TA3.2 In the following table, the tester shall outline the locations of all types of sensitive information and functions, adding to those provided where other types of sensitive information exist within the device. This shall include both long-term and temporary storage locations, as well as information on any programmable logic used in the device as part of the PIN processing circuit. The Storage Area column shall outline where and what type of storage is used for this information (such as internal SRAM of the security processor, or external Flash memory); and the Method of Protection column should outline what mechanisms of the device design protect this information (such as encryption, signature(s), physical protection, etc.). The Method of Protection description should be explicit about this protection, detailing specifically which cryptographic keys are used for encryption, for example, or which physical protection mechanisms are utilized.

It is expected that each type of information may have multiple "storage areas" and "methods of protection" (for example, clear-text PINs may appear in internal memory of the security processor, as well as on the external heap cache and within the processor registers, and the external memory may be protected by encryption with the external bus key as well as with physical protections of the secure area of the device).

The tester shall adapt the table (for example, by adding columns or additional notes) as necessary, to present any additional information. The tester shall reference the relevant aspects of the asset flow.

Sensitive Information	Storage Area	Method of Protection
Clear-text PINs		
Passwords/authentication codes		
Device Firmware		
Public keys		

TA3.3 Using the information from the table above, the tester shall provide details on the different integrated circuits—memory, processors, programmable logic etc.—that are used to store, process, or secure sensitive information. The tester shall reference the relevant aspects of the asset flow.

TA3.4 For each of the integrated circuit elements (described above) that may be programmed or configured in some way, the tester shall enumerate the following:

- The different ways in which that element may be programmed or configured (for example, JTAG).
- Any in-circuit testing or debugging features provided by these elements.

TA3.5 The tester shall detail what methods have been implemented to disable all of the programming/testing features outlined above. The tester shall detail the testing performed to validate that these features are indeed disabled. The tester shall justify why these measures are sufficient and confirm that these features cannot be re-enabled.

TA3.6 If additional memory is implemented and is not included in the sensitive-information storage areas above, the tester shall detail what processes have been used to validate that this is the case. The tester shall detail all memory in the device and detail where sensitive data is stored and how it is protected.

TA3.7 If the device allows for the execution of applications and firmware on the same processor that stores or operates on clear-text passwords/authentication codes, PINs, or public keys, the tester shall note what mechanisms are implemented to prevent these applications from modifying this information. The tester shall detail how this has been validated as sufficient. The tester shall reference the relevant aspects of the asset flow.

TA3.8 Where signatures are used as a method of protection, the tester shall:

a) Validate that only approved algorithms and key lengths are used for the signatures.

b) Detail what padding scheme is used for the signatures and justify how this prevents attacks such as padding oracle attacks.

c) Detail when the signatures are validated, and how modification of the sensitive information is prevented after signature validation.

TA3.9 Where physical protections are used as a method of protection (for example, when clear-text information is stored in external memory), the tester shall:

a) Confirm that the physical protections cover all memory traces, vias, passive elements, or other areas of access. For example, detail whether the vias of the memory bus appear on the same layer as one of the tamper grids.

b) Detail how the memory packages are protected, including access to BGA balls and traces on internal chip carriers of packages.

TA3.10 Where encryption is used as a method of protection of sensitive information, the tester shall:

a) Confirm that the encryption uses approved algorithms and key lengths.

b) Note what mode of operation is used for the encryption.

c) Justify how this prevents the re-location of memory from one area to another.

d) Justify how the method of encryption prevents the exposure of sensitive information through building of a "dictionary" of possible encrypted values.

e) If a key stream mode of encryption is used (for example, OFB), justify how the encryption of different data with the same key is prevented.

TA3.11 The tester shall analyze the device's susceptibility to glitch attacks including, but not restricted to, voltage and EM glitching. At a minimum, the tester shall consider the core and battery input for the security processor. Where applicable, the tester shall also consider embedded memory (SRAM, EEPROM, Flash, and ROM).

TA3.12 The tester shall describe and produce a costing for the most feasible attack to recover sensitive information from the device. The tester shall detail for each step whether the information is based on testing or assumptions and provide justification for any use of assumptions rather than actual testing.

If an attack scenario can be developed that yields an attack potential of less than 26 per device for identification and initial exploitation or less than 13 per device for initial exploitation only, as defined in Appendix A, the vendor assertion cannot be verified. If attack scenarios can only be developed yielding attack potentials above these thresholds, the tester shall present these to demonstrate how the device is compliant to this DTR. At least one attack scenario shall be presented, in a format consistent with Appendix A in this document. Calculations shall include evidence justifying particular rating levels as being appropriate.

If the device is restricted to deployment in an environment meeting at least the security of a controlled environments, the following applies: The tester may drill out visible fasteners (e.g., screws, rivets, or press-fittings) to remove the cover or to create a gap between the covers or cover and housing to insert probes. However, removal shall not consist of drilling, milling, burning, melting, grinding, or dissolving the cover or enclosure.

The tester may perform any test needed to validate the attack scenario.

The tester shall present evidence of the test methodologies followed and the validation results.

DTR A4 Invasive Attacks for Cryptographic Keys

Determination of any PCI-related secret or private cryptographic key resident in or used by the device, by penetration of the device requires an attack potential of at least 35 for identification and initial exploitation with a minimum of 15 for initial exploitation, as defined in Appendix A.

Guidance

"Keys resident in the device or its components" means clear-text secret or private keys. If the encrypted keys are protected in accordance with the minimum key sizes and parameters for the key-encipherment algorithm(s) used as stipulated in Appendix D, they do not need to be considered. For purposes of this requirement, this includes transaction or other security-relevant keys that are only temporarily in the device and are not stored in the device.

Secret or private cryptographic keys that are never used to encrypt or decrypt data (e.g., keys, accounts, PINs), or are not used for authentication, do not need to be considered secret data and therefore are not subject to this requirement.

All physical and logical protections shall be evaluated and reported. Only protections that can be demonstrated as both enabled and efficacious shall be used to determine attack potential.

TA4.1 The tester shall provide details on the type, location, and accessibility, of the security processor(s) used by the device, and any other elements of the device that have relevance to possible attacks. This shall include a list of all barriers obstructing access to keys, beginning with the device exterior case and ending with inner-most barriers, clearly indicating whether or not each barrier is tamper-responsive. The tester shall reference information previously supplied in DTRs A1 and A3 where applicable. The tester shall reference the relevant aspects of the asset flow.n A1.

TA4.2 The tester shall provide details on how cryptographic keys are stored and managed within the device. The tester shall reference this information to the table provided in DTR A3, allowing the location of all applicable keys to be defined. The tester shall detail the testing performed to confirm the storage locations listed are correct. The tester shall reference the relevant aspects of the asset flow.

TA4.3 The tester shall provide details on any specific protections provided by the security processor (or other processors if applicable) that are designed to obstruct obtaining or determining the values of cryptographic keys. If specific protections are unknown, the tester shall provide details on protections strongly inferred through testing.

TA4.4 The tester shall:

a) Describe what protections the cryptographic processing elements implement to protect against glitch attacks to force cryptographic errors.

b) Refer to testing and results from DTR A3 where applicable.

c) Review the source code of the device to confirm that any protection measures relied upon are enabled.

d) Describe why any secure boot-up operations are implemented appropriately to obstruct glitch attacks, referring to any available literature and vulnerability disclosures to support this or otherwise perform practical penetration tests to demonstrate this.

e) Describe why invasive and/or glitch attacks intending to change flags—e.g., flipping a bit value enabling a defense—are effectively obstructed.

TA4.5 The tester shall describe what protections are implemented within the cryptographic processing elements to protect against physical attacks at the chip level to extract the cryptographic keys. The tester shall review the source code of the device to confirm that any protection measures relied upon are enabled and effective. The tester shall also determine whether protections can be disabled and how.

TA4.6 Referring to the information provided in DTR A3, the tester shall perform a review of available literature and vulnerability disclosures to confirm that the programming or in-circuit testing features of the processing elements of the device cannot be re-enabled (either temporarily or permanently). The tester shall validate all documentation provided by the vendor.

TA4.7 If the device stores clear-text cryptographic keys within external memory, the tester shall detail the physical security methods implemented to protect this memory. Note that PCB-based tamper grids are not considered sufficient to protect clear-text cryptographic keys.

TA4.8 The tester shall describe and cost the most feasible attack to recover cryptographic keys from the device, using the above information. The tester shall detail whether steps are based on actual testing or on assumptions and provide justification for any use of assumptions rather than actual testing. This information should include, at minimum:

- The steps needed in any penetration.
- Attacks involving some or all of attacks modeled elsewhere. For example, DTRs A1, A3 and A4 (but not restricted thereto) must be considered and included in this attack, if relevant.

The tester is not required to perform the attack entirely but may perform all or part of the attack to verify its validity. The calculation shall be based on the methodology depicted in Appendix A. If an attack scenario can be developed that yields an attack potential of less than 35 per device for identification and initial exploitation or less than 15 per device for initial exploitation only, as defined in Appendix A, the vendor assertion cannot be verified. At least one attack scenario shall be presented, in a format consistent with the examples shown in DTR Appendix A. Calculations shall include evidence justifying particular rating levels as being appropriate.

If the device is restricted to deployment in an environment meeting at least the security of a controlled environment, the following applies: The tester may drill out visible fasteners (e.g., screws, rivets, or press-fittings) to remove the cover or to create a gap between the covers or cover and housing to insert probes. However, removal shall not consist of drilling, milling, burning, melting, grinding, or dissolving the cover or enclosure.

The tester shall present evidence of the test methodologies followed and the validation results.

TA4.9 If the attack costing for DTR A3 was found to be less than the minimum points required for this DTR, the tester shall justify why the attack for DTR A3 cannot be used to recover cryptographic keys.

DTR A5 Non-Invasive Attacks for Cryptographic Keys

Emanations from the device (including power fluctuations and timing) cannot be feasibly used to recover PIN and/or account data or other PCI security-related cryptographic keys resident in the device.

Guidance

"Keys resident in the device or its components" means clear-text secret or private keys. If the encrypted keys are protected in accordance with the minimum key sizes and parameters for the key-encipherment algorithm(s) used as stipulated in Appendix D, they do not need to be considered. For purposes of this requirement, this includes transaction or other security-relevant keys that are only temporarily in the device and are not stored in the device.

The vendor shall provide mechanisms to facilitate side-channel testing. These mechanisms shall include at least the following: an interface, the ability to vary data and keys, and the ability to set trigger points (for testing purposes only and not for production units), allowing the evaluator to perform analysis with direct access to security-relevant processor(s).

Secret or private cryptographic keys that are never used to encrypt or decrypt data—e.g., keys, accounts, PINs—or are not used for authentication, do not need to be considered secret data and therefore are not subject to this requirement.

SCA tests shall be performed in accordance with Appendix E, including the scope of side-channel testing necessary to validate the device's compliance based on the identification of relevant keys below, and taking into consideration the appropriateness of testing re-use and demonstrably effective countermeasures.

Notwithstanding physical protections, which may be validated by other DTRs, approved devices must not contain cryptographic implementations which, under analysis, can be straightforwardly compromised through SCA.

To pass this requirement, it must be demonstrated that any static-value keys resist SCA attacks by expert-level skill using state-of-the-art tools analyzing at least 100,000 demonstrably high-quality recordings, in-line with the details of this DTR.

***For security-related keys,** the evaluation should determine at least one algorithm to analyze thoroughly, applying the SCA approach most likely to succeed, based on a rigorous assessment of asset value versus feasible attacks. Reasoning for not testing any algorithm must be explained. This reasoning should include reliable assumptions made in the vulnerability analysis based on asset value and attack complexity—for example, limits on collections such as delay insertions or key usage counters, and any additional countermeasures.*

For HSMs restricted to deployment in an environment meeting at least the security of a controlled environment as defined in ISO 13491-2, the evaluation shall perform SEMA and SPA (see Appendix E) at a sufficient depth of investigation to show that cryptographic algorithms processing account-data-protection keys have active and effective SCA countermeasures.

(continued on next page)

Even if physical protections against tampering are robust, and even in the presence of electronic noise and algorithmic complexity obscuring signals, it is expected that an attacker can defeat any undefended cryptographic implementation repeatedly exercising static keys. A device must have at least one demonstrably effective SCA countermeasure protecting all cryptographic algorithms in scope of this requirement. For software implementations and hardware implementations with known countermeasures, the evaluation shall identify one countermeasure, at least, and show its effectiveness. For hardware implementations with unknown countermeasures, the evaluation shall demonstrate the implementation's SCA characteristics are consistent with having at least one effective countermeasure.

As an accelerator and concentration point for cryptographic operations, HSMs are considered especially attractive targets for timing attacks, which may be performed remotely across a network regardless of mitigations implemented through the physical security of the deployed environment. HSMs must implement protections against timing attacks through use of constant time operations, blinding of processed values, etc. Mitigations that rely solely on inherent timing uncertainties of the implementation, such as network jitter, are not considered sufficient.

TA5.1 The tester shall identify and show in a table:

- A list of all security-related cryptographic keys (secret and private) that may be directly or indirectly attacked by side-channel analysis approaches. This list shall indicate whether the most applicable approach is statistical analysis of static keys with variable inputs or outputs, or other approaches such as, but not restricted to, timing analysis.
- Any specific key-management techniques that either prevent or obstruct side-channel analysis, such as unique keys per operation or other constraints on exercising static key values or any other algorithmic constraints on SCA attacks.

The tester shall reference the relevant aspects of the asset flow.

TA5.2 The tester shall check the evidence provided by the vendor on the implemented cryptographic operations and detail all of the different cryptographic operations implemented within the device. The tester shall verify whether they are implemented in software or hardware, and what side-channel analysis protections are implemented to protect each of these operations. The tester shall describe methods used for each side-channel protection—for example, time variance, masking, dual-rail logic, etc. If it is not possible to determine this information from evidence provided by the vendor—for example. if SoC vendor information is proprietary—then evaluation testing should be used to infer this information. Note that any operation that only functions using public keys does not require side-channel protections to be implemented.

TA5.3 Verify the items above, generally using source-code review, to ensure that the side-channel protection methods are implemented. For example, if the device relies on protections provided by the processor hardware cryptographic engine, the tester shall confirm that the registers that enable this protection are correctly set by the device firmware before every use of this cryptographic engine. If the protections are provided by the firmware, the tester shall check that the implementation is as described by the vendor. This evaluation activity may be focused on security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TA5.4 The tester shall perform preliminary side-channel analysis on the device to characterize the cryptographic algorithms used to process sensitive data and/or operate with secret keys. Utilizing characterization results and knowledge of the device physical design and software, the tester shall decide which side-channel attack paths provide the best opportunities for an attacker to compromise high-value assets. The tester shall develop side-channel analysis to explore methodologies most likely to achieve retrieval of sensitive data. The tester shall develop these investigations to derive a demonstrable level of assurance consistent with any reported attack cost rating(s).

The tester shall describe any assistance provided by the vendor to ensure efficient side-channel testing—for example, command scripts, event triggers or access to the processor, etc.

TA5.5 The tester shall justify the methodologies used and findings, in accordance with Appendix E and with regard to published attacks. The tester shall outline why analysis parameters provide a high level of confidence that key recovery through side-channel analysis is not feasible on this device.

Justifications of these to be reported shall include, but are not restricted to:

- Equipment used, and a summary of test parameters such as collection elapsed time and analysis elapsed time,
- Whether the attack can be feasible at any distance away from the processor,
- The number of sample recordings acquired,
- Sampling frequencies,
- Alignment methods,
- Signal analysis / filtering techniques,
- Correlation function(s) used,
- How optimum quality data collection was achieved, etc.,
- How adequate key selection function coverage was applied.

Evidence to support this shall include (but is not restricted to) annotated graphical profiling of side-channel results such as:

- SPA/SEMA characterization,
- Noise-removal test results,
- Alignment test results,
- Demonstrations of non-sensitive data leakage,
- Correlation results for sensitive data leakage modeling,
- Timing analysis,
- DPA/DEMA attacks,
- Results validation and checks.

The tester shall present evidence of the effectiveness of active countermeasures implemented by the vendor in obstructing analysis.

Where no definite sensitive data leakage is achieved, validate why this is the case and provide explanations. For example (but not restricted to), by showing that I/O data leakage successfully localizes sensitive cryptographic operations but key leakage is prevented by demonstrably effective countermeasures.

The evaluation may rely upon appropriate evidence available from existing side-channel evaluation testing to replace some of the testing workload described here. Such evidence must be no older than the last prior major version (i.e., v3.x or other v4.x reviews may supplement work done in the current review) of the PCI HSM Security Requirements prior to the current evaluation's submission. If leveraging separate evidence, it is necessary to justify that this evidence is fully in scope of DTR A5 security requirements.

TA5.6 The tester shall confirm what mitigations exist with respect to the exploitation of timing attacks against asymmetric cryptography, and justify (with reference to best practice in protecting cryptographic operations from timing attacks) why such mitigations are sufficient. Where it is not possible to confirm mitigations or no mitigations exist, the tester shall validate through testing that the HSM is not vulnerable to timing-based attacks.

B – Logical Security Derived Test Requirements

DTR B1 Self-Tests

To ensure that the device is operating as designed, the device runs self-tests (a) when pre-operational and at least once per day or (b) using continuous error checking, to check firmware (authenticity check), security mechanisms for signs of tampering, and whether the device is in a compromised state. When specific critical operations are performed, the device performs conditional tests. The techniques and actions of the device upon failure of a self-test are consistent with those defined in FIPS 140-2/140-3.

Guidance

Firmware is considered to be any code within the device that provides security protections needed to comply with these requirements. The evaluating lab may require copies of source code and assistance from the vendor to make a systematic review of relevant security functions. See Appendix F, "Domain-Based Asset Flow Analysis."

HSMs used in the payments industry frequently operate in non-PCI mode, and self-tests are required to ensure the integrity of those operations.

The HSM must detect hardware errors in order to prevent the return of incorrect results or the disclosure of sensitive information. Examples of methods to detect hardware errors include:

- *Execution of periodic self-tests with a frequency of at least once each 24 hours in addition to pre-operational execution and after a hardware reset.*
- *Automatic continuous error-checking built into the hardware, such as duplicate paths with comparison of results, parity, error-correcting codes, and other similar methods.*

Failure in accordance with FIPS 140-2/140-3 can be either a degraded mode of operation where the module allows functionality not related to the self-test failure or the module halting and blocking output from all external interfaces.

In all cases, initial testing of firmware when it is loaded from mass storage to memory is still required to detect tampering when on the storage device.

Error detection must cover all components of the HSM that could influence security should they fail. This includes memory, microprocessors, cryptographic algorithms, firmware (authenticity and integrity), security mechanisms that detect tampering, and any items indicating the HSM is in a compromised state.

Firmware integrity tests may include techniques such as SHA-2 or equivalent. The hash must be cryptographically protected using a key (e.g., HMAC-SHA-2). Authenticity testing must use cryptographic methods ((H)MACs, digital signatures, or encryption). LRC, CRC, and other non-cryptographic methods and weak cryptographic methods (e.g., SHA-1, MD5) are not allowed as the primary mechanism for either authentication or integrity checking.

Internal generation of a cryptographic signature is valid right after firmware update, assuming it complies with Requirement B3; it is also valid for devices that do not allow for software updates outside of the factory.

(continued on next page)

Failing in a secure manner involves at least disabling any functionality of the device related to processing sensitive data. More specifically, no sensitive data breach can occur as a consequence of device failure.

A device does not require authenticity checking when self-testing its firmware if (all apply):

- *The authenticity checking of firmware—either internally and according to B3 or externally using appropriate procedures within a secured environment under the vendor's control—is performed whenever the firmware is established in that secure area; and*
- *The effort to deliberately modify or replace the firmware or parts of it in order to get access to sensitive information (access to the memory device) must be addressed as an attack scenario under Requirements A1, A3, and A4 and meet the respective attack potentials; and*
- *A periodic integrity check according to Requirement B1 is performed for the firmware, ensuring that random changes will be detected; and if cryptographic authenticity is not performed, the integrity check must be cryptographically based. Although an algorithm using a secret key, such as a keyed hash, can be used, it is not necessary for meeting the integrity criteria.*

These conditions apply regardless of any non-reconfigurable property of the device memory.

When firmware is externally authenticated, the level of security shall be of the same level as for key-injection facilities.

All self-tests shall be performed and determination of pass or fail shall be made by the module without external controls, externally provided input test vectors, expected output results, or operator intervention or whether the module will operate in an approved or non-approved mode.

Periodic self-tests shall be performed automatically and repeatedly within a defined time period, without external input or control. The device must assure that the tests occur within a 24-hour cycle. Resetting, rebooting, and power cycling are acceptable means for the on-demand initiation of pre-operational or conditional tests.

The time period and any conditions that may result in the interruption of the module's operations to perform the pre-operational or conditional self-tests shall be specified in the security policy—e.g., if the module is performing mission-critical services that can't be interrupted and the time period is passed for the initiation of the pre-conditional self-tests, the self-tests may be deferred to the next such time period, However, at all times the self-test must be performed on boot prior to any operational use, and within 2x the self-test cycle during continued use—i.e., never more than 48 hours since the last such test.

Pre-operational self-tests shall be performed and passed successfully by a cryptographic module between the time the module is powered on or instantiated (after being powered off, reset, rebooted, cold-start, power interruption, etc.) and before the module transitions to the operational state. A cryptographic module shall perform the following, as applicable:

- *Pre-operational software/firmware integrity test*
- *Pre-operational bypass test*
- *Pre-operational critical functions test*

(continued on next page)

Conditional self-tests shall be performed when an applicable security function or operation is invoked—i.e., security functions for which self-tests are required). Conditional self-tests shall be performed by a cryptographic module when the conditions specified for the following tests occur: Cryptographic Algorithm Self-Test, Pair-Wise Consistency Test, Software/Firmware Load Test, Manual Entry Test, Conditional Bypass Test and Conditional Critical Functions Test.

A cryptographic module may perform other pre-operational or conditional self-tests in addition to the tests specified below.

TB1.1 The tester shall describe the boot chain of the device. The tester shall Include how initial machine code is loaded and executed by the processing elements, and how any subsequent firmware modules are sequenced, loaded, and executed, up to and including software modules used for processing sensitive data. The tester shall reference the relevant aspects of the asset flow.

TB1.2 The tester shall verify that the device performs self-tests upon startup and on a periodic basis at least once per day to check firmware and security mechanisms for signs of tampering, and whether the device is in a compromised state. The tester shall activate the self-test(s) and look for the result of the self-test(s) as shown by the device.

TB1.3 The tester shall verify that the device self-tests are able to detect failures and in doing so, fail in a secure manner. The vendor shall provide evidence of testing that confirms the relevant component fails securely in the event of self-test failure.

TB1.4 For any self-test functions that are implemented by the built-in functions of the security processing elements, the tester shall detail what sources of information and testing have been used to validate that these processes are in place.

TB1.5 The tester shall review the source code of the device to confirm what algorithms and keys are used to perform the self-test functions that are implemented by the firmware of the device. The tester shall confirm that any register settings required to activate hardware-based self-test functions are correctly assigned.

TB1.6 The tester shall review the source code of the device to confirm how it is ensured that the self-test process(es) are repeated every 24 hours, or through an automatic continuous error-checking built into the hardware.

TB1.7 The tester shall note what methods are implemented to authenticate the cryptographic keys of the device, to ensure that they have not been modified after loading.

TB1.8 The tester shall detail the processes performed by the device if one or more of the self-tests fail. The tester shall confirm this through source-code review.

TB1.9 From the previous descriptions, the tester shall complete the following table indicating the process used to authenticate the firmware images during each stage of the booting process. The tester shall include all self-tests for all processing elements within the device (as detailed in DTR A3). The tester shall adapt the table as necessary—for example, by adding columns or additional notes—to present any additional information.

Boot stage	Algorithms and Key Sizes Used for Authentication	Area/Code/Registers Authenticated	Method and Frequency of Re-authentication	Action Performed if Failed

TB1.10 The tester shall confirm that the self-tests include validation of register settings relied upon for the security of the device—for example, registers used to activate security features of the device. The tester shall validate that these include checking of all features detailed and relied upon for compliance to the physical security requirements.

TB1.11 The tester shall verify that the device performs the following pre-operational self-tests:

a) **Cryptographic algorithm test**

A test using a known answer shall be conducted for all cryptographic functions (e.g., encryption, decryption, authentication, and random number generation) of each approved cryptographic algorithm implemented by a cryptographic module. A known-answer test involves operating the cryptographic algorithm on data for which the correct output is already known and comparing the calculated output with the previously generated output (the known answer). If the calculated output does not equal the known answer, the known-answer test shall fail.

Cryptographic algorithms whose outputs vary for a given set of inputs (e.g., the Digital Signature Algorithm) shall be tested using a known-answer test or shall be tested using a pair-wise consistency test (specified below). Message digest algorithms shall have an independent known-answer test, or the known-answer test shall be included with the associated cryptographic algorithm test (e.g., the Digital Signature Standard).

If a cryptographic module includes two independent implementations of the same cryptographic algorithm:

- The known-answer test may be omitted;
- The outputs of two implementations shall be continuously compared; and
- If the outputs of two implementations are not equal, the cryptographic algorithm test shall fail.

b) **Software/firmware integrity test**

An authenticity validation mechanism using an approved digital signature, MAC, or protected HASH value shall be applied to all software and firmware components within the module's defined cryptographic boundary. If the calculated result is not successfully verified, the test fails and the module shall enter the error state. The digital signature technique may consist of a single encompassing signature or multiple disjointed signatures; failure of any disjointed signature shall cause the module to enter the error state. The private signing key shall reside outside the module.

The pre-operational software/firmware integrity test is not required for any software or firmware for any executable code stored in non-reconfigurable memory.

If a hardware module does not contain either software or firmware—for example where it is an IP block within an SOC or a dedicated ASIC cryptographic chip—the module shall at a minimum implement one cryptographic algorithm self-test as specified in TB1.15 as a pre-operational self-test.

A cryptographic algorithm that is used to perform the approved integrity technique for the pre-operational software/firmware test shall first pass the cryptographic algorithm self-test specified in conditional cryptographic algorithm self-tests.

c) **Bypass test**

If a cryptographic module implements a bypass capability, the module shall ensure the correct operation of the logic governing activation of the bypass capability by exercising that logic. The module shall also verify the data path by:

- Setting the bypass switch to provide cryptographic processing and verify that data transferred through the bypass mechanism is cryptographically processed, and
- Setting the bypass switch to not provide cryptographic processing and verify that data transferred through the bypass mechanism is not cryptographically processed.

d) **Critical functions tests (e.g., RAM test)**

Other security functions critical to the secure operation of a cryptographic module shall be tested as part of the pre-operational tests.

Documentation shall specify all security functions critical to the secure operation of a cryptographic module and shall identify the applicable pre-operational tests performed by the module.

TB1.12 The tester shall induce the pre-operational self-tests and, if applicable, view any status provided by the device to verify that the self-tests executed successfully.

TB1.13 The tester shall verify that the device automatically performs the following continuous or periodic self-tests:

a) Cryptographic algorithm tests
b) Software/firmware integrity test
c) Critical functions tests (e.g., RAM test)
d) Random number generator test
e) Other self-tests that are performed pre-operation and on demand

TB1.14 If applicable, the tester shall induce the periodic self-tests and, if applicable, view any status.

TB1.15 The tester shall verify that the device performs the following conditional self-tests where applicable, when the conditions specified exists:

a) **Cryptographic algorithm test**

A test shall be conducted for all cryptographic functions (e.g., security functions, SSP establishment methods, and authentication) of each approved cryptographic algorithm implemented in the cryptographic module as referenced in Appendix D. The conditional test shall be performed prior to the first operational use of the cryptographic algorithm.

A cryptographic algorithm self-test may be a known-answer test, a comparison test, or a fault-detection test.

A known-answer test consists of a set of known input vectors (e.g., data, keying material, or constants in lieu of random bits) which are operated on by the cryptographic algorithm to generate a result. The result is compared to the known expected output result. If the calculated output does not equal the known answer, the cryptographic algorithm known-answer self-test shall fail.

An algorithm self-test shall use at minimum the smallest approved key length, modulus size, DSA prime, or curves (as appropriate) supported by the module.

If an algorithm specifies multiple modes (e.g., ECB, CBC, etc.), a minimum of one mode shall be selected for the self-test that is supported by the module or as specified by the validation authority.

Examples of known-answer tests:

- One-way functions: Input test vector(s) generate output which shall be identical to expected output—e.g., hashing, keyed hashes, message authentication, RBG (fixed entropy vector), SSP agreement.
- Reversible functions: Both the forward and reverse function shall be self-tested (e.g., symmetric key encryption and decryption, SSP transport encryption and decryption, digital signature generation and verification)

A comparison test compares the output of two or more independent cryptographic algorithm implementations; if the outputs are not equal, the cryptographic algorithm comparison self-test shall fail.

A fault-detection test involves the implementation of fault detection mechanisms integrated within the cryptographic algorithm implementation; if a fault is detected, the cryptographic algorithm fault-detection self-test shall fail.

b) Pairwise consistency tests

If a cryptographic module generates public or private keys, the following pair-wise consistency tests for every pair of generated public and private keys shall be performed:

- If the keys are used to perform key transport, the public key shall encrypt a clear-text value. The resulting ciphertext value shall be compared to the original clear-text value. If the two values are equal, the test shall fail. If the two values differ, the private key shall be used to decrypt the ciphertext and the resulting value shall be compared to the original clear-text value. If the two values are not equal, the test shall fail.
- If the keys are used to perform the calculation and verification of digital signatures, the consistency of the keys shall be tested by the complete calculation and verification of a digital signature. If the digital signature cannot be verified, the test shall fail.
- If the keys are used to perform SSP agreement, the arithmetic validity of the keys shall be tested by verifying the correct mathematical relationship between the public key and private key values.

c) Manual entry test

If SSPs or key components are manually entered into a cryptographic module, or if error on the part of the human operator could result in the incorrect entry of the intended key, the following manual key entry tests shall be performed:

- The SSPs or key components shall have an error-detection code (EDC) applied or shall be entered using duplicate entries.
- If an EDC is used, the EDC shall be at least 16 bits in length.
- If the EDC cannot be verified, or the duplicate entries do not match, the test shall fail.

d) Software/firmware load test

If software or firmware components can be externally loaded into a cryptographic module, the following software/firmware load tests shall be performed:

- The cryptographic module shall implement an approved authentication technique (e.g., an approved message authentication code, digital signature algorithm, or HMAC) to verify the validity of the software or firmware that is loaded;

- The reference authentication key shall be loaded independently in the module prior to the software or firmware loading; and
- The applied approved authentication technique shall be successfully verified or the software/firmware load test shall fail. Loaded software or firmware shall not be used if the software/firmware load test fails.

e) Continuous random number generator test

If a cryptographic module employs RNGs in an approved mode of operation, the module shall perform the following continuous random number generator test on each RNG that tests for failure to a constant value.

- If each call to an RNG produces blocks of n bits (where n > 15), the first n-bit block generated after power-up, initialization, or reset shall not be used, but shall be saved for comparison with the next n-bit block to be generated. Each subsequent generation of an n-bit block shall be compared with the previously generated block. The test shall fail if any two compared n-bit blocks are equal.
- If each call to an RNG produces fewer than 16 bits, the first n bits generated after power-up, initialization, or reset (for some n > 15) shall not be used but shall be saved for comparison with the next n generated bits. Each subsequent generation of n bits shall be compared with the previously generated n bits. The test fails if any two compared n-bit sequences are equal.

f) Bypass test

If a cryptographic module implements a bypass capability where the services may be provided without cryptographic processing (e.g., transferring clear text through the module), the following bypass tests shall be performed to ensure that a single point of failure of module components will not result in the unintentional output of clear text:

- A cryptographic module shall test for the correct operation of the services providing cryptographic processing when a switch takes place between an exclusive bypass service and an exclusive cryptographic service.
- If a cryptographic module can automatically alternate between a bypass service and a cryptographic service, providing some services with cryptographic processing and some services without cryptographic processing, the module shall test for the correct operation of the services providing cryptographic processing when the mechanism governing the switching procedure is modified (e.g., an IP address source/destination table).
- If a cryptographic module maintains internal information that governs the bypass capability, the module shall verify the integrity of the governing information through an approved integrity technique immediately preceding modification of the governing information. Immediately following the modification, a new integrity value using the approved integrity technique shall be generated.
- Documentation shall specify the mechanism or logic governing the switching procedure.

g) Critical functions tests

- Other security functions critical to the secure operation of a cryptographic module performed under specific conditions shall be tested as conditional tests.
- Documentation shall specify all security functions critical to the secure operation of a cryptographic module and shall identify the applicable conditional tests performed by the module.

TB1.16 The tester shall induce the conditional self-tests and, if applicable, view any status provided by the device to verify that self-tests executed successfully.

TB1.17 For both pre-operational and continuous/periodic tests the tester shall:

a) Verify the vendor documentation provides, for each error state
   - The condition name, description of the state,
   - The events that can produce the state, and
   - The actions necessary to clear the state and resume normal operation.

b) Cause each error state to occur and attempt to clear the error state. The tester shall verify that actions necessary to clear the error state are consistent with the vendor documentation. If the tester cannot cause each error state to occur, the tester shall verify the code listing and/or design documentation and whether the actions necessary to clear each error state are consistent with the descriptions in the vendor documentation.

c) Verify that all self-tests are performed regardless of whether the cryptographic module operates in PCI mode or non-PCI mode.

d) Verify by inspection and from the vendor documentation that determination of pass or fail of each self-test is made by the module, without external controls, externally provided input text vectors, expected output results, or operator intervention.

e) Verify that upon entering an error state the cryptographic module does not perform any cryptographic operations or output control and data via the control and data output interface while in an error state.

f) Verify that the cryptographic module does not utilize any functionality that relies upon a function or algorithm that failed a self-test until the relevant self-test has been repeated and successfully passed.

TB1.18 The tester shall verify from vendor documentation that:

- An unauthorized operator cannot access the error log.
- The error-logging functionality provides information on the most recent error event, at a minimum.

TB1.19 The tester shall cause the cryptographic module to enter an error state and verify that:

- The module generates the error log for the most recent error event, at a minimum.
- The error log cannot be accessed without assuming any authenticated role supported by the cryptographic module.
- The error log is protected against unauthorized modification and substitution.

TB1.20 The tester shall present sufficient evidence and/or references for the above, for compliance to this DTR.

DTR B2 Logical Anomalies

The device's functionality shall not be influenced by logical anomalies such as (but not limited to) unexpected command sequences, unknown commands, commands in a wrong device mode, and supplying wrong parameters or data which could result in the device outputting sensitive information.

Guidance

Functionality shall be considered as any functionality, via any internal or external interface, that could impact the security of the device.

Vendors should provide software design rules and specifications to support answers.

All interfaces and associated communication methods of the device must be assessed without exception to ensure that no interface can be abused or used as an attack vector. Specifically, this includes any physical, logical, or application interface that is executed within the device with sufficient privilege to allow for direct interface to sensitive assets within the device (should that protocol be compromised in some way). The interfaces must be documented and assessed regardless of whether they are used for or have access to card data. This analysis must be done in accordance with the Appendix F, "Domain-Based Asset Flow Analysis." Sufficient evidence must be provided to demonstrate the validity of laboratory assessments.

The vendor shall provide evidentiary matter providing details on internal testing including, but not limited to, the following:

- Source-code reviews targeting specific relevant security-critical functionalities.
- Vulnerability analysis that includes gathering and considering evidence necessary to perform practical testing.
- Penetration testing to validate the robustness of the device to protect against feasible attacks by addressing known attack methods. For example (but not restricted to), fuzzing, using appropriate tools and techniques.
- Audits of relevant existing test evidence, which may be utilized where appropriate by giving justifications for validity of evidence and test methodologies overall.

The laboratory shall determine the veracity of the material provided to determine the degree of reliance that may be placed upon the evidence and, where necessary, the laboratory shall extend the testing.

TB2.1 The tester shall describe the vendor's measures that ensure that the device's functionality is not influenced by logical anomalies such as (but not limited to) unexpected command sequences, unknown commands, commands in a wrong device mode, and supplying wrong parameters or data.

TB2.2 The tester shall note the programming languages in which the device's firmware source code is written for each of the security processing elements (as detailed in DTR A3).

TB2.3 The tester shall detail the type, version, capabilities, and configuration of the operating system(s) used on each of the device's security-processing elements (as detailed in DTR A3).

TB2.4 The tester shall enumerate all logical and physical interfaces provided by the device. For example, the tester shall include physical interfaces such as USB input, serial input, IC interface, and TCP/IP interface, as well as API interfaces provided to applications that may execute on the device.

TB2.5 If the device includes command-execution interfaces or parsers: The tester shall detail how each of the above interfaces is configured to accept commands—for example, whether a command executive is used, or other methods are used to parse input commands. The tester shall define which common functionalities exist between interfaces to determine which test approaches may be applied in common to more than one interface.

TB2.6 The tester shall detail in an appendix to the evaluation report a complete list of all APIs as defined by the vendor that are provided on each of the logical interfaces of the device.

TB2.7 The tester shall perform a source-code review of each relevant interface and confirm that it is handled securely, that only documented commands are implemented, and that secure defaults are provided for each interface. The tester shall detail the methods used to verify the length and content of each command before processing. The tester shall derive and describe vulnerability-analysis models from source-code review and other available evidence to determine attack paths and appropriate penetration testing.

The source-code review should be targeted on relevant security-critical functionalities such as (but not restricted to): buffer overflows; unhandled exceptions, read-access violations, and denial-of-service conditions, including factors that are specific to the device's OS, communications protocols, and source code software language(s).

TB2.8 For systems that are designed to execute non-firmware applications, the vendor shall provide a test application containing a buffer-overflow vulnerability to be run into the device, together with the application's source code. The tester shall run the application and determine whether the device fails securely.

TB2.9 The tester shall identify and review publicly available sources of vulnerability disclosure—including but not necessarily limited to those referenced in the vendor-certification process document—and shall check that any vulnerabilities found cannot be exploited on the device.

The tester shall compare his/her analysis with the analysis provided by the vendor and confirm that the vendor has remediated any problems with these vulnerabilities, and that firmware-audit report documents exist and have been updated to validate this assertion.

TB2.10 The tester shall execute a vulnerability assessment, public vulnerability check, and/or testing, to identify vulnerabilities associated with the interfaces and associated communication methods of the device.

TB2.11 The tester shall perform fuzzing of the device's security-relevant interfaces in order to uncover logical anomalies.

The tester shall describe fuzzing methodologies and tools used, as well as any assistance provided by the vendor in the case of white-box fuzzing and results obtained. The description and results should be sufficient to provide a demonstrably high level of confidence in the security of the device's interfaces.

The methodology used should provide an optimum balance between the use of evaluation resources for penetration testing and reliance upon the vendor-provided information.

The tester shall provide any evidence and descriptions necessary to support conclusions.

The evaluation may rely upon appropriate testing results provided by the vendor, which shall be less than two years older than the date of the current evaluation submission, in order to replace/supplement the laboratory testing.

The vendor's testing findings and testing methodology should be documented.

The tester shall describe the methodology and the findings of the vendor's testing, which must be sufficient to demonstrate that they are at least as robust as the criteria described here.

TB2.12 The tester shall identify all command interpreters within the firmware, including all command interpreters within the HSM software which implement commands that can be invoked from the host system. If command interpreters are called, the tester shall describe and justify why a command or environment cannot be manipulated to perform unauthorized functions.

TB2.13 The tester may perform any additional tests necessary to validate the device’s property. The vendor shall provide any necessary test support to access and use the interfaces under test.

DTR B3 Firmware Updates

The device must support firmware updates. The device must cryptographically authenticate the firmware, and if the authenticity is not confirmed, the firmware update is rejected and deleted.

The update mechanism ensures security—i.e., integrity, mutual authentication, and protection against replay—by using an appropriate and declared security protocol when using a network connection.

Guidance

Firmware is considered to be any code within the device that provides security protections needed to comply with PCI requirements. Other code that exists within the device that does not provide security, and cannot impact security, is not considered firmware under PCI requirements.

The authentication must not be performed by a component of lesser protection strength than the one for which the firmware/software is intended, OR the authentication must be performed by the target component of the firmware/software.

The firmware and application version numbers must be available via display, API command, and/or printing during startup or on request. This shall be illustrated by photographic or screen-capture-based evidence provided in the evaluation report.

The displayed firmware version number(s) must represent all firmware in the device that is currently executable.

- *If firmware blocks have independent version numbers, the version number display should include the version number of each firmware block.*
- *If a single version number is used, a documented process must be used to ensure the single version number is updated whenever changes are made to any of the firmware blocks in the device.*

For HSMs that implement virtualized environments or allow for multiple copies of firmware to be resident at one time, users must be able to obtain the version number(s) of the firmware they are currently executing. If additional firmware blocks or images are resident and may be executed at any time without user interaction or permission, the firmware version(s) of those must be provided as well (if not already encapsulated within a single firmware version).

Where multiple users may exist in an HSM Solution and each user may have a separate firmware version, it must not be possible for one user to query or otherwise obtain the firmware version currently used by another user.

TB3.1 The tester shall verify that the device supports firmware updates and the vendor has a methodology for deploying updates to the device as they become available.

TB3.2 The tester shall verify and demonstrate that the device displays or otherwise makes available the firmware revision number or numbers when firmware has different numbering for different modules.

TB3.3 The tester shall determine by which component the authentication is performed.

TB3.4 The tester shall determine the level of protection for the external component involved in firmware/software updates and that the authentication of firmware updates is performed by a component of equal or greater strength.

TB3.5 The tester shall examine the vendor-supplied documentation to verify that the controls provide for unique accountability and utilize key sizes appropriate for the algorithm(s) in question. Examples of appropriate algorithms and minimum key sizes are stated in Appendix D.

Examples of acceptable hashing algorithms include SHA-256, SHA-384 and SHA-512. MD5 and SHA-1 are not allowed for use.

TB3.6 For each of the processing elements listed in DTR A3, the tester shall complete the following table. Where different parts of the code can be updated independently (for example, boot code, main firmware, etc.), or if one part of the firmware cannot be updated, the tester shall ensure that this is detailed in the table as well. The tester shall reference the relevant aspects of the asset flow.

The tester shall adapt the table (for example, by adding columns or additional notes) as necessary, to present any additional information.

Firmware Element	Elements Used to Perform Authentication	Algorithms and Key Sizes Used for Firmware Authentication	Format of Authentication Block	Process Performed if Authentication Failed

If the method used for initial loading of the firmware differs from the method used for code update, provide additional details (including another of the tables above, if deemed necessary) of how initial code is loaded into the device.

In the “Format of Authentication Block Column,” include details on the format and padding of the authentication block (for example X.509 certificate using OAEP padding).

TB3.7 The tester shall review the source code of the device to confirm that the firmware-authentication methods are implemented correctly as noted above, and that the authentication is performed within the secure firmware of the device. This evaluation activity should be focused on relevant security-critical sections of the source code, to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB3.8 If the device allows for loading of multiple types of code (for example, firmware for security processor and application code), the tester shall detail how the various types of update images are differentiated from one another to prevent one type of image being incorrectly loaded into the wrong processing element/location. The tester shall ensure all authentication methods and image types are contained in the table above.

TB3.9 If (H)MAC method(s) are used for firmware authentication, the tester shall confirm through source-code review that the method used to compare the firmware-authentication block does not leak timing information—for example, the “C” memcmp() function is not used. This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB3.10 If a CBC MAC is used for firmware authentication, the tester shall detail what methods are used to mitigate vulnerabilities when authenticating variable-length data.

TB3.11 For each of the methods of authentication, the tester shall obtain a correctly authenticated firmware image and:

a) Confirm that it loads correctly into the device. The tester shall detail the loading process.

b) Change a single bit in the authentication block of the image and confirm that this modified image is rejected when loaded into the device.

c) Change a single bit in the firmware block of the image and confirm that this modified image is rejected when loaded into the device.

TB3.12 The tester shall confirm how any public or private/secret keys are loaded into the device during manufacturing. The tester shall specifically note whether any default values are installed (for example, default public certificates hard-coded into the firmware of the device) and how it is ensured that these must be changed in deployed devices.

TB3.13 Where a network connection is supported, the tester shall describe how a secure channel can be used for protection of software updates. Describe how the firmware-provided secure channel provides mutual authentication, integrity, and replay protection.

DTR B3.1 Application Authenticity

If the device supports applications, the firmware must support the authentication of applications loaded onto the terminal consistent with B3.

Guidance

Applications are considered to be any code or scripts that can be loaded onto the device that is not firmware. Other code that exists within the device that does not provide security, and cannot impact security, is not considered. See Appendix F, "Domain-Based Asset Flow Analysis."

The authentication must not be performed by a component of lesser protection strength than the one for which the access is intended, OR the authentication must be performed by the target component.

If the device allows software application and/or configuration updates, the device cryptographically authenticates updates consistent with B3.

TB3.1.1 The tester shall determine by which component the authentication is performed.

TB3.1.2 The tester shall determine the rank of protection strength for the component involved in application loads—and if applicable, software and/or configuration updates—and that the authentication is performed by an appropriate component

TB3.1.3 The tester shall examine the vendor-supplied documentation to verify that the controls provide for unique accountability and utilize key sizes appropriate for the algorithm(s) in question. Examples of appropriate algorithms and minimum key sizes, along with examples of acceptable hashing algorithms, are stated in Appendix D.

TB3.1.4 For each of the processing elements listed in DTR A3, the tester shall complete the following table. Where different applications or application components—e.g., executable code, scripts, etc—can be updated independently, or if one part of the application cannot be updated, the tester shall ensure that this is detailed in the table as well.

The tester shall adapt the table (for example, by adding columns or additional notes) as necessary, to present any additional information.

Processing/ Application Element	Elements Used to Perform Authentication	Algorithms and Key Sizes Used for Application Authentication	Format of Authentication Block	Process Performed if Authentication Failed

If the method used for initial loading of the application differs from the method used for code update, provide additional details (including another of the tables above, if deemed necessary) of how initial code is loaded into the device.

In the "Format of Authentication Block" column, include details on the format and padding of the authentication block (for example, X.509 certificate using OAEP padding).

TB3.1.5 The tester shall review the source code of the device to confirm that the application authentication methods are implemented correctly as noted above, and that the authentication is performed within the secure firmware of the device. This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB3.1.6 If (H)MAC method(s) are used for application authentication, the tester shall confirm through source-code review that the method used to compare the application-authentication block does not leak timing information—for example, the "C" memcmp() function is not used. This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB3.1.7 If a CBC MAC is used for application authentication, the tester shall detail what methods are used to mitigate vulnerabilities in this method when used to authenticate variable-length data.

TB3.1.9 For each of the methods of authentication the tester shall obtain a correctly authenticated application image and:

a) Confirm that it loads correctly into the device. The tester shall detail the process involved in performing the loading process.

b) Change a single bit in the authentication block of the image and confirm that this modified image is rejected when loaded into the device.

c) Change a single bit in the application block of the image and confirm that this modified image is rejected when loaded into the device.

TB3.1.10 The tester shall confirm how any public or private/secret keys are loaded into the device during manufacturing. The tester shall specifically note whether any default values are installed (for example, default public certificates hard-coded into the firmware of the device) and how it is ensured that these must be changed in deployed devices.

DTR B4 Logical Interfaces

The device provides secure interfaces that are kept logically separate by distinguishing between data and control for inputs as well as between data and status for outputs.

Guidance

The device shall provide for the following four logical interfaces:

- *Data input interface – All data (except control data entered via the control input interface) that is input to and processed by a cryptographic module (including clear-text data, ciphertext data, cryptographic keys and CSPs, authentication data, and status information from another module) shall enter via the "data input" interface.*
- *Data output interface – All data (except status data output via the status output interface) that is output from a cryptographic module (including clear-text data, ciphertext data, cryptographic keys and CSPs, authentication data, and control information for another module) shall exit via the "data output" interface. All data output via the data output interface shall be inhibited when an error state exists and during self-tests.*
- *Control input interface – All input commands, signals, and control data (including function calls and manual controls such as switches, buttons, and keyboards) used to control the operation of a cryptographic module shall enter via the "control input" interface.*
- *Status output interface – All output signals, indicators, and status data (including return codes and physical indicators such as Light Emitting Diodes and displays) used to indicate the status of a cryptographic module shall exit via the "status output" interface.*

TB4.1 The tester shall examine any relevant documentation distributed by the vendor such as schematics, data sheets, asset flow diagrams, and assembly drawings submitted by the vendor to ensure that the vendor has exhaustively defined all protocols and interfaces supported by the device. The vendor must also identify the source of its code used to implement protocols, as well as document all protocols and the physical interfaces to which they apply. When public libraries are used to implement protocols, their versions must be specified.

TB4.2 The tester shall review vendor documentation to verify that the device distinguishes between data and control for inputs and also between data and status for outputs. To provide evidence that the device distinguishes between commands and data inputs, the tester shall verify that the device responds with error messages or ignores inputs when erroneous commands and data are input.

DTR B5 Clearing of Internal Buffers

The device must automatically clear or reinitialize its internal buffers that hold sensitive information once the information is no longer needed, including when:

- ***The transaction is completed, or***
- ***The device has timed out or***
- ***The device recovers from an error state.***

Guidance

Vendor shall provide documentation of test results for inspections of internal buffers.

This applies to sensitive data, such as passwords/authentication codes, clear-text cryptographic keys outside of the crypto-processor, and clear-text PIN values.

TB5.1 The tester will verify that and summarize how the vendor has identified all data that is automatically cleared once the information is no longer needed, including when the transaction is completed, the device has timed out, or the device recovers from an error state, and that all sensitive data is included. Passwords, clear-text cryptographic keys, key components outside of the crypto-processor, and PIN values are considered sensitive data.

TB5.2 The tester shall review the source code of the device and confirm that—and summarize how—sensitive information is cleared from all storage locations after use, including local variables (before exiting the function) and registers. The tester shall detail the methods used to perform the erasure.

TB5.3 The tester shall detail the method used by the vendor to ensure that this buffer-clearing code/function cannot be removed by compiler optimizations or other means of code optimization, if employed by the vendor.

TB5.4 The tester shall detail the testing performed to verify that buffer-clearing code/function is robust. This requires assistance from the vendor and may involve, for example:

- Review of a small sample of compiled object code to validate that the code to clear the buffer remains in the compiled code;
- Extraction of memory from a special sample device after execution of the buffer-clearing code; or
- Confirmation that any compiler flags to ensure optimization are functioning as expected.

TB5.5 The tester shall confirm that—and indicate how—via review, the vendor has the requirement for buffer clearing documented in their software-development practices documentation, including specific notes on how this should be done to prevent removal by compiler optimization, and that the correct implementation of this guide is reviewed as part of the firmware-verification process validated as part of DTR B3.

TB5.6 The tester shall perform any additional tests necessary to verify that all data is automatically cleared including when the transaction is completed, the device has timed out, or the device recovers from an error state for instance, by performing a partial simulated transaction to verify the behavior at time-out, or in general by entering the states that have been defined by the vendor under TB5.1.

DTR B6 Protection of Sensitive Functions or Services

Access to sensitive services requires authentication. Sensitive services provide access to the underlying sensitive functions. Sensitive functions are those functions that process sensitive data such as cryptographic keys, PINs, and passwords/authentication codes. Entering or exiting sensitive services shall not reveal or otherwise affect sensitive data.

Guidance

Authentication shall require dual-control techniques when manually entering unprotected sensitive information into the device through a secure user interface, or cryptographic techniques when electronically transferring protected data into the device. The use of other techniques to access sensitive services results in the device being unable to use previously existing keying material. The device requires the cooperation of at least two separately authenticated operators for administration services not normally available, such as clear text or split knowledge of manual CSP loading or CSP output, enabling or disabling device security functions, or the modification of authentication data. The manual entry or output of CSPs in enciphered form requires at least one authenticated operator. The device limits the number of function calls (services) and the time limit on these services. If the limits are exceeded, re-authentication is required.

"CSPs" are critical security parameters, such as cryptographic keys and passwords/authentication codes.

The device's key-management functions are designed so that no key can be disclosed without collusion amongst individuals. Specifically:

- *The device's highest-level keys are manually loaded as at least two components under dual control;*
- *Any function used to input or output key components does not operate until at least two different passwords/authentication codes have been entered.*

A sensitive service (state) allows the execution of functions that are not available during normal use—e.g., load a master key, delete stored transactions, alter device configuration, etc. The authority to execute functions that are not available during normal use is controlled through use of a sensitive state or other special authorization that controls the ability to execute these functions. Services not normally available may include CSP loading, CSP output, enabling or disabling device security functions, or the modification of authentication data.

It must not be feasible to determine a key or other secret information by the use of diagnostic or special test modes.

The functionality implemented within the device is such that there is no feasible way in which clear-text secret information (e.g., cryptographic keys), or secret information enciphered under other than the legitimate key, can be obtained from the device, except in an authorized manner

Passwords/authentication codes used for authentication are at least seven characters or an equivalent strength.

Key components entered manually constitute sensitive data during entry and the device shall not differentiate via sound or display the entry of different values.

(continued on next page)

The following operations are regarded as sensitive services requiring authentication as delineated in this requirement:

- *Disablement/enablement of device functionality;*
- *Changing of passwords/authentication codes or data that enable the device to enter a sensitive state;*
- *Configuring or modifying the security policy examined in C1 and B10.*

TB6.1 The tester shall examine vendor documentation and verify that the vendor has identified all sensitive services, data, and secure modes. Sensitive functions are those functions that process sensitive data such as cryptographic keys, PINs, and passwords/authentication codes.

TB6.2 The tester shall verify from vendor documentation and from functional testing that sensitive services require authentication.

TB6.3 The tester shall verify from vendor documentation and from functional testing that entering and exiting sensitive services does not reveal or otherwise affect sensitive information.

TB6.4 The tester shall verify from vendor documentation that sensitive services are entered, used, and exited securely and that mode transitions (for example, from operational to maintenance) do not reveal or otherwise affect sensitive information.

TB6.5 The tester shall determine that for each authentication attempt, the probability is less than one in one million that random attempts for authenticating both operators will succeed. For multiple consecutive attempts in a one-minute period, the probability is less than one in one hundred thousand that a random attempt will succeed***.***

TB6.6 The tester shall attempt to perform any other services listed in this section and verify that each service requires the cooperation of at least two separately authenticated operators for administration services not normally available, such as clear text or split knowledge of manual CSP loading or CSP output, enabling or disabling device security functions, or the modification of authentication data.

TB6.7 The tester shall verify that placing the device into a diagnostic or special test mode does not allow the determination of a secret key or other secret information.

TB6.8 The tester shall examine any additional documentation (e.g., API reference, design documentation, key-management specification) that describes authentication procedures for the manual entry or output of enciphered CSPs to determine whether it supports the assertions made by the vendor.

TB6.9 The tester shall examine any additional documentation (e.g., API reference, design documentation, key-management specification) that describes sensitive service limits to determine whether it supports the assertions made by the vendor.

TB6.10 The tester shall attempt to exceed the vendor specified limit on the number of function calls (services) and time limit. Once the limit is exceeded, the tester shall verify that the device has returned to its normal state and requires re-authentication.

TB6.11 The tester shall verify that the vendor documentation is consistent and provides sound rationale for the service limits set by the vendor.

- The vendor has provided a rationale for the value chosen as a limit on the number of actions and the time limits imposed.
- The vendor has provided a rationale as to how the limits minimize the risks from unauthorized use of sensitive services.

TB6.12 If access to sensitive services requires input by a secure user interface, the tester shall verify that the protections such as the following are afforded to the data entered while accessing sensitive services:

- Data inputs cannot be discerned from any displayed characters.
- Data inputs cannot be discerned by monitoring audible or electro-magnetic emissions.
- Sensitive data is cleared from internal buffers upon exiting the sensitive mode.

The testing shall include:

- Entering data while accessing sensitive services.
- Document review.

TB6.13 If mode transitions require input by a separate interface device, such as a key loader or other vendor-supplied interface device, document the mechanism(s) and methodology used.

TB6.14 The tester shall detail in the table below all methods that can be used to load cryptographic keys into the device. This shall include situations in which there is more than one loading method for any particular key, and in which different cryptographic keys may have different methods of loading. The tester shall include any APIs provided by the firmware that allow for the loading of cryptographic keys (reference the API list provided as part of DTR B2).

The items provided in the table below are for the purposes of example only:

Cryptographic key	Method of loading per TB6.7	Authentication
MFK	Components through external device	Two seven-character passwords through key loader
	Encrypted under Public Key	Provided by encryption
	Internally generated	Operator authentication provided via chip cards
Authentication Key	Embedded in firmware	Two eight-character passwords required to operate firmware loading

TB6.15 The tester shall justify why each of the methods that can be used to load cryptographic keys enforces both dual control and split knowledge.

TB6.16 The tester shall detail any further sensitive services provided by the device that have not been addressed as defined above. This must include the disabling or alteration of any systems or functions relied upon by the device to meet the PTS requirements

.

TB6.17 Where public keys are loaded into the device in a non-secure environment—for example, during firmware loading—the tester shall justify why dual control is maintained and alteration or manipulation of the public key value(s) is not possible.

***Note:** This applies to the loading of initial high-level (e.g., root) keys. Where those keys are embedded in the firmware, they will leverage the mechanisms used for the firmware as required under L5. Other public keys that are loaded after the device has received its FW can be authenticated using cryptographic mechanisms (like a certificate or a MAC), and the loading can happen in a non-secure area without compromising security.*

TB6.18 The tester shall detail any other sensitive services offered by the device and detail the authentication provided for these services. The tester shall justify how this ensures dual control is provided for all sensitive services.

TB6.19 The tester shall validate that—and describe how—all passwords/authentication codes implemented to provide dual control are at least seven characters or an equivalent strength.

TB6.20 The tester shall attempt to load cryptographic keys or components into the device without changing the default values of the passwords/authentication codes. The tester shall detail the results. The requirements of this DTR are not met if this can be done.

TB6.21 The tester shall attempt to set the passwords/authentication codes of the device to a value that is less than seven characters or an equivalent strength. The tester shall detail the results. The requirements of this DTR are not met if this can be done.

DTR B7 Key Entry

Private and secret key entry is performed using accepted techniques according to the table below.

Key Form	**Technique**		
	Manual	**Direct**	**Network**
Clear-text keys	No	Yes	No
Clear-text key components	Yes	Yes	No
Enciphered keys/components	Yes	Yes	Yes

Guidance

Key entry may be implemented in the following ways:

a) Manual (e.g., key-component entry via a keypad);

b) Electronic direct loading (e.g., direct key injection via a cable from the originating device or a key-transfer device);

c) Network distribution and loading (e.g., remote key transport via a network), including remote key loading using asymmetric techniques.

Keys loaded using any other mechanism can only be loaded enciphered.

See ISO 11568 for further information.

HSM commands that include operational keys encrypted under an HSM master or storage key are not in scope for this requirement.

The tester shall verify all that apply.

External SCDs (secure key loaders or other interface devices) used for loading clear-text secret or private keys or their components must either be approved against the PCI PTS HSM Modular Security Requirements *or be compliant to the requirements for devices with key-transfer and loading functionality as described in ISO 13491-2. The vendor must either provide or describe a specific, existing device that can be used for this functionality.*

Clear-text Key Entry:

TB7.1 The tester shall verify the following:

- Any clear-text keys entered uses direct techniques.
- Any clear-text keys are directly entered without traveling through any enclosing or intervening systems.
- Vendor documentation indicates that the device used to load keys is a secure cryptographic device including physical protections.

Clear-text key entry requires dual authentication prior to loading of the new key.

TB7.2 The tester shall verify that the device does not allow entry of clear-text keys using manual or network techniques.

TB7.3 The tester shall verify that an audit log is kept by the device and the device used to perform the electronic key loading. Additionally, it is procedurally specified in the vendor security policy where necessary to track actions to the specific individuals involved (i.e., automated authentication mechanisms are role based and not identity-based). In either case, the electronic log must implement cryptographic mechanisms as specified in B15. The audit log must be able to be used to track the individuals who perform clear-text key entry.

Clear-text Key Components Entry:

TB7.4 For clear-text key components, the tester shall verify the following:

- Key components are directly entered without traveling through enclosing or intervening systems or a directly connected SCD is used as key loader.
- A minimum of two components is required to reconstruct the original key.
- If knowledge of *n* components is required to reconstruct the key, knowledge of any *n*-1 components provides no information about the original key other than the length.
- Clear-text key-component entry requires the use of dual-control techniques, such that the key is entered as separate key components by the assigned key custodians. Each component requires the use of a separate passwords/authentication codes to authenticate the individual entering that specific component OR the zeroization of pre-existing secret keys within the associated key hierarchy prior to the loading of the new key—i.e., the invoking of the key-loading function/command causes the zeroization prior to the actual loading of the new key.

TB7.5 The tester shall verify that the device does not allow entry of clear-text key components using network techniques**.**

TB7.6 The tester shall verify that an audit log is kept by the device and the device used to perform electronic key loading. Additionally, it is procedurally specified in the vendor security policy, where necessary to track actions to the specific individuals involved—i.e., automated authentication mechanisms are role-based and not identity-based. In either case, the electronic log must implement cryptographic mechanisms as specified in B14. The audit log must be able to be used to track the individuals who perform clear-text key component entry.

Enciphered and Remote Key Entry:

TB7.7 The tester shall verify that any enciphered key or key component input or output from the device is enciphered using an acceptable cryptographic algorithm and key size. KEKs must be at least as strong as the keys they protect.

Examples of appropriate algorithms and minimum key sizes are stated in Appendix D.

The tester shall verify that if asymmetric keys are used for entry or output, they are used in accordance with Annex A of the PCI PIN Security Requirements and Requirement B10.

TB7.8 The tester shall verify that an audit log is kept by the device used to perform electronic key loading. Additionally, it is procedurally specified in the vendor security policy where necessary to track actions to the specific individuals involved—i.e., automated authentication mechanisms are role-based and not identity-based. In either case, the electronic log must implement cryptographic mechanisms as specified in B14. The audit log must be able to track the individuals who perform enciphered key entry.

DTR B8 Random Numbers

If random numbers are generated by the device in connection with security over sensitive data, the random number generator has been assessed to ensure that it is generating sufficiently unpredictable numbers.

Guidance

Unpredictability of random numbers is as important as distribution. The implementation shall ensure that seeding or initializing the random number generator at startup cannot be abused to intentionally reproduce a given random value or sequence.

The device shall be capable of meeting the statistical tests of NIST SP PUB 800-22. See Appendix C.

Random numbers generated for use in the creation of cryptographic keys require a strong source of entropy in order to prevent predictability. Other uses, such as the generation of random nonces for the prevention of replay attacks when using asymmetric techniques to distribute symmetric keys, may only require the guarantee of uniqueness.

Source code will be required to validate this requirement.

The device output of those numbers cannot be influenced—e.g., by varying environmental conditions of the device such as manipulating the power supply/electro-magnetic injection.

PRNG designs (Deterministic Random Bit Generator, or DRBG) from NIST SP800-90A or ANSI X9.82 shall be used. Specifically, HASH_DRBG, HMAC_DRBG or CTR_DRBG. DEA and 2-key TDEA, as well as DUAL_EC_DRBG are not acceptable for use in a DRBG.

The evaluating lab may require assistance from the vendor to make a systematic review of relevant security functions.

TB8.1 The tester shall detail the method used by the device to generate random numbers, including any seed values used, hardware systems, and software-based deterministic pseudo random number generators (DPRNG).

The tester shall outline the process used by the vendor to ensure that any secret values relied upon for random number generation (such as seed values, or keys used in DPRNGs) are sufficiently random, and unique per device.

The tester shall justify why the method used by the device to generate random numbers is robust.

TB8.2 The tester shall confirm that the process outlined above includes a method to provide entropy from a hardware-based source, as well as a software-based "whitening" process such as a DPRNG to remove any bias in the hardware-based system. The tester shall provide a justification that this method ensures that sufficient entropy is provided for all uses of the random number generator.

The tester shall ensure that initializing and/or seeding of the RNG cannot be abused to intentionally reproduce a given random value or sequence.

TB8.3 The tester shall list all security services implemented within the device that require or rely upon the use of random data. This may include generation of padding data—for example, for use in certificates or key bundles, generation of cryptographic keys, etc.

TB8.4 The tester shall review the source code of each of these services and confirm that they correctly utilize the random number generator reviewed in this requirement. This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB8.5 The tester shall obtain at least 128MB of random data from the device under test. This data may be supplied directly by the vendor. The tester shall detail the method used to generate this data and justify why this sufficiently replicates the way in which the RNG will be used by the device. The tester shall pass the 128MB of data through the NIST STS test program and detail the results, indicating pass and fail results and how these demonstrate compliance to this DTR.

If the data does not pass all tests and the failure is marginal, the tester should acquire additional data from the vendor and repeat the testing using both the initial data and the additional vendor-supplied data. See Appendix C, "Configuration and Use of the STS Tool."

DTR B9 Cryptographic Algorithms

The device uses accepted cryptographic algorithms, modes, and key sizes.

Guidance

All PCI related algorithms used must be implemented in accordance with standards referenced in the PCI PTS HSM Modular Security Requirements *and the* PCI PTS Testing and Approval Program Guide*.*

Any method used to produce encrypted text that relies on "non-standard" modes of operations (e.g., format-preserving Feistel-based Encryption Mode (FFX)) shall be approved by at least one independent security evaluation organization (e.g., a standards body) and subjected to independent expert review; such methods shall also be implemented following all guidelines of said evaluation and peer review including any recommendations for associated key management.

TB9.1 The tester shall verify that accepted algorithms are used for the following (as applicable):

- Key loading
- Log data protection
- Data authentication
- User authentication
- Key encryption
- Software/firmware updates
- PIN management
- All security protocols

Examples of appropriate algorithms and minimum key sizes are stated in Appendix D.

Key-encryption keys must be of equal or greater strength than the keys they are used to protect.

Examples of acceptable hashing algorithms include SHA-256, SHA-384 and SHA-512. MD5 and SHA-1 are not allowed for use for protocols impacting the device's security, such as firmware updates.

The vendor must provide the rationale for the use of any other algorithms used for the aforementioned purposes.

TB9.2 In the event of a non-standard mode of operation being used, the tester shall examine documented credentials of the expert reviewer and assess his/her ability to perform the review. The tester shall also examine documentation supporting the assertion of independence of review and confirm that the reviewer is indeed independent. The tester will use his or her judgment in determining the appropriate due diligence.

TB9.3 The tester shall verify that the mechanism used has been implemented following all guidelines of any security evaluation and independent expert review including any recommendations for associated key management and configuration of the mode of operation.

TB9.4 For each of the accepted cryptographic algorithms implemented within the device, the tester shall verify the correct implementation of the algorithms through any of the following means:

- Review of algorithm verification certifications (e.g., NIST CMVP)
- Review of algorithm test results submitted by the vendor from an independent accredited testing organization
- Evaluation of sample data or internal testing performed by the vendor
- Testing performed by the evaluator

DTR B10 Key Management

The key-management techniques implemented in the device conform to ISO 11568 and/or ANSI X9.24. Key-management techniques must support key blocks as defined in DTR B10.

Guidance

The device must provide support for AES Master File Keys (MFKs), whether loaded from components or internally generated. It may also provide support for MFKs that are equivalent to or greater in strength than triple-length TDES keys. The device may optionally provide support for backwards compatibility for double-length TDES Master File Keys.

A device may include other key-management schemes not specified below, such as country-specific techniques when operating in PCI mode, as long as their use cannot in any way compromise the security of the PCI-compliant methods. Performing simulated transactions is not required for these key-management techniques.

Similar key-management techniques should be used for other services, such as card personalization and data protection.

Symmetric key components shall be combined either XOR'ing of full-length key components or implementation of a recognized secret-sharing scheme—for example, Shamir. Private key components shall be combined using a recognized secret-sharing scheme. Devices must implement unique secret and private keys for any function directly or indirectly related to PIN or account-data protection. The basic rule is that any private or secret key resident in the device that is directly or indirectly used for PIN protection whose compromise would lead to the compromise of the same key in another device must be unique per device. For example, this means not only the PIN-encryption key(s), but keys that are used to protect other keys, firmware-update and authentication keys, and display-prompt control keys. As stated in the requirement, this does not apply to public keys resident in the device.

When a check value is generated for a key or key component, it shall be generated by a cryptographic process such that all portions of the key or key component are involved in generating the check value.

A device may include more than one compliant key exchange and storage scheme.

Devices must support the ASC X9 TR 31 or ANSI X9.143 key-derivation methodology for TDES keys, and for AES keys must support the TR 31 and/or X9.143 methodology and/or the ISO 20038 methodology. The device may optionally support, in addition, the TR 31 or ANSI X9.143 Key Variant Binding Method for TDES keys. TR 31 and X9.143 are recognized as interoperable methods for both TDEA and AES. ISO 20038 is recognized as an interoperable method for AES.

In either case, equivalent methods can be used where subject to an independent expert review and said review is publicly available as described below. Other methods may additionally be supported where required by legal or regulatory requirements in specific markets but can no longer be supported in lieu of the aforementioned.

(continued on next page)

Any equivalent method must include the cryptographic binding of the key-usage information to the key value using accepted methods. Any binding or unbinding of key-usage information from the key must take place within the secure cryptographic boundary of the device. For all methods used, the encrypted key and its attributes in the key block shall have integrity protection such that they cannot be modified without detection. Modification includes, but is not limited to:

- *Changing or replacing any bit(s) in the attributes or encrypted key*
- *Interchanging any bits of the protected key block with bits from another part of the block*

Documentation must be provided demonstrating how the methodology meets these criteria.

Equivalent methods must be subject to an independent expert review, and said review must be publicly available:

- *The review by the independent expert must include proof that in the equivalent method the encrypted key and its attributes in the key block have integrity protection such that it is computationally infeasible for the key to be used if the key or its attributes have been modified. Modification includes, but is not limited to:*
  - *Changing or replacing any bit(s) in the attributes or encrypted key*
  - *Interchanging any bits of the protected key block with bits from another part of the block*
- *The independent expert must be qualified via a combination of education, training, and experience in cryptology to provide objective technical evaluations that are independent of any ties to vendors and special interests. "Independent expert" is further defined below.*
- *The PTS laboratory will validate that any device vendors implementing this methodology have done so following all guidelines of said evaluation and peer review, including any recommendations for associated key management.*

An Independent Expert possesses the following qualifications:

- *Holds one or more professional credentials applicable to the field, e.g., doctoral-level qualifications in a relevant discipline or government certification in cryptography by an authoritative body (e.g., NSA, CES, or GCHQ); and*
- *Has ten or more years of experience in the relevant subject; and*
- *Has published at least two articles in peer-reviewed publications on the relevant subject, or*
- *Is recognized by his/her peers in the field—e.g., awarded the Fellow or Distinguished Fellow or similar professional recognition by an appropriate body, e.g., ACM, BCS, IEEE, IET, IACR; and*
- *Subscribes to an ethical code of conduct and would be subject to an ethics compliance process if warranted.*

Independence requires that the entity is not subject to control, restriction, modification, or limitation from a given outside source. Specifically, independence requires that a person, firm, or corporation who holds itself out for employment as a cryptologist or similar expert to more than one client company is not a regular employee of that company, does not work exclusively for one company, and where paid, is paid in each case assigned for time consumed and expenses incurred.

(continued on next page)

This does not imply that the device must enforce TR 31 or an equivalent scheme, but it must be capable of implementing such a scheme as a configuration option.

These methods must be used for key loading whenever a symmetric key (e.g., AES or TDES) is loaded encrypted by another symmetric key. This applies whether symmetric keys are loaded manually—i.e., through a secure interface, using a key-injection device, or from a remote host. It does not apply when clear-text symmetric keys or their components are loaded using standard dual-control techniques.

The evaluating lab may require copies of source code and assistance from the vendor to make a systematic review of relevant security functions.

The software library/chipset used to generate asymmetric keys shall be identified in the Asset Flow Diagram to help safeguard against weak key-generation algorithms.

Any key calculated as a variant of another key, or derived from another key, shall be protected in a manner equivalent to or greater than the security of the original key. The use of variants is not allowed for AES keys.

AES keys can only be:

- *Loaded using asymmetric keys of equivalent or greater strength.*
- *Encrypted by another AES key of equal or greater strength.*
- *Manually loaded using dual-control techniques.*
- *Internally generated using a random number generator compliant with B8.*

***Note:** RSA keys with a modulus of 2048 bits may be used to load 128-bits AES keys in conformance with ANSI TR-34. Other public key techniques such as Diffie-Hellman or Elliptic Curve must be used to convey AES keys greater in strength than 128 bits.*

Devices are allowed to have keys that are not unique per device if these keys are used for load-balancing purposes. This does not preclude cryptographic-keying relationships with POI devices or other organizations—for example, symmetric keys that exist both in the HSM and within another system that communicates to the HSM using that key, such as a POI device.

TB10.1 The tester shall determine from vendor documentation the key-management technique used for firmware and application updates. Symmetric key techniques must include the use of Unique Key(s) per Device.

TB10.2 The tester shall examine any additional documentation (e.g., user guide, API reference, design documentation, key-management specification) that describes the implemented key exchange and storage techniques to determine whether it supports the assertions made by the vendor.

TB10.3 The tester shall verify that the loading of private and secret keys uses one or more of the following methods:

a) When manually entering clear-text secret keys through a secure interface device, they must be entered as two or more components with each component requiring the use of a separate password/authentication codes. The passwords/authentication codes must be entered through the secure interface device or else conveyed encrypted into the device. These passwords/authentication codes must either be unique per device (and per custodian), except by chance, or if vendor default, they are pre-expired and force a change upon initial use. Passwords/authentication codes that are unique per device can be made optionally changeable by the acquirer, but this is not required. Passwords/authentication codes are at least seven characters or an equivalent strength.

b) For injecting clear-text secret or private keys from a key loader (which has to be some type of secure cryptographic device), either the key loader or the device or both must require two or more passwords/authentication codes before injecting the clear-text key into the device*. (**Note:** This may be the entire key—if components, each component requires a separate password/authentication code).* These passwords/authentication codes are entered directly through the secure interface of the applicable device or are conveyed encrypted into the device and must be at least seven characters or an equivalent strength These passwords/authentication codes must either be unique per device (and per custodian), except by chance, or if vendor default, they are pre-expired and force a change upon initial use. Clear-text keys or their components are never permitted over a network connection.

c) For encrypted values injected into the device, either from a key loader or from a network host, or via a secure interface device, the ability of the device to successfully decrypt the value and use it is sufficient. In this case, the key-encipherment key is trusted because it was previously loaded under dual control—e.g., in examples a) and b) above or using another secure and trusted method, e.g., loaded encrypted as part of a key hierarchy.

d) Remote key-loading techniques using public-key methods require compliance with PCI defined criteria for key sizes and mutual authentication between host and device. For devices generating their own key values, the generation process must meet the criteria defined in the random number appendix of the DTRs and validation that appropriate key sizes are used. The protocol must meet the criteria stipulated in Annex A of the *PCI PIN Security Requirements.*

TB10.4 The minimum key sizes and parameters for the algorithm(s) in question that must be used for key transport, exchange or establishment are stated in Appendix D:

If a public-key technique for the remote distribution of symmetric secret keys related to PIN or account-data encryption is used, it must:

a) Use public and private key lengths that are deemed acceptable for the algorithm in question (e.g., 2048-bits minimum for RSA).

b) Use key-generation techniques that meet the current ANSI, ISO, and NIST standards for the algorithm in question.

c) Provide for mutual device authentication as specified in Annex A of the *PCI PIN Security Requirements* for both the host and the device, including assurance to the host that the device actually has obtained or computed (or actually can compute) the key and that no other entity other than the device specifically identified can possibly compute the session key.

TB10.5 If used for online PIN translation, the tester shall verify that the device supports TDES and AES for online PIN translation. The tester shall perform simulated transactions for each accepted combination of key-management technique and cryptographic algorithm (TDES, AES) supported by the device (e.g., DUKPT; MK/SK and Fixed Key).

TB10.6 The tester shall determine from vendor documentation all storage and usage locations for each key (e.g., ROM, external RAM, EPROM, processor chip, etc.) that is stored within the device and list the details in a key summary table. The tester shall reference the relevant aspects of the asset flow.

TB10.7 The tester shall determine from vendor documentation how each key is destroyed—e.g., active or passive erasure—for all device states (power-on, power-off, sleep mode) and list the details in a key summary table.

TB10.8 For systems that support Master/Session key management, the tester shall define what (if any) standard this key management complies with—for example, where the device implements country-specific key management. The tester shall note the name, version, and authoring party of this standard.

TB10.9 The tester shall review the API guide and operations manual of the device and confirm that this does not detail any other key-management schemes or methods that the device may use. If the device supports extensible key management for use with non-PIN transactions, the tester shall justify what prevents this from being used with PCI-based PINs, using extracts from the vendor documentation to support this claim. The tester shall provide references to all extracts used.

TB10.10 The tester shall detail any other types of key management or cryptographic keys used by the device. This should include any keys used for firmware or application authentication, self-testing, boot strapping, remote key injection, local key injection, dual control, etc.

TB10.11 The tester shall provide a key table for the device, accurately including all of the keys and key-management methods outlined above. This does not include keys temporarily present in the device as part of transaction processing. An example of key types in such a table is:

Key Name	Purpose/ Usage	Algorithm	Size (Bits)	Generated by:	Form Factor Loaded to Device In	Number of Available Key Slots (Registers)	Unique per device/ acquirer/ vendor-specific/ other (describe)	How the key is identified by the device so that it is used only as intended
Master Key	Encryption of working keys (PEK, MAC) for local storage	TDES	128	Acquirer	2 or 3 clear-text components	One	Device	Designated Key Register
Manufacturer Authentication Root Public Key	Authentication of firmware updates	RSA	2048	Manufacturer	Self-signed Public Key Certificate	One	Vendor-specific	Designated Key Register
Manufacturer Authentication Private Root Private Key	Signing firmware updates	RSA	2048	Manufacturer	Managed at manufacturer's secure facility under dual control	One	Vendor-specific	N/A
KDH Private Key	Used for signing messages to POIs	RSA	2048	Acquirer	Generated by device	Varies	Acquirer	Cryptographic Authentication under Vendor PKI

Note: *Other keys may be loaded to the device as components (e.g., a terminal master key) or internally generated for external storage under the device's Master File Key or a variant thereof. If internally stored, the tester shall note this in the summary table.*

Note: *The "Size" column must note both the maximum and minimum sizes that can be used for that key.*

TB10.12 Using the key table as a reference, the tester shall note which keys are actively erased by the device during a tamper event, and which keys are not erased but instead rely upon the erasure of a KEK to prevent their subsequent misuse.

TB10.13 Using the key table and API guide as a reference, the tester shall note which keys can be loaded by applications in clear text.

TB10.14 Using the table of sensitive-information storage from Requirement A3 and the key table above, the tester shall confirm:

a) No key is encrypted under a key of lesser strength; and

b) Clear-text cryptographic keys are not stored encrypted under bulk data-encryption keys (such as keys used to encrypt external memory).

TB10.15 The tester shall detail any ways in which the device generates keys from other keys, specifically:

a) Variants are not used across different levels of the hierarchy—for example, variants of KEKs are not used to produce working keys. The only allowable exception to this requirement is the use of TR 31 variant method(for backwards compatibility).

b) The tester shall detail any key-generation methods used and justify why these are valid key-generation functions as required by ISO 11568 and ANSI X9.24.

TB10.16 The tester shall note whether the device generates any keys using an internal random number generator. The tester shall confirm through source-code review that these keys are generated using the same process validated under Requirement B8. This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB10.17 The tester shall detail the method used by the device to confirm that no one key can take the same value as any other key within the device. Through source-code review confirm the following:

a) The method used does not provide a potential timing attack on the device—for example, by using a standard C "memcmp()" function to compare all keys.

b) If key check values (KCVs) are used for this purpose, the KCVs stored are limited to values as defined in TB10.18 or they are never output from the device.

c) The method used does not rely on the check digits (e.g., mod 10 calculation) of a (T)DES key as part of the key comparison.

This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

TB10.18 Referencing the device API, user guides, and other documentation, the tester shall confirm that it is not possible to output a KCV value except as noted below.

Note: *Check values may be computed by two methods. TDEA may use either method. AES must only use the CMAC method. In the first method, check values are computed by encrypting an all-binary zeros block using the key or component as the encryption key, using the leftmost n-bits of the result, where n is at most 24 bits (6 hexadecimal digits/3 bytes). In the second method the KCV is calculated by MACing an all-binary zeros block using the CMAC algorithm as specified in ISO 9797-1 (see also* NIST SP 800-38B*). The check value will be the leftmost n-bits of the result, where n is at most 40 bits (10 hexadecimal digits). The block cipher used in the CMAC function is the same as the block cipher of the key itself. A TDEA key or a component of a TDEA key will be MAC'd using the TDEA block cipher, while a 128-bit AES key or component will be MAC'd using the AES-128 block cipher.*

TB10.19 The tester shall note what methods are implemented to authenticate the cryptographic keys of the device to ensure that they have not been modified after loading.

TB10.20 The tester shall detail the key-block method(s) supported by the device, providing details on exact contents of any header, payload, padding, etc. as well as noting which parts are encrypted and which are included in any authentication calculation. If ASC X9 TR 31 or ANSI X9.143 key derivation or ISO 20038 method is not used, the tester shall justify why the method used provides the same level of security. Specifically, the tester shall note how the key-blocking method supports each of the following properties:

- Key confidentiality
- Key integrity
- Key purpose
- Algorithm used for the key
- Key length

If TR 31 and/or X9.143 key-block variant method is used, the tester shall confirm that at least one other method of key blocks is used and that this other method does not have the KEK and MAC keys related as variants.

TB10.21 The tester shall confirm that any key-block key can only be used for that purpose and cannot be used as a "generic" master or working key, as part of a non-bundled key-management scheme.

TB10.22 For any methods that rely on the use of TDES full-length key components for enforcing split knowledge, the tester shall attempt to load all but one of the components as an all-zero value. If this does not succeed, the tester shall attempt to load a zero-value component where the parity bits have been modified so that the actual value of the component entered is not composed entirely of zeros. For key shares, the tester shall use the same value for all but one share to perform the aforementioned.

The requirements of this DTR are not met if it is possible to load a key where the value of that key can be known through knowledge of only one component. The findings of these tests shall be explained.

DTR B11 Cryptographic Key Loss

The device ensures that if cryptographic keys within the secure device boundary are rendered invalid for any reason—e.g., tamper or long-term absence of applied power—the device will fail in a secure manner.

Guidance

Any unintentional or malicious modification (i.e., corruption) of cryptographic security parameters is detected and the device fails in a secure manner.

TB11.1 The tester shall force the device to erase its cryptographic keys (e.g., via a command, removal of power, tamper event). The tester shall verify that the device fails in a secure manner (e.g., reverts to an un-initialized state and not a normal operational state).

DTR B12 Encryption or Decryption of Arbitrary Data Within the Device

The device ensures that each cryptographic key is only used for a single cryptographic function. It is not possible to encrypt or decrypt any arbitrary data using any PIN-encrypting key, account-data encryption, data-encrypting key, or key-encrypting key contained in or protected by the device. The device does not permit any of the key-usage information to be changed in any way that allows the key to be used in ways that were not possible before the change.

Guidance

PIN-encryption keys used for acquiring shall only be used to encrypt PIN data. Key-encrypting keys shall only be used to encrypt keys. PIN keys shall never be used to encrypt keys. Key-encrypting keys shall never be used to encrypt PIN data.

Account-data encryption keys shall only be used to encrypt account data. Account-data encryption keys shall never be used to encrypt keys.

A secret key used to encrypt a PIN must never be used for any other cryptographic purpose. A key used to protect the PIN-encrypting key must never be used for any other cryptographic purpose. However, variants of the same key may be used for different purposes, as may keys derived from the original key. Any variant of the PEK or a key used to protect the PEK must be protected in the same manner (i.e., under the principles of dual control and split knowledge).

***Note:** The use of variants is not allowed for AES keys.*

Key-usage information may be changed where the usage is made more restrictive (e.g., made non-exportable).

Key variants are only used in devices that possess the original key (e.g., the device and the host it communicates with). Key variants are not used at different levels of the key hierarchy (e.g., a variant of a key-encipherment key used for key exchange cannot be used as a working key or as a Master File Key for local storage).

Private keys shall only be used to create digital signatures and to perform decryption operations, and in some cases to compute negotiated keys. Private keys shall never be used to encrypt other keys.

The evaluating lab may require copies of source code and assistance from the vendor to make a systematic review of relevant security functions.

TB12.1 The tester shall examine any additional documentation (e.g., API reference, design documentation, key-management specification) that describes the use of cryptographic keys and relating to encryption and decryption of arbitrary data to determine whether it supports the assertions made by the vendor. For example: A public key shall not be used by the device for both signature authentication and encryption. Examples of a key's "cryptographic purpose" are: Data encryption, key encryption, and MAC.

TB12.2 The tester shall verify that the device uses one of the following techniques to ensure that a key is only used for its intended purpose:

- Physical segregation
- Storing keys enciphered under a KEK dedicated to encipherment of a specific type of key
- Modifying or appending information to a key as a function of its intended purpose, prior to encipherment of the key for storage, e.g., key tags

TB12.3 Through source-code review, the tester shall confirm what methods the device uses to confirm the purpose and integrity of each key. The tester shall validate that this does not reduce the effective strength of any key (for example, by including a CRC or purpose tag within the effective value of the cryptographic key).

The tester shall verify the following:

a) PIN-encryption keys are only used to encrypt PIN data.

b) Account-data encryption keys are only used to encrypt account data.

c) Key-encrypting keys are only used to encrypt keys.

d) Data keys shall never be used to encrypt other keys or PIN or account data.

This evaluation activity should be focused on relevant security-critical sections of the source code to provide an optimum balance between use of evaluation resources against evaluation goals overall.

DTR B13 Clear-Text Key Security

There is no mechanism in the device that would allow the outputting of private or secret clear-text keys, the encryption of a key or PIN under a key that might itself be disclosed, or the transfer of a clear-text key from a component of high security into a component of lesser security. All cryptographic functions implemented shall not output clear-text critical security parameters (CSPs) to components that could negatively impact security.

Guidance

Clear-text secret and private keys and clear-text PINs must not exist in unprotected environments. ***Note:*** *This does not apply to cryptographic keys that are never used to encrypt or decrypt data; or are not used for authentication.*

Clear-text secret and private keys shall only be output into another secure cryptographic device under dual control if the device is designed for key injection.

Components (shares) of secret and private keys may be output in the clear to key mailers or key-transfer devices.

Clear-text PINs may be output in issuer environments. Clear-text PINs are not allowed to be output under the security policy (see C1 and B14) for devices used for PIN processing by an acquirer or its agent.

The evaluating lab shall require copies of source code and assistance from the vendor to make a systematic review of relevant security functions.

TB13.1 The tester shall examine any documentation—i.e., the Asset Flow Analysis, API programmer's guide, specifications, block diagrams, etc.—containing information relating to:

- Encryption of a key or PIN under a key that might itself be disclosed, and
- The transfer of a key from a high-security component to a lower-security component.

TB13.2 Referencing the table of sensitive-information storage provided in Requirement A3, the tester shall note whether cryptographic keys, customer PINs, or other sensitive data are exported outside the security processor to other components (including memory components) within the device. The tester shall justify why any such transfer of keys ensures that they remain secure.

DTR B14 PIN Management

If the device is designed to be used for PIN management, the device shall meet the PIN management requirements of ISO 9564. The PIN-encryption technique implemented in the device is a technique included in ISO 9564.

Guidance

PIN block formats intended for use by Issuers in connection with PIN unblock or PIN change on chip cards may also be supported but are still subject to the PIN translation restrictions enumerated below.

Note: *If supporting online PIN processing:*

1. *The device must support at least one of the following key-management techniques using AES as described in ANSI X9.24 and ISO/IEC 18033-3:*
   - *DUKPT*
   - *Master/Session*
   - *It may also support Fixed Key*
2. *TDES as described in ANSI X9.24 and ISO/IEC 18033-3 may also be supported using the following key-management techniques:*
   - *DUKPT*
   - *Master/Session*
   - *It may also support Fixed Key*

If supporting online PIN entry, the device must support ISO PIN-block format 4 and may support any of the following PIN-block formats:

- *ISO Format 0*
- *ISO Format 1*
- *ISO Format 3*

Note: *The HSM API must ensure that the restrictions cannot be circumvented by chaining multiple calls, such as first translating a PIN block from ISO format 1 to ISO format 0 and the invoking a PVV/offset calculating or verifying function.*

TB14.1 The tester shall verify that the security policy configuration examined in C1 only allows the following translations when the policy is implemented.

Standard PIN-block formats (i.e., ISO formats 0, 1, 2, 3, and 4) shall not be translated into non-standard PIN-block formats.

PINs enciphered using ISO format 0, ISO format 3, or ISO format 4 must not be translated into any PIN-block format other than ISO format 0, 3, or 4 except when translated to ISO format 2 as specified in the table below. PINs enciphered using ISO format 1 may be translated into ISO format 0, 3, or 4 but must not be translated back into ISO format 1. ISO format 1 may be translated into ISO format 2 as specified in the table below.

Translations between PIN-block formats that each include the PAN shall not support a change in the PAN. The PIN-translation capability between ISO formats 0, 3, or 4 (including translations from ISO format 0 to ISO format 0, from ISO format 3 to ISO format 3, or from ISO format 4 to ISO format 4) must not allow a change of PAN. The following illustrates translations from formats 0, 1, 3 and 4:

Note: *This translation restriction is not applicable to surrogate PANs used in tokenization implementations.*

This table illustrates restrictions on translations between PIN block formats that are applicable when the HSM does not enforce unique-key-per-transaction encryption for the resulting PIN block:

Translation			
To → / **From ↓**	**ISO Format 0, 3, 4**	**ISO Format 1**	**ISO Format 2**
ISO Format 0, 3, 4	– Permitted anywhere without change of PAN – Change of PAN only permitted in sensitive state for card issuance – Change of PAN token to real PAN only permitted with cryptographic binding of PAN token to real PAN	Not permitted	Permitted for submission to an IC card
ISO Format 1	Permitted	Permitted	Permitted for submission to an IC card
ISO Format 2	Not permitted	Not permitted	Permitted for submission to an IC card

TB14.2 From the above list of PIN-block formats, the tester shall confirm that the HSM supports ISO PIN-block format 4 if the device supports online PIN. The device may optionally support format 0, 1, or 3 for online PINs.

TB14.3 The tester shall verify the following (as applicable):

- Journaled transaction messages do not contain a PIN.
- All key-encryption keys must have an appropriate length and are used with an accepted algorithm.

TB14.4 The tester shall verify the following restrictions apply for compact PIN block formats (0, 1, 2, and 3) regardless of key-management technology used.

- Use of format 2 PIN blocks shall be constrained to offline PIN verification and PIN change operations in ICC environments only.
- Only ISO formats 0 and 3 shall be supported in calculating values used for PIN verification that are derived from the PIN and PAN—e.g., PIN offsets and PIN verification values (PVV).
- When calculating values derived from the PIN and PAN, if the portion of the account number enciphered in the input encrypted PIN block does not agree with the input PAN, the calculated value shall not be returned except in the following case: Where the introduction of a new PAN is required to support account number changes for card issuance, support for change of PAN during calculation of the derived value shall be provided only while the host security modules are in a sensitive state and under dual control. (See ISO 13491-1.)

TB14.5 The tester shall verify the following restrictions apply for format 4 PIN Blocks

- ISO format 4 may be supported in calculating values used for PIN verification that are derived from the PIN and PAN, e.g., PIN offsets and PIN verification values (PVV).
- When calculating values (such as PVVs or offsets) derived from the PIN and PAN, if PAN 1 used for the derivation of the calculated value does not agree with PAN 2 used in the clear-text PAN field, the calculated value shall not be returned except in the following case: Where the introduction of a new PAN is required to support account number changes for card

issuance, support for change of PAN during calculation of the derived value shall be provided only while the host security modules are in a sensitive state and under dual control. (See ISO 1349-1.)

- No integrity checks shall be performed on the PIN digits themselves. If integrity checks are performed on the deciphered PIN field, they shall only be performed on the first byte of that field (control field and PIN length field) and the fill digits.

DTR B15 Secure Logging

The device includes cryptographic mechanisms to support secure logging of transactions, data, and events, to enable auditing.

Guidance

Logs shall not contain clear-text sensitive data unless those logs are protected in accordance with Requirement A3. Logs exported from the device shall never contain clear-text sensitive data.

TB15.1 The tester shall verify that the device provides mechanisms that allow log data stored within the device to be protected—e.g., using cryptographic mechanisms that allow log data stored within the device to be protected from unauthorized modification and deletion or from modification if output for storage. The vendor security policy must address administrative actions regarding periodic management and archiving of log data.

TB15.2 The tester shall verify that the device supports secure logging, including time stamps, of security sensitive commands such as:

a) Key management

b) Modification of authentication data

c) Software/firmware updates

d) Enabling and disabling device security functions

TB15.3 The tester shall verify that the device requires dual control to delete secure logs stored internally.

TB15.4 The tester shall verify that the device never exports logs that contain clear-text sensitive data.

DTR B16 Application Separation

If the device supports multiple applications, it must enforce the separation between applications. It must not be possible that one application interferes with or tampers with another application or the OS/firmware of the device, including, but not limited to, modifying data objects belonging to another application or the OS/firmware. Similarly, enforcement of separation must be provided if the device supports virtualization such that it can act as multiple logically separate devices.

Guidance

Applications, in this context, are functional entities that execute within the boundary of the device and may or may not provide services external to the device. Applications are typically processes or tasks that execute under the control of an operating system (OS) or software executive routine.

Applications are considered to be separated by access rights. OS/firmware is considered all code, which is responsible to enforce, manage, or change such access rights. Therefore, OS/firmware code is necessarily part of the firmware as defined with B18.

Applications may share data by design.

The addition of applications that replace or disable the PCI-evaluated firmware functionality invalidates the device approval for each such implementation unless those applications are validated for compliance to PTS HSM Modular Security Requirements and listed as such in the approval listings.

Specifically, those applications must be validated to ensure that devices may allow customers or integrators to install additional applications where the vendor can show that by permitting this:

a) It cannot adversely affect the security features of the product that are relevant to the PCI device certification.

b) It cannot modify any of the cryptographic functionality of the device or introduce new primitive cryptographic functionality. However, new composite functionality that builds on existing primitives is permitted.

c) The application is authenticated to the device by digital signature.

d) The application does not have access to sensitive keys.

e) The application can only work on the keys it alone manages and cannot affect or see any other keys.

f) Applications must not share process spaces with each other or with the firmware.

See Appendix F, “Domain-Based Asset Flow Analysis.”

A mechanism must exist to display or retrieve the application version upon request.

The vendor must provide clear security guidance in the user available security policy referenced in C1 for the development and implementation of the aforementioned additional applications. This guidance at a minimum must define procedural controls to ensure that the applications are properly reviewed, tested, and authorized.

Vendors should provide configuration lists and description of the separation mechanism to support answers.

Focal point is for sensitive data and sensitive functions.

TB16.1 The tester shall note whether the device allows for non-firmware code to be executed. If not, no further testing is necessary for this requirement.

TB16.2 The tester shall analyze the vendor’s measures that ensure that the device enforces the separation between applications with security impact from those without security impacts. The tester will verify that it is not possible that one application interferes with or tampers another application or the OS/firmware of the device, especially to access, use or modify data objects belonging to another application, even if they are distributed over separate components of the device.

TB16.3 The tester shall note whether the device is designed to allow for non-firmware applications to be executed, and what firmware functions are provided by the processor on which such non-firmware applications would execute (for example, PIN processing, cryptographic-key operations, etc.).

TB16.4 **I**f the OS/firmware and/or any application with security impact are distributed over separate components (i.e., sets of hardware, software, firmware, or some combination thereof that implement cryptographic functions or processes that are contained within independent defined cryptographic boundaries) of the device, the tester will verify that the communication in between separated parts is consistent with the separation mechanisms as described by the vendor. The vendor shall provide evidence concerning the communication between the separated parts and how the communication protocols maintain the separation of applications with security impact from those without security impacts.

TB16.5 If the device allows customers or integrators to install additional applications, the tester will verify that the device’s design prevents the embedded application from:

- Having access to the top-level master keys that protect the working keys—i.e., the application cannot extract or modify the top-level master key.
- Having access to operator or security officer functions, and so cannot change security configurations or change privileges.
- Introducing new primitive cryptographic functions (although it can use these to implement new composite functionality).

Additionally, the embedded application is separated from the approved device functionality by an internal security boundary that prevents embedded applications from obtaining any elevated privilege or access to any data belonging to other embedded or host-side applications.

TB16.6 The tester shall detail what mechanisms exist within the device to allow for the execution of non-ROM-based configuration or program data—for example, processors, micro-controllers, FPGAs, etc. The tester shall note which of these mechanisms execute code that has been considered in-scope (code that impacts these security requirements) of the evaluation, and which do not.

TB16.7 If the device relies upon the use of different processors to provide for the separation between the firmware and any applications, the tester shall review and briefly describe the method of communications provided between these processors, including any physical interface and API(s).

TB16.8 If the device allows for different applications to be executed on the same processor, or for the execution of one or more applications on the same processor that is used to execute firmware, the tester shall detail the mechanisms provided to ensure that code and data objects of different applications/firmware are kept separate.

TB16.9 The tester shall review the configuration of the mechanisms described above and confirm that it maintains application separation.

TB16.10 The tester shall note whether the firmware processor(s) provide mechanisms to prevent the execution of memory used to hold data objects. Where such mechanisms exist, the tester shall detail whether they are utilized correctly by the firmware.

TB16.11 The tester shall verify that clear security guidance for the development and implementation of the aforementioned additional applications is included in the security policy cited in C1. This guidance at a minimum must define procedural controls to ensure that the applications are properly reviewed, tested, and authorized.

DTR B17 Minimal Configuration

The operating system/firmware of the device must contain only the software (components and services) necessary for the intended operation. The operating system/firmware must be configured securely and run with least privilege.

Guidance

The intended operation is considered as the functionality relevant to B2 and is representative of the functionality provided by the device. For multi-application devices, the intended operation furthermore includes the operation of applications without security impacts.

OS/firmware modules such as, for example, peripheral drivers, file systems, or inter-process communication protocols shall be regarded as components. Applications responding to external interfaces or communicating with the firmware shall be regarded as services.

Firmware and software running on the device shall be designed to run with minimal privilege and ensure that no process requires root (or equivalent) privilege following startup. Additionally, only software necessary for the intended purpose of the device shall be present. Least privilege requires that only those components and services that are required to have access to sensitive information, functions, and/or peripherals be permitted to have such access.

Vendors should provide configuration lists and software specifications to support their answers.

Vendor guidance includes configuration options for specific use cases, i.e., what are the set of commands required or not required.

For the purposes of this requirement, sensitive information, functions, and peripherals, include any method that may provide access to clear-text cryptographic keys and components, customer PINs, passwords/authentication codes used for entry into sensitive states, or other items of information or configuration that is required to meet the core logical and physical requirements.

TB17.1 The tester shall verify that all components and services indicated in the configuration list are necessary for the intended operation. For that purpose, the vendor shall provide a configuration list, which shows all OS/firmware components and other software with an explanation of the purpose and a rationale for its presence.

TB17.2 The tester shall verify that the security policy enforced by the device does not allow unauthorized or unnecessary functions.

TB17.3 The tester shall verify that API functionality and commands that are not required to support specific functionality are removable whenever possible or disabled if removal is not feasible.

TB17.4 The tester will examine the methods and tools provided by the vendor, which ensure that the intended software configuration of the device is maintained and that updates and release changes do not affect the secure configuration of the OS.

TB17.5 The tester shall note whether the device implements a commercial operating system, custom operating system, function executive, or other mechanism. If the device uses a commercial operating system, the tester shall note the name and version of this system.

The tester shall verify that the operating system is enforcing least privilege.

TB17.6 If the device uses a commercial operating system, the tester shall review publicly available sources of vulnerability information and note whether any vulnerabilities exist for this system. The tester shall note the sources reviewed and any potential vulnerabilities found and justify why any such vulnerabilities are mitigated by the vendor configuration(s).

TB17.7 The tester shall obtain the configuration information for the operating system used in the device. The tester shall compare this configuration with the supplied documentation and note whether they agree or have differences. If differences are detected, it is necessary to address why these occur with regard to compliance to this requirement.

TB17.8 The tester shall describe the testing and methodology used by the vendor to determine the functions necessary for the device's execution environment. The tester shall justify that this description sufficiently details the steps necessary to reduce the functionality of the device to only those components and services necessary for the intended operation of the device.

DTR B18 Unique Device ID

The device has the ability to return its unique device ID.

Guidance

Devices must each be uniquely identified through acceptable cryptographic methods for their authentication. Examples of appropriate algorithms and minimum key sizes are stated in Appendix D.

TB18.1 The tester shall examine and cite any additional documentation (e.g., API reference, design documentation, key-management specification) that describes the unique device ID.

TB18.2 The tester shall obtain the unique device ID from the device using the methods described in vendor documentation.

DTR B19 PCI Mode

Devices that are designed to include both a PCI mode and a non-PCI mode must not share secret or private keys between the two modes, must provide indication as to when the device is in PCI mode and not in PCI mode, and must require dual authentication when switching between the two modes.

Guidance

When switching between the two modes, keys may either be zeroized or mechanisms must be in place to prevent keys from being used in both modes. This does not apply to keys generated internal to an IC that are used only internal to the IC and are never output.

Internally generated keys that cannot be updated or output do not need to be zeroized and may be shared between the two modes if their only use is for internal storage protection.

Keys used to authenticate that it is a valid device or keys used to sign information like a device identifier may be shared between the two modes.

If the device allows the two modes to be selected remotely, the method used must enforce mutual authentication and prevent reply attacks.

Authentication data must include at least seven characters or an equivalent strength.

The mode indication may be either permanent (e.g., an indicator on the device or a prompt on a display) or transitory (e.g., an indication of the mode upon transition from one mode to the other and when querying the current device state), or both.

TB19.1 The tester shall review source code to verify that the device zeroizes keys or prevents keys from being shared between the two modes.

TB19.2 The tester shall verify that all conditions, as described above in the guidance, are met if the device implements a PCI mode and a non-PCI mode.

C – Policy and Procedures Derived Test Requirements

DTR C1 Security Policy

A user-available security policy from the vendor addresses the proper use of the device in a secure fashion, including information on key-management responsibilities, administrative responsibilities, device functionality, identification, and environmental requirements. The security policy must define the roles supported by the device and indicate the services available for each role in a deterministic tabular format. The device is capable of performing only its designed functions—i.e., there is no hidden functionality. The only approved functions performed by the device are those allowed by the policy.

Guidance

Devices evaluated in Section A – Physical Derived Security Requirements where the device will be restricted to deployment in environments meeting at least the security of a controlled environment as defined in ISO 13491-2 must stipulate those deployment restrictions in the device's security policy.

If the roles are user-configurable, the security policy must describe the roles that can be configured in the device.

The policy must include all configuration settings necessary to meet these security requirements.

Devices may include unapproved functionality. When such functions are in use, the device is considered to be operating in an unapproved mode and must provide a mode indication as described in B19.

The policy must include procedures for the decommissioning of devices that are removed from service, including the removal of all keying material that could be used to decrypt any sensitive data processed by the device. Procedures may differentiate between temporary and permanent removal.

Also see B14 for validation of allowed PIN translations.

An English-language version of the security policy must be made available for posting to PCI SSC.

TC1.1 The tester shall check that a security policy exists and is well-formatted, accurate, consistent, complete, and does not contain ambiguous or misleading information. Tester shall verify any URL in the security policy as being valid and usable.

TC1.2 The tester shall review the device security policy and note whether this clearly states the environments in which the device is restricted to use in environments meeting at least the security of controlled environments.

TC1.3 The tester shall confirm that the security policy includes all configuration settings necessary to meet the security requirements defined in this document.

TC1.4 The tester shall confirm that the security policy defines all CSPs or classes of CSPs stored or used in the module. The required information for predefined cryptographic keys includes key name or identifier, associated algorithm, length, usage, storage location, and erasure method. The required information for other predefined CSPs includes name or identifier, usage, storage location, erasure method, and any associated algorithms.

TC1.5 The tester shall confirm that the security policy contains specific details on the cryptographic algorithms (AES, TDES, SHA-2, etc.) and key-management methodologies supported by the device. It is not required that this be expressed as a key table as required by DTR B10, but it must detail the specific keys and usages of these keys for all key-management methods exposed to the device operators. Key-management operations that are only used within the device, or between integrated device components, are not required to be detailed.

TC1.6 The tester shall examine the security policy to verify that it states that keys should be replaced with new keys whenever the compromise of the original key is known or suspected, and whenever the time deemed feasible to determine the key by exhaustive attack elapses, as defined in NIST SP 800-57-1.

TC1.7 The tester shall examine the security policy to verify that it provides guidance on secure key archival storage to protect keys against unauthorized disclosure and substitution and to provide key separation**.**

The tester shall examine the security policy to verify that it describes how to verify security-relevant settings such as PIN block formats enabled and the commands in place, such as the ability to output clear-text PINs.

TC1.8 The tester shall examine the security policy to verify that instructions are provided with regard to tamper evidence, inspection frequency and procedures.

TC1.9 The tester shall examine the security policy to verify that instructions are provided with regard to auditing/log inspection frequency and procedures. The security policy shall include events that are logged as defined in B15, any restrictions on operator access to logs, and procedures for accessing and deleting logs.

TC1.10 The tester shall examine the security policy and relevant vendor documentation to verify that any periodic maintenance procedures required for the secure operation of the device are included in the security policy.

TC1.11 The tester shall examine the security policy and relevant vendor documentation to verify that documentation includes procedures for authentication of the device when received via shipping. Note that this may include visual or cryptographic methods.

TC1.12 The tester shall examine the security policy to verify that all physical interfaces are identified, along with the functionality and type of data that is carried over each interface.

TC1.13 If the device permits access to internal areas, the tester shall examine the security policy to verify that it specifies the internal components and operating procedures requiring access to these internal areas.

TC1.14 The tester shall examine the security policy to verify that it identifies all self-tests that the module performs, procedures that an operator may need to initiate any self-tests, and the conditions under which each self-test is executed (e.g., power up, periodic, conditional, on operator demand). This includes the time period and any conditions that may result in the interruption of the module's operations to perform the pre-operational or conditional self-tests. For example, if the module is performing mission-critical services that cannot be interrupted and the time period is passed for the initiation of the pre-conditional self-tests, the self-tests may be deferred to the next such time period. However, at all times the self-test must be performed on boot prior to any operational use, and within 2x the self-test cycle during continued use—i.e., never more than 48 hours since the last such test.

TC1.15 The tester shall examine the security policy to verify that it identifies all roles supported by the device and indicates the services and permissions available for each role.

TC1.16 The tester shall examine the security policy to verify that the device is properly identified. Product name, hardware version, and software version information should be included in the identification of the approved device. The tester shall validate that the security policy includes pictures of the device, and how to validate the hardware, firmware, and application versions.

TC1.17 The tester shall examine the security policy to verify that it identifies all conditions (e.g., voltage, humidity, temperature) that will cause environmental failure-protection (EFP) mechanisms to trigger.

TC1.18 The tester shall examine the security policy and other relevant documentation submitted by the vendor to verify that the security policy identifies any other operational-environment restrictions that must be met for the device to be operated in a secure manner (e.g., ambient-temperature range, support hardware, support software, power conditioning, environmental protections assumed for proper operation).

TC1.19 The tester shall examine the security policy and other relevant documentation submitted by the vendor to verify that the security policy can be implemented to support the following configuration and that the implementation is easily identifiable in reviewing system settings.

This table illustrates restrictions on translations between PIN block formats that are applicable when the HSM does not enforce unique-key-per-transaction encryption for the resulting PIN block:

Translation			
To → / **From ↓**	**ISO Format 0, 3, 4**	**ISO Format 1**	**ISO Format 2**
ISO Format 0, 3, 4	– Permitted anywhere without change of PAN – Change of PAN only permitted in sensitive state for card issuance – Change of PAN token to real PAN only permitted with cryptographic binding of PAN token to real PAN	Not permitted	Permitted for submission to an IC card
ISO Format 1	Permitted	Permitted	Permitted for submission to an IC card
ISO Format 2	Not permitted	Not permitted	Permitted for submission to an IC card

TC1.20 The tester shall configure the device according to the policy and confirm that only functions allowed in the policy can be executed when in approved mode.

TC1.21 The tester shall confirm that the security policy includes procedures for the decommissioning of devices that are removed from service, including the removal of all keying material that could be used to decrypt any sensitive data processed by the device. The procedures may differentiate between temporary and permanent removal.

TC1.22 The tester shall confirm the security policy of the device and confirm that it clearly outlines the exact details of the key-management systems supported by the device—i.e., simply using the term “MK/SK” is not sufficient—and specifies that use of the device with different key-management systems will invalidate any PCI-approval of this device.

TC1.23 The tester shall confirm that the security policy contains specific details on how key loading must be performed for operation of the device. This must include any requirements for dual control and split knowledge, as required by DTR B6 and assessed by the PTS laboratory.

TC1.24 The tester shall confirm that the security policy contains specific details on how to change any default values, including passwords/authentication codes and certificates.

TC1.25 The tester shall confirm that the security policy contains information on how the device will indicate a tamper-response event, and any requirements for the return of this device to the vendor for examination following such an event (as required for compliance to DTR A1).

TC1.26 The tester shall examine the security policy and relevant vendor documentation to verify that the device has update and patch procedures required for the secure operation of the device and that the procedures are included in the security policy. The policy will include both local and remote update and patch downloading procedures for software, firmware, and configuration parameters.

TC1.27 The tester shall confirm that the security policy includes any communication methods and protocols, including wireless, used by the device. Use of any method not listed in the policy invalidates the device approval.

TC1.28 For HSMs intended for cloud usage, the tester shall confirm that the security policy outlines how the HSM Solution is implemented, what cryptographic services are provided, how the user and components of the HSM Solution are authenticated, and how to securely disable or deprecate the storage of user keys from any or all components of that solution. Alternatively, the tester shall confirm that the information is in a separate publicly available security policy. See DTR K9.

DTR Module 2: Key-Loading Devices

D – Key-Loading Devices Derived Test Requirements

DTR D1 Secret or Asymmetric Key Pair Protection

If the device is capable of generating asymmetric key pairs and/or secret keys, the private or secret key or its precursors will not be visible in clear-text form at any time during the generation process.

TD1.1 The tester shall examine any relevant documentation submitted by the vendor, such as a user guide, the specification of the device's logical structure, or any other relevant documentation, to verify that it supports the vendor response that the secret and private key or its precursors, are not be visible in clear-text form at any time during the generation process.

TD1.2 The tester shall analyze the device to determine whether the secret or private keys or their precursors are visible in clear-text form at any time.

TD1.3 The tester shall generate secret and/or asymmetric key pairs to determine whether the secret and private key or its precursor is visible in clear-text form at any time during the generation process.

DTR D2 Keys are Deleted Immediately after Transfer Process

If the device is capable of generating symmetric keys or asymmetric key pairs that are not used by the device, the key or key pair and all related secret and private seed elements are deleted immediately after the transfer process.

Guidance

The device cannot store unused keys, key pairs, or seed elements subsequent to completion of the transfer process. Once the transfer process is completed, the device deletes all related secret information, keys, key pairs, and seed elements.

TD2.1 The tester shall examine any relevant documentation submitted by the vendor, such as a user guide, the specification of the device's logical structure, or any other relevant documentation, to verify that the key or key pair and all related secret and private seed elements are deleted immediately after the transfer process for those keys not used by the device.

TD2.2 The tester shall analyze the device to determine that for key pairs that are not used by the device, the key or key pair and all related secret and private seed elements are deleted immediately after the transfer process

TD2.3 The tester shall generate secret and/or asymmetric key pairs and verify that the key or key pair and all related secret and private seed elements are deleted immediately after the transfer process.

DTR D3 Device Retains No Information on Previously Transferred Keys

The device retains no information that could disclose any key that the device has already transferred into another cryptographic device.

Guidance

This is not intended to preclude the following:

- *The use of the KLD for loading multiple HSMs with the same Master File Key (e.g., when the HSMs are used for load sharing with a single key database);*
- *The use of the KLD to generate unique keys per device, load them into a PED, and later transfer the file of keys to an HSM;*
- *The use of Base Derivation Keys or similar to generate initial DUKPT keys or similar UKPT schemes;*
- *The loading of the same public key to multiple devices;*
- *The use of Key-Encipherment Keys for importing/exporting of cryptograms.*

TD3.1 The tester shall examine any relevant documentation submitted by the vendor, such as a user guide, the specification of the device's logical structure, or any other relevant documentation, to verify that the device clears all information that could disclose any key previously transferred.

TD3.2 The tester shall review the source code of the device and confirm that the buffers and memory locations used for the (temporary or long-term) storage of cryptographic keys transferred to other devices are cleared upon the completion of that transfer process.

TD3.3 The tester shall use the device to transfer a cryptographic key into other systems/devices, then confirm that any subsequent key-transfer operations do not contain any traces of the previously transferred keys. Where possible, the tester shall use a smaller key for subsequent key-transfer operations—for example, transferring a 256 bit key and then transferring a 128 bit key—and confirm that the buffer does not contain key material of the larger initial key.

TD3.4 Where possible, the tester shall use the device to transfer a cryptographic key, then produce an output of some other type—such as an error message—from the device, and confirm that this subsequent output does not contain any key material from the initial key-transfer process.

DTR D4 Component Security

If the device is composed of several components, it is not possible to move a secret or private key within the device from an asset domain of higher security to an asset domain providing lower security.

Guidance

See Appendix F, "Domain-Based Asset Flow Analysis" for definitions of security domains within the HSM. Assets may be passed within components sharing the same or higher security domain, as long as all of the components within that domain meet the security requirements of that domain.

TD4.1 The tester shall examine any relevant documentation submitted by the vendor, such as a user guide, the specification of the device's logical structure, or any other relevant documentation, to verify that the vendor has detailed the transfer process of keys for devices composed of several components and that it is not possible to move a cryptographic key within the device from a component of higher security to a component providing lower security.

TD4.2 The tester shall attempt transfer a cryptographic key to a component providing lower security.

DTR D5 Modification of Functional Capabilities

Once the device has been loaded with cryptographic keys, there is no feasible way in which the functional capabilities of the device can be modified without causing the automatic and immediate erasure of the cryptographic keys stored within the device or causing the modification to be otherwise detected before the device is next used to load a key.

Guidance

This is not meant to preclude authenticated firmware changes.

TD5.1 The tester shall examine any relevant documentation submitted by the vendor, such as a user guide, the specification of the device's logical structure, or any other relevant documentation, to verify that it is not possible to modify the device's functional capabilities without either causing the erasure of cryptographic keys stored within the device or otherwise being detected before the device is next used to load a key.

TD5.2 The tester shall attempt to modify the device's functional capabilities and document how the device responds.

DTR Module 3: Remote Administration

E – Logical Security Derived Test Requirements

DTR E1 Remote Administration Operational Service

The device is designed in such a way that it cannot be put into operational service until the device initialization process has been completed. This will include all necessary keys and other relevant material needed to be loaded into it.

Guidance

The remote administration device cannot be put into an operational state until an initialization process has been completed and a secure relationship (i.e., cryptographic) has been established between the remote administration platform and the target device(s).

Remote admin connectivity must be terminated during the initialization sequence and can only be established after the initialization sequence is complete.

TE1.1 The tester shall verify that all operational services cease during the initialization processes.

TE1.2 The tester shall verify that no operational services are started until the initialization is complete.

TE1.3 The tester shall verify that remote admin connectivity is terminated during the initialization process and cannot be established until the initialization sequence is complete.

DTR E2 Operator Functions

The following operator functions that may influence the security of a device are permitted only when the device is in a sensitive state—i.e., under dual or multiple control:

- ***Disabling or enabling of device functions;***
- ***Change of passwords/authentication codes or data that enable the device to enter the sensitive state.***

The secure operator interface is so designed that entry of more than one password/authentication code (or some equivalent mechanism for dual or multiple control) is required in order to enter this sensitive state and that it is highly unlikely that the device can inadvertently be left in the sensitive state.

Guidance

Operator functions need to be controlled and defined. Certain operator functions can only be done when the device is in a sensitive state. A sensitive state is defined as to when the device is under dual or multiple control and that more than one password/authentication code is required to enter a sensitive state. To activate the secure operator interface, more than one password/authentication code or equivalent mechanism for dual or multiple control is required.

Passwords/authentication codes used for authentication are at least seven characters in length or an equivalent strength.

Sensitive state limits are consistent with B6.

The secure operator interface must be designed in such a way that the sensitive state cannot be left open when not in use or left open indefinitely.

TE2.1 The tester shall examine the supporting documentation submitted by the device vendor. The documents should be representative of a configuration control process that can be audited. These documents shall be referenced. The documentation could include firmware revision lists with updates documented, current operator manuals, and other materials that show clear evidence that the device is operated in a secure manner

TE2.2 The tester shall examine details provided by the vendor that the documented process explicitly addresses how configuration settings are changed, including the changing of the device’s passwords or other authentication data.

TE2.3 The tester shall verify and demonstrate that the device configuration settings can only be done when the device is in the sensitive state.

TE2.4 The tester shall verify that any change to device passwords or other authentication data that allows a device to enter a sensitive state happens under a sensitive state.

TE2.5 The tester shall attempt to initiate a configuration change and password or other authentication data change while the device is in other than a sensitive state.

TE2.6 The tester shall validate that the operator interface requires more than one password/authentication code or equivalent mechanism for dual or multiple control to enter the sensitive state.

TE2.7 The tester shall verify that the device cannot be left in a sensitive state by putting the device into a sensitive state and leaving the connection open to determine whether the device closes the sensitive state.

TE2.8 The tester shall attempt to exceed the vendor-specified limit on the number of function calls (services) and time limit. Once the limit is exceeded, the tester shall verify that the device has returned to its normal state and requires re-authentication.

F – Devices with Message Authentication Functionality Derived Test Requirements

DTR F1 Message Authentication Device

If the message authentication device can be manually activated and can contain different MAC keys, the identity of the key used is displayed by the device. The device only outputs a confirmation or denial of a MAC provided for verification, never the clear-text-computed MAC.

Guidance

For devices that manually activate the MAC keys, the identity of the key is displayed on the device to ensure the correct key is used if multiple keys are present in the device.

Clear-text-computed MACs should never be outputted on the denial or confirmation of the MAC.

TF1.1 The tester shall determine from vendor documentation that the message authentication device can be manually activated and the identity of the key used is displayed by the device.

TF1.2 The tester shall verify that the device only outputs a confirmation or denial during a MAC verification and that the clear-text MAC is never output.

DTR F2 MAC Key Generation and Verification

The length of the MAC being generated or verified is in accordance with ISO 16609.

Guidance

One of the following as defined in ISO 16609 must be used: MAC algorithm 1 using padding method 3, or MAC algorithm 5 using padding method 4.

TF2.1 The tester shall examine and cite any relevant documentation (e.g., design documentation, key-management specification) that describes the MAC length.

TF2.2 The tester shall list the MAC length values used by the device and all values supports.

TF2.3 The tester shall verify and list all MAC generation algorithms used by the device.

DTR F3 Dual MAC Key Generation and Verification

If the device uses two keys for MAC generation or verification, the technique utilized is in accordance with ISO 16609.

Guidance

One of the following should be used: MAC algorithm 1 using padding method 3, or MAC algorithm 5 using padding method 4.

TF3.1 The tester shall examine and cite any additional documentation (e.g., design documentation, key-management specification) submitted by the vendor that describes the MAC generation and verification.

TF3.2 The tester shall verify that the MAC generation and verification techniques meet ISO 16609.

TF3.3 The tester shall list the techniques used for MAC generation and verification.

DTR F4 Unidirectional MAC Keys

If the message authentication device is designed to use unidirectional MAC keys, a MAC key is only used for one type of MAC function—i.e., verify the MAC of received text or generate and output a MAC for a text being transmitted.

Guidance

MAC keys shall be used for only one function such as verify the MAC of received text or generate and output a MAC for a text being transmitted.

TF4.1 The tester shall examine and cite any additional documentation (e.g., design documentation, key-management specification) that describes unidirectional MAC keys.

TF4.2 The tester shall verify that in the use of unidirectional MAC keys, they are used for one type of MAC function only. The tester shall try multiple functions and document the results.

TF4.3 The tester shall list all MAC functions to ensure that each MAC key is used for a single function.

G – Devices with Key-Generation Functionality Derived Test Requirements

DTR G1 Removal Detection

Unauthorized removal of the device from its operational location is deterred by one or more of the following mechanisms:

- ***The device includes mechanisms such that the removal of the device from its operational location will cause the automatic erasure of the cryptographic keys contained within the device; or***
- ***Removal of the device would be of no benefit because its tamper-resistance or tamper-responsive characteristics ensure that the extraction of cryptographic keys or other secret data is not feasible.***

TG1.1 The tester shall examine and cite any relevant documentation—such as the user guide, specification of the device's logical structure, installation guide—submitted by the vendor to verify that the device has protections from unauthorized removal and meets at least one of the outlined requirements.

TG1.2 The tester shall note which techniques are employed by the vendor to meet the requirement.

DTR G2 Clear-text Key Output

The device will not output any clear-text key except under dual control. Such dual control is enforced by means such as the following:

- ***The device requires that at least two passwords/authentication codes be correctly entered within a period of no more than five minutes before the device will output a key.***
- ***The device requires that at least two different, physical keys (marked "not to be commercially reproduced") be concurrently inserted in the unit before it will output a key.***

TG2.1 The tester shall examine and cite any documentation (e.g., key-management specification, operator manual) that describes the techniques used to export clear-text keys.

TG2.2 The tester shall verify the techniques used to export clear-text keys meet the requirement of dual control as described in vendor documentation.

DTR G3 Special Operator Functions

The following operator functions (if available) require the use of sensitive states:

- ***Manual input of control data—e.g., key-verification code—to enable export, import or use of a key; and***
- ***Permitting movement of the device without activating a key-erasure mechanism.***

TG3.1 The tester shall examine and cite any documentation (e.g., key-management specification, operator manual) that describes the special operator functions that place the device into a sensitive state,

TG3.2 The tester shall verify the device is placed in a sensitive state when permitting movement of the device without activating a key-erasure mechanism as described in vendor documentation.

TG3.3 The tester shall verify the device is placed in a sensitive state when permitting manual input of control data (e.g., key-verification code) to enable export, import or use of a key, as described in vendor documentation.

DTR G4 Proprietary Functions

Any proprietary functions are either:

- ***Totally equivalent to a series of standard and approved functions; or***
- ***Limited to use only keys that, by virtue of key separation, cannot be used with keys, or modified keys, of non-proprietary functions.***

TG4.1 The tester shall examine and cite any documentation (e.g., key-management specification, operator manual) that describes any proprietary functions in the device.

TG4.2 The tester shall verify that any proprietary function is totally equivalent to a series of standard and approved functions or is limited to use only keys that, by virtue of key separation, cannot be used with keys, or modified keys, of non-proprietary functions.

H – Devices with Digital Signature Functionality Derived Test Requirements

DTR H1 Private Key Management

The private key is managed such that:

- ***The asymmetric private and public key pair is generated within the digital signature device; and***
- ***The asymmetric private key is only exported outside the original digital signature device under dual control; and***
- ***Mechanisms for the control of the use of the private key are provided.***

TH1.1 The tester shall examine and cite any relevant documentation—such as a user guide, operator manual, key-management guidance documents)—submitted by the vendor

.**TH1.2** The tester shall verify whether the asymmetric private and public key pair is generated within the digital signature device.

TH1.3 The tester shall verify that the asymmetric private key is not exported outside the original digital signature device except under dual control for backup and archival purposes.

TH1.4 The tester shall verify that mechanisms for the control of the use of the private key are provided.

DTR H2 Device Management for Digital Signature Verification

For audit and control purposes, the binding between the public key and the identity of the owner of the private key is readily determined by:

- ***Use of public key certificates, where the public key certificate was obtained from an authorized certificate authority—e.g., the vendor PKI; or***
- ***Use of public key certificates and appropriate certificate management procedures; or***
- ***Other equivalent mechanisms to irrefutably determine the identity of the owner of the corresponding private key.***

TH2.1 The tester shall examine and cite any relevant documentation—such as a user guide, operator manual, key-management guidance document—submitted by the vendor.

TH2.2 The tester shall verify that the binding between the public key and the identity of the owner of the private key is readily determined by one of the following:

- Use of public key certificates, where the public key certificate was obtained from an authorized certificate authority;
- Use of public key certificates and appropriate certificate management procedures; or
- Other equivalent mechanisms to irrefutably determine the identity of the owner of the corresponding private key.

DTR Module 4: Cloud-Based HSMs as a Service – Multi-tenant Usage and Remote Management Security Requirements

I – Cloud Physical Security Derived Test Requirements

DTR I1 HSM Processing Element

The HSM processing element must meet all PCI HSM physical and logical requirements, including protection of sensitive data to an attack potential of at least 35 per HSM processing element for identification and initial exploitation, with a minimum of 15 for initial exploitation, as defined in Appendix A.

TI1.1 The tester shall examine the Asset Flow Analysis of the HSM Solution, as provided by the HSM Solution Provider, and confirm that it details the flow, storage, and processing areas for all secret and private cryptographic keys used in the solution. If the HSM processing element is previously approved to PCI HSM requirements, the tester shall quote the approval number and confirm that the hardware and firmware versions remain the same.

TI1.2 The tester shall confirm the HSM processing element(s) are clearly defined within the solution, referencing the Asset Flow Analysis to show this is included correctly.

TI1.3 The tester shall confirm through their understanding of the HSM processing element(s) reviewed during testing that the Asset Flow Analysis is correct and complete, providing an accurate picture of how secret and private keys and key components are managed by the system.

TI1.4 The tester shall confirm that the HSM processing element definition includes a physical boundary and that no clear-text cryptographic keys or key components are accessible outside of this boundary.

TI1.5 The tester shall confirm that the HSM processing element physical boundary, and all firmware code therein, has been assessed as compliant to all PCI HSM physical and logical security requirements, including protection of sensitive data to an attack potential of at least 35 per HSM processing element for identification and initial exploitation, with a minimum of 15 for initial exploitation, as defined in Appendix A.

TI1.6 The tester shall confirm that compliance to the PCI HSM physical and logical security requirements is provided entirely by the features of the HSM device defined by the physical boundary, and that there is no reliance on the security of the environment or procedural controls to achieve a protection level of at least 35 per HSM processing element for identification and initial exploitation, with a minimum of 15 for initial exploitation, as defined in Appendix A.

DTR I2 HSM Virtualization System

HSM virtualization systems must either:

- ***Be installed and operated in an environment meeting at least the security requirements of a controlled environment, or***
- ***Protect the stored and processed sensitive data to an attack potential of at least 26 per HSM virtualization system for identification and initial exploitation, with a minimum of 13 for initial exploitation, as defined in Appendix A.***

Guidance

Sensitive data in the context of this requirement is considered to be any data, firmware, scripts, or other values (such as passwords) that may be relied upon to meet the requirements of this standard. Examples include the code/firmware of the virtualization system itself, cryptographic keys used to terminate or authenticate interaction with HSM Solution Providers, secure channels, etc.

TI2.1 The tester shall examine the Asset Flow Analysis of the HSM Solution, as provided by the HSM Solution Provider, and confirm that it details the flow, storage, and processing areas for all aspects of the solution required for the HSM virtualization system. The implementation and scope must include any features relied upon for compliance to other requirements within this standard that are not otherwise implemented in the HSM processing element.

TI2.2 The tester shall confirm the HSM virtualization system(s) are clearly defined with the HSM Solution, referencing the Asset Flow Analysis to show this is included correctly.

The tester shall confirm through their understanding of the HSM virtualization system(s) reviewed during testing that the Asset Flow Analysis is correct and complete, providing an accurate picture of how the HSM Solution is interfaced to, provisioned, managed, and overall meets the requirements of this standard.

TI2.3 Where any part of the HSM virtualization system(s) are not implemented within a physical system that is tamper responsive, the tester shall confirm that the entire HSM virtualization system is implemented and managed within an environment meeting at least the security requirements of a controlled environment.

TI2.4 If the HSM virtualization system relies upon an environment meeting at least the security requirements of a controlled environment for compliance, the tester shall confirm that this environment has been audited and validated to meet at least the security requirements of a controlled environment as outlined in ISO13491-2, by a PCI-approved QSA or member of PCI lab staff.

TI2.5 If the HSM virtualization system relies upon a physical system that is tamper responsive, the tester shall develop attack methods to bypass the tamper-responsive features and obtain access to sensitive data processed by this HSM virtualization system—such as TLS private or secret keys, used for the termination of a secure channel. The tester shall provide a costing for the most feasible of these attacks.

If a scenario can be developed that requires an attack potential of less than 26 per HSM virtualization system for identification and initial exploitation or an initial exploitation of less than 13, the tester shall confirm that the HSM documentation, including the security policy, requires that the HSM is deployed in an environment meeting at least the security of a controlled environment.

DTR I3 HSM Virtualization System and HSM Processing Element Interaction

HSM virtualization systems that provide for switching/routing of secure channels between the HSM Solution Consumer and one or more HSM processing elements, must manage the cryptographic keys and operations used for these functions within a tamper-responsive system (to attack potential of at least 26 per HSM virtualization system for identification and initial exploitation, with a minimum of 13 for initial exploitation, as defined in Appendix A).

Guidance

A secure channel must exist to ensure the security of the use of cryptographic keys, commands, and other interactions with the HSM Solution. This secure channel may be implemented through physical means, such as within a tamper-responsive boundary, or by logical means such as a cryptographically secure connection using a protocol such as TLS. Requirement K1 covers the testing that must be performed to validate the secure channels implemented in the solution.

This requirement is for situations where the HSM virtualization system may act as a switch or broker of commands from the HSM Solution Consumer, routing these to HSM processing elements based on capacity or other factors. In such a case, the logical secure channel to the HSM Solution Consumer may terminate at the HSM virtualization system, and therefore the HSM virtualization system will be responsible for securing the onward authenticity of those connections (through the establishment or use of another secure channel to the HSM processing element(s)).

The intent of the attack costing for this requirement is to ascertain the feasibility of attacks aiming at determining or subverting the cryptographic keys or operations used to route secure channels between an HSM Solution Consumer and one or more HSM processing elements.

TI3.1 The tester shall confirm whether the HSM Solution provides for switching and/or routing of secure channels, commands, or messages from the HSM Solution Consumer to individual HSM processing elements, and confirm that this is correctly documented in the Asset Flow Analysis. Where each user connection to an HSM Solution Consumer is individually and uniquely connected directly to an HSM processing element and no switching and/or routing of connections to or between HSM processing elements is possible once a secure channel is established, no further testing is required for this DTR.

TI3.2 The tester shall detail how each secure channel is established from an HSM Solution Consumer to an individual HSM processing element. The tester may reference their responses in the secure channel requirements in response to this answer.

TI3.3 For each point at which the secure channel is terminated, or where cryptographic keys used in the secure channel are generated, managed, operated upon, or present for any reason without cryptographic protections, the tester shall confirm that a physical tamper-responsive system is implemented for these operations.

TI3.4 For each point at which the secure channel is switched and/or routed without the direct consent or interaction of the affected HSM Solution Consumer, the tester shall detail how this is implemented.

TI3.5 The tester shall provide a costing for the most feasible of these attacks. If an attack scenario can be developed that requires an attack potential of less than 26 per HSM virtualization system for identification and initial exploitation or less than 13 for initial exploitation, as defined in Appendix A, this requirement shall be marked as Not Verified.

DTR I4 HSM Virtualization System and HSM Processing Element Environment

***Where the HSM processing element and HSM virtualization system are contained in the same physical execution environment, the HSM virtualization system must meet all HSM processing element security requirements*.**

Guidance

This requirement is designed for solutions where the HSM virtualization system and HSM processing element are contained on the physical execution environment, such as when a single processor is used to implement both systems. If the HSM virtualization system is contained within the same physical enclosure but not executed on the same processor, the majority of the testing in this requirement does not apply. However, in all cases a response is required detailing how the HSM virtualization system and HSM processing element are implemented such that they are kept physically separate.

TI4.1 The tester shall detail how the HSM virtualization system(s) and the HSM processing element(s) are implemented within the solution. This must clearly show where code is executed, memory locations where sensitive data is maintained, and the security boundaries of each system. Where the HSM virtualization system can be clearly shown to execute in a separate physical execution environment, such that it is not possible for the virtualization code to have any access to clear-text keys or components, no further testing is required for this DTR.

TI4.2 Where the HSM virtualization system(s) and the HSM processing element(s) cannot be shown to have clear and physically distinct processing and memory storage areas, the tester shall confirm that the HSM virtualization system has been considered in scope for all testing requirements assessed against the HSM processing element(s), and that the findings confirm compliance in all areas.

DTR I5 HSM Processing Element Key Management

The HSM processing element must ensure that clear-text secret and private keys are processed in execution paths and memory areas that are isolated from keys of any other HSM Solution Consumer and/or code that is not included in the scope of the HSM processing element evaluation.

Guidance

Strong isolation between the cryptographic keys of different HSM Solution Consumers and non-firmware code is required to help mitigate cache and execution path side-channel attacks in HSM Solution environments. Such attacks may occur even if the HSM processing element is only able to execute signed firmware through exploitation of vulnerabilities that allow for local execution on those processing elements.

Any code, scripts, or customized functions not included in the scope of the PCI HSM evaluation must be implemented outside the scope of approval for the HSM processing element and must not directly manipulate or handle clear-text cryptographic keys. Only chaining or use of existing HSM commands is permitted. Commands that require custom cryptographic processing must be implemented through updates to the firmware of the HSM only. Chaining of existing HSM commands into a single atomic command must be performed externally to the HSM processing element or implemented in firmware approved under these requirements.

The intent is to ensure that the creation of customized functions or special operations does not allow for code loaded into the HSM processing element execution environment.

Atomic command is a single command. Common to have custom firmware loaded into an HSM to chain commands. E.g., a "check MAC" command may be a chaining of "start MAC," "continue MAC," and "end MAC," or could have a command that is "translate message," which decrypts the message block, checks the MAC, translates the PIN Block, re-encrypts the PIN block, and adds a new MAC for transmission onto the next hop in the acquiring chain.

TI5.1 The tester shall detail where all clear-text secret and private cryptographic keys are processed or stored, including any key components, referring back to the asset flow diagram. This must consider temporary storage during execution, in register and cache memory, as well as external storage.

TI5.2 The tester shall note whether the HSM Solution supports the processing of more than one set of clear-text keys at any one time within the same HSM processing element—i.e., where processing on another key may begin prior to the execution of the buffer-clearing functions used by the HSM to erase key material. Where this is the case, this requirement shall be marked as non-compliant.

TI5.3 The tester shall note whether the HSM Solution supports or implements code that is not included within the scope of firmware and assessment under the HSM processing element requirements, such as customer scripts or application code Where this is the case, this requirement shall be marked as non-compliant. I.e., you cannot have customer code in the HSM processing element. Any code in the HSM processing element is firmware and must be assessed as part of these requirements.

J – Cloud Logical Security Derived Test Requirements

DTR J1 Cryptographic Key Import/Export

It must not be possible to import or export the keys of an HSM Solution Consumer from the HSM processing element without approval from the HSM Solution Consumer. The approval must be cryptographically authenticated.

Guidance

"Key export," in the context of this requirement, refers to the output of clear-text secret and private keys, clear-text key components for such keys, or encrypted values under a key that is not in the direct control of the user and/or is not an operational key that is known only to the HSM, such as an HSM processing element storage key. Key export includes transmission of keys between HSM processing elements.

"Key import" refers to provisioning or loading new keys owned by the HSM Solution Consumer into the HSM Solution. This includes those transferred during provisioning processes.

"Keys of the HSM Solution Consumer" in the context of this requirement includes both working keys and HSM Solution Consumer master keys (which may be separate from the tamper keys for any individual HSM processing element).

This requirement covers all operations that may be used to import, export, or otherwise convey HSM Solution Consumer keys. This includes dedicated APIs for key loading and export, but also administration operations such as backup or restore functions and synchronization of HSM Solution Consumer keys across multiple HSM processing elements—e.g., for the purposes of load balancing or fail-over.

It is not the intent of this requirement to prevent the export or sharing of HSM Solution Consumer keys between multiple HSM processing elements, but to ensure that any such export or sharing only occurs with the express permission of the key owner(s). The method of providing permission must be more than purely procedural, there must be a cryptographically verifiable authorization from the HSM Solution Consumer permitting the key export or sharing. This authorization either may be performed for each occasion the export or sharing occurs, or may be a "blanket" approval covering multiple events and/or HSM processing elements.

TJ1.1 The tester shall detail what methods, if any, there are for the import and export of cryptographic keys from the solution. This must be based on review of vendor documentation, as well as functional testing of the solution.

TJ1.2 The tester shall review the API handling code of the HSM processing element to confirm there are no hidden or undocumented functions that allow for import or export of cryptographic keys. This review should consider only commands which may not have been already considered under standard PCI HSM testing.

TJ1.3 Where any key-transfer (import/export) functions are provided by the HSM, the tester shall detail how this transfer is performed, and what methods are used to ensure that the HSM Solution Consumer has authorized the key-transfer process.

TJ1.4 The tester shall confirm that the user authorization process involves a cryptographically secure method that verifies the identity of the HSM Solution Consumer and is resistant to man-in-the-middle, pre-play, and replay attacks.

TJ1.5 The tester shall confirm that all cryptography relied upon for compliance to this requirement is detailed in the HSM key table and enforces compliant cryptography and key lengths.

DTR J2 Identification and Authentication Methodology

Each HSM processing element must require the use of cryptographic methods for identification and authentication prior to the provisioning of secret keys, or establishment of any logical secure channel.

TJ2.1 The tester shall detail the provisioning process involved in assigning HSM Solution Consumer keys to a specific HSM processing element. This must include all provisioning methods supported, including both direct provisioning by the customer, as well as any automated provisioning processes implemented between individual HSM processing elements.

TJ2.2 The tester shall detail each secure channel that can be implemented as a connection to or from the HSM processing element. This must include both direct connections from HSM Solution Consumers, as well as any switched or re-routed secure channels passed through another part of the HSM Solution.

TJ2.3 The tester shall document how authentication is implemented for each of the provisioning and secure channel establishment methods listed above. In each case the tester shall confirm that it implements cryptographically sound methods resistant to man-in-the-middle, pre-play, and re-play attacks.

TJ2.4 The tester shall confirm that all cryptography relied upon for compliance to this requirement is detailed in the HSM key table and enforces compliant cryptography and key lengths.

DTR J3 Operations Performed with HSM Solution Consumer Keys

Operations performed with the cryptographic keys of an HSM Solution Consumer must require a cryptographically verifiable approval or request from the key owner.

TJ3.1 The tester shall detail the commands exposed by the HSM processing element and note which of these allow for operations to be performed with cryptographic keys owned by the HSM Solution Consumer.

TJ3.2 For each of these commands, the tester shall confirm there are cryptographic methods implemented to authenticate the request from the key owner. The tester shall note how these methods authenticate the HSM Solution Consumer as the owner of the keys being operated upon.

TJ3.3 The tester shall verify that these methods are resistant to man-in-the-middle, pre-play, and re-play attacks.

TJ3.4 Where a secure channel is relied upon for some aspect of the authentication process, or otherwise some form of compliance to this requirement, the tester shall confirm that the secure channel implements mutual authentication.

TJ3.5 The tester shall confirm that all cryptography relied upon for compliance to this requirement is detailed in the HSM key table and enforces compliant cryptography and key lengths.

DTR J4 Clear-text Sensitive Data

Clear-text sensitive data, including PINs and secret/private cryptographic keys, must never be exposed outside of an HSM processing element. Clear-text cardholder data, including PAN data, must be managed within an HSM processing element or an environment compliant to PCI DSS.

Guidance

This requirement does not include keys that are used as part of a secure channel that does not terminate onto the HSM processing element itself.

Clear-text PAN data must be processed entirely within the HSM processing element, or the Cloud HSM environment in which the PAN data is exposed in clear text must meet the requirements of PCI DSS. If the HSM Solution is not intended to handle clear-text PAN data, this must be explicitly noted in the security policy, including informing the HSM Solution Consumer that any use of PANs in this system may impact their PCI DSS compliance.

The term "PAN" is used in this requirement to cover any situation where a PAN may be involved, including use of PANs in PIN blocks, within track or track-equivalent data, etc. Solutions aiming to remove the use of the PAN through PAN tokens or similar must make this clear in the security policy documentation.

TJ4.1 Referring to the asset flow diagram and the API list, the tester shall confirm where all clear-text sensitive data is operated, stored, and processed within the solution. Where any of this data is exposed outside of the tamper-responsive boundary of the HSM processing element, the tester shall mark this requirement as non-compliant.

TJ4.2 The tester shall note any secure channels that may be terminated on the HSM processing element and confirm that any secret or private keys used in this secure channel are included in this assessment.

TJ4.3 The tester shall confirm that there are no methods that allow for the export of sensitive data under a key that is not directly owned by the customer of the HSM Solution or is not a provisioning or storage key directly managed by the HSM processing element.

TJ4.4 The tester shall review the HSM API documentation and confirm whether there are any commands that could reasonably be expected to accept the input, or provide an output, of clear-text PAN data. In any such case, the tester shall confirm that the environment in which the HSM Solution is deployed is compliant to PCI DSS requirements.

TJ4.5 Where an HSM Solution does not provide explicit functions for the import or export of clear-text PAN data, and is not to be deployed within an environment compliant to PCI DSS controls, the tester shall confirm that the security policy provides clear messaging noting that use of this solution for handling PAN data could impact the PCI DSS compliance of the HSM Solution Consumer.

TJ4.6 Where an HSM Solution implements PAN tokens, or other methods for replacing PAN values for the purposes of protecting the clear-text PAN, the tester shall confirm that the security policy documents this use and the purposes for which it is implemented.

DTR J5 Key Management for HSM Solution Consumers

All key-management operations involving the cryptographic keys of HSM Solution Consumers, including validation of key wrapping, must be performed within the HSM processing element.

Guidance

There may be more commands exposed to the HSM Solution Consumer other than those that are actually implemented by the HSM processing element. All key-management operations must be implemented by and within the HSM processing element, not by any other part of the overall HSM Solution.

Key-management operations may include use of public keys or defined operations on data within encrypted structures; therefore, this requirement remains distinct from the requirement that all secret/private keys are never exposed outside of an HSM processing element.

TJ5.1 The tester shall review the API provided to the customer and reference how commands are implemented and managed between the HSM virtualization system and HSM processing element.

TJ5.2 The tester shall confirm that all key-management operations are performed within the tamper-responsive boundary of the HSM processing element. This must include all operations using public keys, as well as defined operations manipulating encrypted content which may contain sensitive data.

TJ5.3 The tester shall note how key wrapping is implemented by the solution, referring back to previous key-management findings where appropriate. The tester shall confirm that all key-wrapping and key-unwrapping processes are performed within the tamper-responsive boundary of the HSM processing element.

DTR J6 HSM Processing Element Storage Areas

All HSM processing element storage areas, temporary or otherwise, must be cleared of sensitive data prior to allowing processing using a set of cryptographic keys belonging to another HSM Solution Consumer. This includes but is not limited to registers, cache, scratchpad memory, etc. Erasure must accommodate for any memory virtualization or wear-leveling implemented in the system.

Guidance

This requirement goes beyond what is normally required in HSM or POI systems, due to the expectation for shared environments where cryptographic keys for different HSM Solution Consumers may co-exist. It is expected that validation of this requirement will require a significant burden of evidence and testing for any system where the cryptographic keys of an HSM Solution Customer are exposed in clear text within a complex operating system or code where memory allocation and clearing is abstracted.

Examples of ways this requirement could be met are:

- *The HSM processing element is an ASIC (or interfaces to such) that has RTL level functions to manage clear-text keys (decrypt with a tamper key, perform crypto, clear keys before loading other keys).*
- *The HSM processing element does "touch" clear-text keys with high-level code but memory areas are provably cleared—e.g., through a reset process on that system—before loading other customer keys.*

TJ6.1 The tester shall detail the execution environment used by the HSM processing element(s) of the solution, specifically indicating the short- and long-term storage areas used during processing of clear-text cryptographic keys and components.

TJ6.2 The tester shall confirm whether the HSM Solution allows for multiple HSM Solution Consumers to share the execution environment of any single HSM processing element. This includes processing of this data at different times but without complete decommissioning and re-commissioning of the HSM processing element, or at the same time using different processor cores or execution paths.

TJ6.3 Where sharing of execution environments is possible, the tester shall confirm what methods are implemented to clear clear-text cryptographic key material from all short- and long-term storage areas of the HSM processing element. This may involve hardware or software methods but must accommodate for any and all virtualized memory, wear-leveling, internal registers, and data cache locations.

TJ6.4 For systems where it is not feasible to manage data clearance at the register level from software—such as when complex operating systems and high-level languages are involved—the tester shall confirm that hardware features are implemented. These may involve specific "data purge" commands to erase data paths, or power cycling of the HSM processing element prior to operation on the cryptographic keys of another HSM Solution Consumer.

TJ6.5 The tester shall review the source code or hardware diagrams of the data-clearance methods used to confirm that they are implemented as expected. Reverse engineering of object software to validate buffer-clearing code is not removed during compiling is required for any software methods implemented.

TJ6.6 The tester shall execute the buffer-clearing operation on the HSM processing element and confirm that it operates as expected.

DTR J7 HSM Solution Identification

Each HSM Solution must be able to provide a cryptographically verifiable unique ID that includes unique identification of the hardware and firmware versions of each HSM processing element and HSM virtualization system used in that solution. Any and all updates to the firmware must be logged and include references to previous ID numbers if the firmware updates these.

Guidance

Unique identification of the individual HSM processing elements is important in the case of a suspected compromise event, and to ensure correct revocation.

Firmware update logs must be authenticable by cryptographic methods using acceptable algorithms and key lengths. Best practice is to ensure that any firmware updates do not overwrite or update the HSM processing element or HSM virtualization system IDs—but in the cases where this does (or can) occur, the update log should ensure that there is a clear and verifiable chain of the ID values for any individual element or system. At no time should it be possible for an ID value to be lost or unable to be linked to an HSM element or system, nor for such an HSM element or system to have a period of time when its unique ID is not able to be determined.

TJ7.1 The tester shall review the API provided by the HSM Solution and confirm that there is a command to return the unique identification of the HW and FW versions of each HSM element and HSM virtualization system.

TJ7.2 The tester shall review the API provided by the HSM Solution and confirm there is a command to return a log of HSM processing elements used in the HSM Solution implemented for operating on the cryptographic keys of the HSM Solution Consumer.

TJ7.3 The tester shall confirm that the log is able to uniquely identify each HSM processing element and HSM virtualization system used, and the hardware and firmware details of each of these.

TJ7.4 The tester shall execute this command and confirm it works as expected.

TJ7.5 The tester shall update the firmware of an HSM processing element and confirm that this update is noted in the log.

TJ7.6 The tester shall obtain a copy of an HSM log and confirm that it is implemented as documented.

TJ7.7 The tester shall detail what cryptographic methods are implemented to authenticate the log. Methods must use acceptable algorithms and key lengths. Any MAC implementations used must be secure against length extension attacks.

DTR J8 HSM Processing Element Firmware Updates

Firmware updates for HSM processing elements must be signed using a dual-control process, using cryptographic keys maintained within a FIPS 140-2/3 level 3 or higher cryptographic module or a PCI-approved HSM. The firmware update method must prevent installation of any version of firmware older than the currently installed version unless all cryptographic keys are erased from the HSM processing element.

Guidance

Due to the online and elastic nature of HSM Solutions, signing of firmware images must be tightly controlled, hence additional security controls are required beyond standard PCI HSM implementations.

Installation of older firmware is acceptable if the installation is confirmed to erase all cryptographic keys from the HSM processing element.

TJ8.1 The tester shall confirm that there is a documented process for signing firmware updates for HSM processing elements, and this process requires the approval of at least two authorized individuals.

TJ8.2 The tester shall confirm that the authorization of each individual includes a method that can uniquely identify the individual(s) signing the firmware, and that either passwords/authentication codes are at least ten characters or an equivalent strength; or cryptographic methods (such as using a cryptographic challenge/response token) are implemented for identification purposes.

TJ8.3 The tester shall detail the methods used to prevent "roll back" of a previous firmware image. Where it is noted that installation of older firmware images is possible, the tester shall confirm that such installation will erase all cryptographic keys from the HSM processing element.

TJ8.4 The tester shall attempt to install a correctly signed firmware version that is identified as being older than the one currently in the HSM processing element under test. The tester shall verify that the operation is rejected or all cryptographic keys stored by the HSM are erased.

TJ8.5 The tester shall confirm that all requirements from DTR L3 are met.

K – Cloud Provisioning/Management Security Derived Test Requirements

DTR K1 Secure Channels

Connections between the HSM virtualization system, HSM processing elements, as well as to any HSM Solution Consumers, must implement a secure channel. The HSM Solution must support independent secure channels for interfaces to HSM Solution Providers and HSM Solution Consumers, including independent secure channels for each unique HSM Solution Consumer.

Guidance

A secure channel may be considered as provided through physical security by containment within the same tamper-responsive environment, use of a VPN-like cryptographic secure channel, or application/API level cryptographic controls that ensure the security of commands issued. Although independent secure channels are required for interface and routing of commands used by the HSM Solution Provider and HSM Solution Consumer, these may be provided by the same tamper-responsive functions, if two separate physical secure channels are used. Logical secure implementations must use different cryptographic keys for each channel.

It is not a requirement to have network connection to all components of the HSM Solution, but it is requiredthat the security of the connection from the HSM Solution Consumer or HSM Solution Provider can be validated across the whole path, including to the HSM processing element.

For any implementation, there must be either physical or logical separation of operational and administrative/management interfaces. Logical isolation must implement cryptographic controls for authenticity across all messages, and confidentiality for any sensitive data transmission.

TK1.1 The tester shall detail how separate interfaces to both the HSM Solution Provider and HSM Solution Consumer are implemented in the HSM Solution, specifically noting how commands are routed from HSM Solution Consumers and HSM Solution Providers to the separate HSM processing elements used in the solution.

TK1.2 The tester shall verify that each connection provides for a secure channel, implementing either logical or physical security controls (or a combination of both).

TK1.3 For secure channels (or parts thereof) which implement logical controls, the tester shall confirm that all messages are provided with controls which cryptographically authenticate each message, and confidentiality controls for any messages that may contain sensitive data.

TK1.4 For secure channels (or parts thereof) which implement physical controls, the tester shall verify that the secure channel path is fully protected by the tamper-responsive features of the HSM component, and attack methods to access these channels were considered during the assessment of those features.

TK1.5 The tester shall verify that there is logical or physical separation of secure channels used to connect to the HSM Solution Provider and HSM Solution Consumer.

TK1.6 The tester shall verify that there are separate secure channels for each HSM Solution Consumer.

TK1.7 The tester shall verify that logical isolation of secure channels is implemented through use of different cryptographic keys to establish and secure the channel.

TK1.8 The tester shall connect to each of the interfaces provided by the HSM Solution and confirm that the secure channels are implemented correctly.

DTR K2 Firmware Configuration

Updates to the HSM Solution that may impact the configuration or operation of that solution must be approved by the HSM Solution Consumers prior to implementing the change. If the HSM Solution Consumer is able to update or configure the firmware and/or settings of the HSM Solution with which they interface, such configuration must be isolated from any other HSM Solution Consumer.

Guidance

HSM Solutions may have multiple levels of firmware, each with different access and administrative controls. The intent of this requirement is to ensure that configurations or alterations to the HSM Solution are approved by the HSM Solution Consumers prior to deployment. This includes ensuring that changes made by any one HSM Solution Consumer do not affect the operation of the solution by another HSM Solution Consumer.

If the HSM Solution Provider is able, by means such as a hardware or firmware change, to make changes to the solution that may affect its security or operation, those changes must be communicated to and accepted by the HSM Solution Consumer prior to being implemented.

TK2.1 The tester shall detail how firmware, configurations, and settings are managed on the HSM Solution for each HSM Solution Consumer instance. This may include multiple sets of firmware and settings for each aspect of the overall solution.

TK2.2 The tester shall confirm whether the HSM Solution Provider is able, by means such as through firmware or hardware updates, to make changes to the HSM Solution that may affect the security or operation of the solution. Where this is possible, the tester shall confirm that there is a process in place to ensure that the HSM Solution Consumers are informed and able to provide consent to the changes prior to them being deployed.

TK2.3 The tester shall verify whether the HSM Solution Consumer is able to update any aspect of the HSM configuration they are using. If so, the tester shall confirm that such updates do not affect any other HSM Solution Consumers or their settings when using the HSM Solution.

TK2.4 The tester shall establish two HSM Solution Consumer instances on a single HSM processing element and change the settings for one of these HSM Solution Consumer instances. The tester shall confirm that the changes do not affect the second HSM Solution Consumer who is able to use that HSM processing element.

DTR K3 Configuration Options

Changes to HSM configuration options that may affect the compliance of an HSM Solution Customer must be cryptographically authenticated. Changes made by one HSM Solution Consumer must not affect the compliance status of any other HSM Solution Consumer.

Guidance

Authentication for any changes that affect the compliance of the solution must be performed using cryptographic controls that are additional to any secure channels implemented.

The intent of this requirement is to prevent one HSM Solution Consumer from changing a configuration that then affects the configurations or compliance status of other HSM Solution Consumers. Changes made by the entity operating the HSM Solution may affect many or even all HSM Solution Consumers, and although such changes should be communicated to HSM Solution Consumers prior to any potential impacts, these solution-level administration changes are out of scope of this requirement.

This requirement does not mandate that any HSM Solution must implement a "PCI mode," nor how that mode is to function if it is implemented. HSM Solutions may allow for the HSM Solution Consumer to force all processing to operate in a way that is compliant to the PCI HSM requirements, tag-specific commands as requiring PCI compliance, or other methodologies. However, at all times, cryptographic keys used in any way to ensure compliance to PCI HSM requirements must be separate from any keys that may be used in a way that is not compliant to these requirements.

TK3.1 The tester shall confirm whether any configuration options or settings exist which may affect the compliance of the HSM Solution. Where this is possible, the tester shall detail how the settings or configuration items may be changed.

TK3.2 The tester shall verify that any changes to settings or configuration options that may affect the compliance of the overall HSM Solution implement cryptographic authentication controls separate to those used in any secure channel.

TK3.3 The tester shall detail what cryptographic methods are implemented to authenticate the changes and confirm that these controls prevent man-in-the-middle, pre-play, and replay attacks. Methods must use acceptable algorithms and key lengths.

TK3.4 The tester shall perform some selected configuration and settings changes on the HSM Solution and confirm that the cryptographic authentication methods are correctly applied.

DTR K4 Provisioning Keys

The HSM processing element must establish a unique provisioning key for each HSM Solution Consumer. Key establishment must implement a key-agreement process, such as Diffie-Hellman, which provides perfect forward secrecy.

Guidance

This requirement is intended to be assessed for processes used to provision new high-level cryptographic keys for an HSM Solution Consumer to the HSM processing elements. It should not be assessed for any keys used in secure channels, nor for any working keys that are protected by the cryptographic keys of other HSM Solution Consumers.

TK4.1 The tester shall confirm, for each method of provisioning HSM Solution Consumer keys to an HSM processing element, that a transport key is established for the conveyance of the HSM Solution Consumer key.

TK4.2 The tester shall confirm that a unique transport key is established for each provisioning process, and that transport keys cannot be reused or shared between different HSM Solution Consumers or by the same HSM Solution Consumer between different provisioning processes.

TK4.3 The tester shall confirm that the process(es) used to establish the transport key implement the principles of perfect forward secrecy.

TK4.4 The tester shall confirm that the cryptographic keys of an HSM Solution Consumer are encrypted within a key-wrapping process to protect them during provisioning.

TK4.5 The tester shall confirm that the transport key(s) are erased using methods compliant with the HSM Solution buffer-clearing requirement once the provisioning process is complete, and no later than once the provisioned keys of the HSM Solution Consumer are able to be used for cryptographic operations (excluding any key-verification processes used to check the key is valid after loading).

TK4.6 The tester shall detail what cryptographic methods are implemented and confirm that these controls prevent man-in-the-middle, pre-play, and replay attacks. Methods must use acceptable algorithms and key lengths.

DTR K5 Tamper Keys

HSM processing element tamper keys used to protect the keys of an HSM Solution Consumer must be unique per HSM processing element. This key must be generated within the HSM processing element using an approved random number generation process and must not be exportable, or able to be disclosed, outside of the HSM by any means.

Guidance

This requirement is intended to be assessed for cryptographic keys used to extend the tamper-responsive boundary of the HSM processing element outside of the physical boundary of the HSM processing element itself—that is, tamper keys maintained within the HSM processing element that are erased on a tamper event. It should not be assessed for any keys used in secure channels, nor for any working keys that are protected by cryptographic keys of the HSM Solution Consumer.

Additionally, this requirement does not apply to master keys that may be distributed across a finite set of HSM processing elements, with the permission of the HSM Solution Consumer as per requirement J1, to achieve elastic processing during use. However, such master keys would be in scope of this requirement if used for storage of the cryptographic keys of more than one HSM Solution Consumer.

Cryptographic keys directly managed by the HSM Solution Consumer are not in scope of this requirement.

Solutions that use cryptographic keys of the HSM Solution Consumer as the HSM tamper key for that HSM Solution Consumer must ensure that these keys are not shared between different HSM Solution Consumers (except by chance, or through methods outside of the control of the HSM Solution Provider). It is not acceptable for an HSM Solution to implement a master or tamper key that is under the control of the HSM Solution Provider (where it is open to compromise through backup or loading procedures and may expose the keys of multiple HSM Solution Consumers using that solution).

This requirement does not mandate the storage of cryptographic keys owned by the HSM Solution Consumers—systems that dynamically allocate user keys on a per-operation basis and do not store user cryptographic keys do not need to meet this requirement. However, such systems would still be required to meet other requirements of this standard, including but not necessarily limited to those for provisioning and secure channels.

.

TK5.1 The tester shall confirm how the HSM Solution stores the cryptographic keys of HSM Solution Consumers, and how this ensures isolation between the cryptographic keys of different HSM Solution Consumers present on the same HSM processing element.

TK5.2 For solutions where the cryptographic keys of HSM Solution Consumers may be stored external to the tamper-responsive boundary of the HSM processing element, the tester shall confirm that the tamper key that is implemented for the storage of keys for each separate HSM Solution Consumer is either internally generated by the HSM processing element itself, or is loaded and managed by the HSM Solution Consumer directly.

TK5.3 The tester shall confirm that the tamper key(s) created by the HSM processing element(s) are generated by a secure key-generation process, using either the random number generator validated under the PCI HSM requirements, or a key-derivation process approved by NIST. Tamper keys generated using a derivation process must be derived from other cryptographic keys that are also erased upon a tamper event (that is, another tamper key).

TK5.4 The tester shall confirm that any keys used to protect cryptographic keys stored external to the tamper-responsive boundary of the HSM Processing Element, do so in compliant (i.e., ASC X9 TR 31/ANSI X9.143/ISO 20038), key-wrapped blocks at all times.

TK5.5 The tester shall confirm that any tamper key not directly managed by an HSM Solution Consumer cannot be exported from the HSM processing element—this includes for purposes of backup, load balancing, or fault tolerance. HSM processing element keys that may be used to protect the keys of more than one HSM Solution Consumer must not be able to be exported from the HSM processing element in any form.

TK5.6 The tester shall review the source code of the HSM processing element to confirm that no export or other functions exist that would disclose the values of HSM keys used to manage the keys of more than one HSM Solution Consumer.

TK5.7 For solutions that implement tamper keys generated by the HSM processing element, the tester shall establish two HSM Solution Consumer instances on a single HSM processing element and load the same cryptographic key into each instance. The tester shall confirm that the wrapped keys stored external to the HSM processing elements are different in each case.

TK5.8 For solutions that implement the ability for HSM Solution Consumers to load their own tamper keys, the tester shall detail what cryptographic methods are implemented for the establishment or loading of any master keys used by the HSM Solution Consumer(s),and confirm that these controls prevent man-in-the-middle, pre-play, and replay attacks. These controls must include preventing such attacks by the HSM Solution Provider itself. Methods must use acceptable algorithms and key lengths.

DTR K6 Decommissioning

The HSM Solution must support the ability to disable or suspend access to cryptographic keys owned by an HSM Solution Consumer, for any HSM processing element.

Guidance

This requirement is designed to ensure that any specific HSM processing element can be revoked by an individual or group of HSM Solution Consumer(s)—for example, in the case it is considered compromised or suspect, or where a specific HSM Solution Consumer has needs that are not met by some subset of the HSM processing elements. The requirement does not enforce who is authorized to disable the HSM processing element—this may be an action of the HSM Solution Consumer or HSM Solution Provider, but actions by an HSM Solution Consumer to disable a specific HSM processing element must not disable that HSM processing element for any other HSM Solution Consumer.

For example, an HSM Solution Consumer may want to prevent a set of HSM processing elements from accessing their cryptographic keys due to the configuration, location, or status of those HSM processing elements—but those same HSM processing elements may be acceptable for use by another HSM Solution Consumer. Alternatively, the HSM Solution Provider may wish to block access by all HSM Solution Consumers to a subset of HSM processing elements as they are updated, retired, or examined in some way.

TK6.1 The tester shall detail the methods provided to disable an HSM processing element within the solution.

TK6.2 The tester shall confirm that these methods are able to be deployed against any HSM processing element as required.

TK6.3 The tester shall confirm that any HSM Solution Consumer command to disable an HSM processing element does not disable that HSM processing element for any other HSM Solution Consumers.

TK6.4 The tester shall confirm that these methods implement cryptographic controls to authenticate the disablement request, and detail the cryptography implemented. The tester shall confirm that these controls provide protections to man-in-the-middle, pre-play, and replay attacks.

DTR K7 External Logs

It must be possible for an HSM Solution Consumer to maintain an external log of all HSM operations, indicating which HSM virtualization system(s) and HSM processing element(s) were involved in all operations performed.

Guidance

HSM Solution Consumers should have access to this information in order to have some visibility into actions impacting their cryptographic materials.

This log may be implemented by either returning these details in each command or maintaining a log that can be queried by the HSM Solution Consumer at any time. If a queriable log is maintained, controls must be implemented so the HSM Solution Consumer is able to obtain only that data related to the associated instance of HSM processing element(s).

The intent of this requirement is not to log the full details of every HSM Solution Consumer command and HSM response, but to allow for an HSM Solution Consumer to know which commands and cryptographic keys were operated by which HSM processing elements. An acceptable method of meeting this requirement would be to provide a log allowing sufficient granularity for the HSM Solution Consumer to know which elements may have processed which commands or had access to which cryptographic keys.

It is acceptable for the log to be provided with an opt-in process, requiring that HSM Solution Consumers configure this for use prior to performing operations with the HSM Solution. If such an opt-in method is implemented, it must be possible for the HSM Solution Consumer to opt-in at any time, not only upon initial configuration of the HSM Solution.

TK7.1 The tester shall confirm that the HSM Solution is able to provide data on each HSM virtualization system and each HSM processing element involved in any command issued by an HSM Solution Consumer.

TK7.2 The tester shall detail how this data is provided to the user and confirm that cryptographic controls are implemented to ensure data authenticity.

TK7.3 Where this data is provided in a queriable log, the tester shall confirm what methods are implemented to ensure that only the HSM Solution Consumer is able to retrieve this log.

TK7.4 The tester shall establish two HSM Solution Consumer instances on a single HSM processing element and perform some operations. The tester shall confirm that the logs for the operations contain the data required and are not visible to the second HSM Solution Consumer on the system.

TK7.5 The tester shall detail what cryptographic methods are implemented and confirm that these controls prevent man-in-the-middle, pre-play, and replay attacks. Methods must use acceptable algorithms and key lengths.

DTR K8 Key Operations Performed by HSM Solution Consumers

Where it is possible that more than one HSM processing element may be operating on the cryptographic keys of the same HSM Solution Consumer at any one time, these HSM processing elements must be maintained in an environment meeting at least the security requirements of a controlled environment.

Note: This requirement is designed to reduce the risk of key compromise as the cryptographic keys of HSM Solution Consumers are spread across multiple HSM processing elements for the purpose of elastic processing.

Guidance:

Although this requirement does not prohibit the use of more than one HSM processing element from operating on the cryptographic keys of the same HSM Solution Consumer, it should not be possible for two or more HSM processing elements without additional security to access the same key space at any one time. Additionally, each HSM processing element including multiple instances running on the same physical device must not have any knowledge of key values for separate instances.

As it is not possible to validate a limit beyond "only one" within this standard, once there is the ability for two or more processing elements to access the cryptographic keys of the same HSM Solution Consumer, there can be no clear limit on the number of elements that do have access. Therefore, to mitigate the risk this poses to leakage of cryptographic keys owned by HSM Solution Consumer, environments meeting at least the security of a controlled environment must be used.

Solutions that ensure that cryptographic keys are only accessible to a single HSM processing element at any one time are exempt from this requirement and may be deployed in uncontrolled environments.

TK8.1 The tester shall confirm whether the HSM Solution allows for the HSM Solution Consumer to provision more than one HSM processing element to store the cryptographic keys of that same HSM Solution Consumer at any time. If this is not the case, no further testing for this requirement is necessary.

TK8.2 The tester shall confirm that HSM processing elements used in the solution are maintained in an environment meeting at least the security of a controlled environment and that this assertion is part of the security policy posted to the PCI website.

TK8.3 The tester shall detail the controls implemented in the environment, and how they were validated.

DTR K9 Cloud Security Policy

There must be a publicly available security policy that outlines how the HSM Solution is implemented, what cryptographic services are provided, how the HSM Solution Consumer and components of the HSM Solution are authenticated, and how to securely disable or deprecate the storage of cryptographic keys of specific HSM Solution Consumers from any or all components of that solution.

Guidance:

The information may be contained in the security policy that is created in conjunction with Requirement C1 or in a separate, publicly available security policy that is reviewed by the PCI PTS recognized evaluation laboratory. If in a separate document, it must be referenced in the security policy created as part of C1.

TK9.1 The tester shall confirm that there is a publicly available security policy for the HSM processing solution.

TK9.2 The tester shall confirm that this policy includes all items required for a PCI HSM compliance solution, including what cryptographic services are provided/supported by the solution.

TK9.3 The tester shall confirm that the policy includes details on how HSM Solution Consumers are authenticated to the solution.

TK9.4 The tester shall confirm that the policy includes details on how the cryptographic keys of HSM Solution Consumers are provisioned to the HSM Solution.

TK9.5 The tester shall confirm that the policy includes details on how compliance to these requirements may be maintained when using the HSM Solution.

TK9.6 The tester shall confirm the policy includes details on how cryptographic keys of specific HSM Solution Consumers can be decommissioned from the HSM Solution.

DTR K10 Cloud Vulnerability Management

The HSM Solution must implement a public vulnerability management and disclosure policy. This must provide:

- ***Methods for the secure disclosure to the impacted user organization of any vulnerabilities as they are found,***
- ***Ranking and addressing vulnerabilities in a timely manner, and***
- ***Methods for any HSM Solution Consumers of the HSM Solution to be informed of vulnerabilities.***

Guidance:

This is intended to describe general practices and not individual actual occurrences.

TK10.1 The tester shall confirm that if a vulnerability-reporting program exists for the system, there is evidence of accepting and remediating security vulnerabilities found through this program. The vendor must be able to demonstrate that any such vulnerabilities are processed through its risk and update process and patched accordingly.

TK10.2 The tester shall confirm that the vendor has a documented process that is demonstrably in use for the discovery and remediation of security vulnerabilities in its system.

TK10.3 The tester shall confirm that a penetration test has been performed on the system. The scope of the penetration test(s) must include all components of the overall HSM Solution. The tester shall further confirm that a documented policy exists to require the repeat of the penetration test at least every year. Where sampling is performed, the tester shall validate the sampling is appropriate.

TK10.4 The tester shall confirm that the vendor has an explicit procedure in place for the acceptance and processing of new vulnerabilities through external communications. This requirement does not mandate the implementation of a "bug bounty" program but does require that all reported vulnerabilities are formally registered and managed according to a documented process.

TK10.5 The tester shall confirm that there is a public-facing procedure for the reporting of vulnerabilities in the solution. This procedure must implement methods to ensure the confidentiality of the vulnerability as it is reported—e.g., a process that requires the reporting of a vulnerability to a shared "info@[company]" e-mail address, without additional encryption, would be non-compliant to this requirement. Use of a specific web portal (not e-mail system) secured with TLS (using acceptable cipher suites) and/or e-mails secured with strong cryptography are examples of acceptable methods to secure the confidentiality of vulnerability reporting.

TK10.6 The tester shall submit a vulnerability report using the public process and ensure that it is correctly received and processed. This submission may be of a “dummy” vulnerability and is intended to validate the process only.

TK10.7 The tester shall obtain a list of vulnerabilities reported through the vulnerability-management program and penetration testing and validate that these have been addressed as per the vendor’s documented vulnerability-management program.

DTR Module 5: Life Cycle Security Requirements

L – During Manufacturing Derived Test Requirements

Compliance can be proven by evidence that the procedures comply by design via:

- Evidence of existence—i.e., that these procedures are implemented*. Example: The lab has received procedures from the company that implements the requirement.*
- When these procedures are in use, samples of the process in operation. For this purpose—in a timeframe of 3, 6, or 12 months—samples can be randomly collected and validated. Example: For L3, the lab can randomly select a few occasions and ask for the related log belonging to the dual-control or standardized cryptographic authentication procedure used.

DTR L1 Change Control

Change-control procedures are in place so that any intended change to the physical or functional capabilities of the device causes a re-certification of the device under the impacted security requirements of this document. Immediate re-certification is not required for changes that purely rectify errors and faults in software in order to make it function as intended and do not otherwise remove, modify, or add functionality.

Guidance

The organization must utilize a documented and approved, secure change management system, such that all development is traceable and access restricted. All changes should be reviewed and approved prior to release. The change control procedures shall include the unique identification of all configuration items and their developer, including the device, its parts—and its firmware.

***Note:** Hardware and firmware version number identifiers may consist of a combination of fixed and variable alphanumeric characters, whereby a lowercase "x" is used by PCI to designate all variable fields. The "x" represents fields that the vendor can change at any time to denote a different device configuration. Options that cannot be a variable character include those that directly pertain to meeting security requirements.*

Pure bug fixes, including patches for known security flaws, do not require a delta submission or a change to the non-wildcarded firmware version number fields.

TL1.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail change control processes and procedures for physical and functional changes to the device, describing how the items are uniquely identified such that it is possible to track the different versions of the items, and including criteria for when changes are submitted for PCI evaluation and details of the vulnerability management process.

TL1.2 The tester shall review any vendor documentation describing all changes to the software to ensure that changes that rectify errors or faults did not remove, modify, or add functionality that impacts security unless they are submitted immediately upon completion for evaluation. This is to validate the process of deferring submissions for evaluation unless all changes are submitted for evaluation immediately upon completion.

TL1.3 The tester shall examine a sample of documentation of physical and functional changes to devices, such as change control logs, to validate that procedures— including the submission and sign-off processes—are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL1.4 The tester shall interview those responsible for the change control processes—including engineers and programming staff, supervisory staff, technical management, etc.—to verify that the documented and approved procedures are known and understood by all affected parties.

TL1.5 The tester shall validate that either:

- All changes to PCI-approved devices have been submitted for re-evaluation; or
- Only changes that purely rectify errors and faults in software in order to make it function as intended and do not otherwise remove, modify, or add functionality are deferred from submission and not submitted immediately, but are instead submitted on an aggregate basis.

DTR L2 Firmware Certification

The firmware and any changes thereafter have been inspected and reviewed using a documented and auditable process and verified by the vendor as being free from hidden and unauthorized or undocumented functions.

Guidance

Firmware is considered to be any code within the device that provides security protections needed to comply with PCI requirements. This is true regardless of how labelled—e.g., OS, EPROM code, DLL's, parameter files, applications, kernel code. This includes any code that is not in the Vendor Domain as defined in Appendix F, "Domain-Based Asset Flow Analysis." Other code that exists within the device that does not provide security, and cannot impact security, is not considered firmware under PCI requirements.

"Certify firmware" refers to self-certification. This requirement, in essence, requires the vendor to have implemented and to use internal quality control and change-control systems. With these systems in place, the vendor is in control of the code and can attest to the fact that the code is free of hidden or unauthorized functions.

The vendor shall indicate the compiler settings used in order to maximize the mitigation of known vulnerabilities.

The vendor shall implement measures to help prevent common exploits of "buffer overflow" and similar vulnerabilities. Strategies by the developer to address these vulnerabilities may include:

- *Avoiding them by design (extreme example: using a programming language, which prevents buffer overflow by definition);*
- *Finding them by adequate testing (for example, static code analysis or comprehensive fuzz testing); and/or*
- *Mitigating them by techniques that include but are not limited to: address space layout randomization (ASLR), Data Execution Prevention (DEP), Harvard Architecture, and stack canaries.*

The vendor shall document where labs may place reliance upon this in connection with B2 and other relevant requirements.

TL2.1 The tester shall examine details provided by the vendor to confirm that the documented process explicitly addresses how testing/auditing has been carried out to check for unauthorized and undocumented functions.

TL2.2 The tester shall detail any compiler settings used by the vendor in order to maximize the mitigation of known vulnerabilities. If no specific compiler settings are used, the tester shall detail how known vulnerabilities are mitigated.

TL2.3 The tester shall detail any mitigation techniques used by the vendor to help prevent common exploits. The tester must state and justify any reliance placed on these technique(s) in minimizing testing. If no specific techniques are used, the tester shall detail how the common exploits are prevented.

TL2.4 The tester shall confirm that—and summarize how—the device vendor has a documented software-development process that details how firmware must be written, reviewed, and tested to ensure the firmware is free from security vulnerabilities.

The tester shall confirm that—and summarize how—the device vendor has a documented software-development process that details how firmware developers need to be trained to recognize and avoid common security problems.

The tester shall reference the names and versions of all relevant documents that define the software-development process.

TL2.5 The tester shall confirm that—and summarize how—the documented software-development process provides specific guidance for programmers, reviewers, and testers, and does not rely on non-specific statements—for example, the guidance "does not create buffer overflows" would be insufficient as it does not provide information to the programmer on how to prevent these problems.

TL2.6 The tester shall confirm that—and summarize how—the certification process includes checking of all code that executes in all HSM processing elements as listed in DTR A3.

TL2.7 The tester shall confirm that—and summarize how—the process described above includes checking sources of vulnerability disclosure (such as the national vulnerability database) for public vulnerabilities that may apply to the device firmware.

TL2.8 The tester shall confirm and describe, via documentation review, that the certification process requires that the code review and security testing is performed by a person who was not involved in the authorship of the device code.

TL2.9 The tester shall confirm and describe, via documentation review, that the certification process requires that the code review and security testing be performed after every security-relevant change, and before the firmware is released to production.

TL2.10 The tester shall confirm and describe, via documentation review, that the certification process is designed to result in an auditable process that shows the code review and security testing have been performed and require sign-off by the person(s) performing the code review and security testing. The tester shall confirm that the process will show any problems noted during the code review and security testing.

TL2.11 The tester shall obtain, review, and summarize the firmware-verification audit results provided by the vendor for the version of firmware submitted for evaluation. The tester shall confirm that this shows that no problems were found during review of this firmware (or that any problems found have been addressed).

TL2.12 The tester shall review previous firmware-verification audit reports provided by the vendor and summarize these reports and their findings. The tester shall ensure that the reports sampled include security problems found during the review and confirm that these problems have been addressed. If all audit reports reviewed indicate that no security problems have been found, the tester shall justify why it is possible that the firmware image is, and has always been, completely free of security defects.

TL2.13 The tester shall detail from the vendor-provided documentation the process used to manage firmware from development, through certification, to release to production, and installation into the device. The tester shall justify how this process ensures that it is infeasible for the code image to be altered after it has been reviewed and signed off during the firmware-verification process.

TL2.14 The tester shall outline the following information: If the firmware is based on a general-purpose operating system (like Linux), the steps described in TL2.13 hold for this operating system. The documentation provided by the developer shall show that state-of-the-art rules for "hardening" the operating system have been applied. For example, the developer should provide a table showing a list of all known issues—like patch levels; not including unused packages, etc.; not being able to log into the operating system without cryptographic authentication in operational mode of the device; not being able to use debug functions like "netdump" during operational use—with remarks indicating how each issue has been addressed for the firmware under evaluation. Similar steps need to be performed for other firmware parts that are taken from external sources. The evidence provided by the developer should also include acceptance procedures, showing how the developer ensures that external software can be considered secure.

DTR L3 Certified Firmware Control

The certified firmware is protected and stored in such a manner as to preclude unauthorized modification during its entire manufacturing lifecycle—e.g., by using dual-control or standardized cryptographic authentication procedures.

Guidance

All certified firmware must be signed prior to distribution and signed using dual control. It should be managed such that no single person is able to sign files. If two secrets (passwords/authentication codes) are required for operation of the signing tool, no single person should know both secrets. All certified firmware must be reviewed against the organization's coding standards prior to signature.

TL3.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail firmware control and validation processes and procedures for storage during the manufacturing process.

TL3.2 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail dual-control and cryptographic authentication processes and procedures during manufacturing and describe how this shows compliance to this requirement.

TL3.3 The tester shall examine a sample of documentation for firmware authentication (i.e., signing) and storage during manufacturing—including change control logs and firmware validation procedures—to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL3.4 The tester shall interview those responsible for the firmware control processes—including engineers and programming staff, peer reviewers, supervisory staff, technical management, etc.—to verify that the documented and approved procedures are known and understood by all affected parties.

TL3.5 If firmware signing is done on site, the tester shall observe the signing process, the signing tools, and ensure they are under dual control and that the signing procedures are followed.

TL3.6 The tester shall observe the firmware storage and validation process to ensure that only signed and valid firmware is used during manufacturing.

DTR L4 Device Hardware Component Control

The device is assembled in a manner that the hardware components used in the manufacturing process are those hardware components that were certified by the Section A Security Requirements evaluation, and that unauthorized substitutions have not been made.

TL4.1 The tester shall examine and cite any supporting documentation submitted by the vendor and describe how this shows compliance to this requirement. The documentation should detail the components used and the hardware component control processes and procedures for storage and verification during the manufacturing process including hardware components identification verification, with the procedures checked in TL1.

TL4.2 The tester shall examine and cite any supporting documentation submitted by the vendor and describe how this shows compliance to this requirement. This shall include:

- Describing how it is ensured that the hardware components used in assembly are the same as those used in the devices submitted for certification.
- Providing details on how components are stored before being assembled into devices.
- Describing how the manufacturer verifies parts before using them in manufacture.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL4.3 The tester shall interview those responsible for hardware component control processes—including engineers and line staff, supervisory staff, technical management, etc.——to verify that the documented and approved procedures are known and understood by all affected parties.

TL4.4 The tester shall examine the device parts listing and sample hardware components during manufacturing, to ensure the correct components are used.

TL4.5 The tester shall observe hardware component control to ensure that authorized components are verified prior to manufacturing and that unauthorized substitutions are not made.

DTR L5 Production Firmware Control

Production software—e.g., firmware—that is loaded to devices at the time of manufacture is transported, stored, and used under the principle of dual control, preventing unauthorized modifications and/or substitutions.

Guidance

All software (e.g., firmware) loaded into the device must be signed prior to use; and all software (e.g., firmware) needs to be controlled and protected during manufacturing. All software (e.g., firmware) must be validated before each use to prevent unauthorized modifications and/or substitutions.

TL5.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail firmware control processes and procedures for loading firmware during the manufacturing process.

TL5.2 The tester shall examine, cite, and reference any supporting documentation submitted by the vendor. The documentation should detail firmware control processes and procedures for transporting and storing firmware during the manufacturing process and that the principle of dual control is followed to prevent unauthorized modifications and/or substitutions.

TL5.3 The tester shall examine a sample of documentation for firmware control and storage during manufacturing—including change control logs and firmware validation procedures—to ensure the principle of dual control is followed to prevent unauthorized modifications and/or substitutions.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL5.4 The tester shall interview those responsible for the software (e.g., firmware) control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the documented procedures are known and understood by all affected parties.

TL5.5 The tester shall observe the firmware storage and validation process to ensure that procedures are followed during manufacturing.

DTR L6 Post-Production Storage

Subsequent to production but prior to shipment from the manufacturer's or reseller's facility, the device and any of its components are stored in a protected, access-controlled area or sealed within tamper-evident packaging to prevent undetected unauthorized access to the device or its components and to prevent unauthorized modifications to the physical or functional characteristics of the device.

Guidance

Completed devices and any of their components must be controlled and stored in tamper-evident packaging and/or stored in an accessed controlled area until shipped. The device or components must be checked prior to shipping to ensure the device has not been modified or tampered.

This requirement applies to all devices, or parts thereof, that are vulnerable to tampering after manufacturing but before delivery to the customer. Manufactured devices that have tamper detection and response enabled are exempt from this requirement.

TL6.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail documented and approved post-production control processes and procedures for storage and validation of devices or their components subsequent to production but prior to shipping.

TL6.2 The tester shall examine, cite, and reference any supporting documentation submitted by the vendor. The documentation should detail tamper-evident packaging or the access-controlled area where devices and components are stored prior to shipping.

TL6.3 The tester shall examine a sample of documentation for post-production storage—including inventory logs, shipping documentation, and device-validation procedures—to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL6.4 The tester shall interview those responsible for the post-production control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TL6.5 The tester shall observe the device and component storage and validation process to ensure that procedures are followed subsequent to production.

TL6.6 The tester shall examine the vendor's tamper-evident packing or access-controlled area to ensure unauthorized access to devices or their components is not possible.

DTR L7 Secret Information

The device will be authenticated at the facility of initial deployment by means of secret information placed in the device during manufacturing, this secret information is unique to each device, unknown and unpredictable to any person, and installed in the device. Secret information is installed under dual control to ensure that it is not disclosed during installation, or the device may use an authenticated public-key method.

Authentication by secret information is mandatory in HSM v4.

Guidance

One example of such information is the private key of an asymmetric key pair with the public key of the device signed by a private key known only to the manufacturer.

Dual control is not required if the device generates the secret information and never outputs it, for example if it generates a public/private key pair and never outputs the private key.

TL7.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail that if the device will be authenticated at the key-loading facility or the facility of initial deployment by means of secret information placed in the device during manufacturing, this secret information is unique to each device, unknown, and unpredictable to any person.

TL7.2 The tester shall examine, cite, and reference any supporting documentation submitted by the vendor. The documentation should detail that if the device will be authenticated at the facility of initial deployment by means of secret information placed in the device during manufacturing. The secret information is installed in the device under dual control to ensure that it is not disclosed during installation, or the device may use an authenticated public-key method.

TL7.3 The tester shall examine a sample of documentation for secret information—including user documentation and device-validation procedures—to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL7.4 The tester shall interview those responsible for the control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TL7.5 The tester shall observe the device secret installation and validation process to ensure that documented and approved procedures are followed subsequent to production. The tester shall verify that if secret information is placed in the device during manufacturing, this secret information is unique to each device, unknown and unpredictable to any person, and installed in the device under dual control to ensure that it is not disclosed during installation.

DTR L8 Component Control during Design and Development

Security measures are taken during the development and maintenance of the device's security-related components. The manufacturer must maintain development-security documentation describing all the physical, procedural, personnel, and other security measures that are necessary to protect the integrity of the design and implementation of the device's security-related components in their development environment. The development-security documentation shall provide evidence that these security measures are followed during the development and maintenance of the device's security-related components. The evidence shall justify that the security measures provide the necessary level of protection to maintain the integrity of the device's security-related components.

TL8.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail the design and development processes and procedures to ensure security measures are followed during the development and maintenance of the device's security-related components. Furthermore, the approved documentation must justify that the security measures provide the necessary level of protection to maintain the integrity of the device's security-related components.

TL8.2 The tester shall examine a sample of approved documentation for component control during design and development—including user documentation and component validation procedures—to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL8.3 The tester shall interview those responsible for component control processes—including engineers and line staff, supervisory staff, technical management, etc.—to verify that the approved documented procedures are known and understood by all affected parties.

TL8.4 The tester shall observe the approved component control procedures during design and development to ensure security measures are followed during the development and maintenance of the device's security-related components.

TL8.5 The tester shall verify that the manufacturer maintains approved development-security documentation describing all the physical, procedural, personnel, and other security measures that are necessary to protect the integrity of the design and implementation of the device's security-related components in their development environment and that this provides evidence that these security measures are followed during the development and maintenance of the device's security-related components.

DTR L9 Repair and Inspection Control

Controls exist over the repair process and the subsequent inspection/testing process to ensure that the device has not been subject to unauthorized modification.

Guidance

Completed devices and any of their components must be controlled and stored in tamper-evident packaging and/or stored in an accessed controlled area until shipped. The device or components should be checked prior to shipping to ensure the device has not been modified or tampered.

Devices undergoing repair must be in a dual-access-controlled area or decommissioned such that all sensitive information and CSPs are cleared from the device prior to being returned to the device manufacturer.

TL9.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail repair—including the resetting of tamper mechanisms, inspection, and post-inspection control processes—and procedures for storage and validation of devices or their components subsequent to repair and inspection but prior to shipping.

TL9.2 The tester shall examine, cite, and reference any supporting documentation submitted by the vendor. The approved documentation shall detail the inspection process and tamper-evident packaging or the access-controlled area where devices and components are stored prior to shipping.

TL9.3 The tester shall examine a sample of approved documentation for repairs, including the resetting of tamper mechanisms; inspection and post-inspection storage, including inventory logs, repair logs, shipping documentation; and device-validation procedures to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TL9.4 The tester shall interview those responsible for repair, inspection, and post-inspection control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TL9.5 The tester shall observe the device repair, inspection, and post-inspection storage process to ensure that procedures are followed subsequent to production.

TL9.6 The tester shall examine the vendor's tamper-evident packing or accessed-controlled area to ensure unauthorized access to devices or their components is not possible.

M – Between Manufacturer and Facility of Initial Deployment Derived Test Requirements

Compliance can be proven by evidence that the procedures comply by design via:

- Evidence of existence—i.e., that these procedures are implemented. *For example, the lab has received procedures from the company that implements the requirement.*
- When these procedures are in use, samples of the process in operation. For this purpose—in a timeframe of 3, 6, or 12 months—samples can be randomly collected and validated. *Example: For M3, the lab can randomly select a few occasions and ask for the related log belonging to the received shipments proving that the packaging is validated on tamper evidence.*

DTR M1 Shipping Tamper-Protection Documentation

The device should be protected from unauthorized modification with tamper-detection security features, and customers shall be provided with documentation (both shipped with the product and available securely online) that provides instruction on validating the authenticity and integrity of the device.

Where this is not possible, the device is shipped from the manufacturer's facility to the facility of initial deployment and stored en route under auditable controls that can account for the location of every device at every point in time.

Where multiple parties are involved in organizing the shipping, it is the responsibility of each party to ensure that the shipping and storage they are managing is compliant with this requirement.

Guidance

The product shall be protected for at all points during the shipping process. Tamper-detection security features, instructions for receiving and inspection, and documentation covering suspected tampering must be provided and used.

Prior to initial financial-key loading, the device should be protected against modification. Such protection shall be a combination of the characteristics of the device (i.e., tamper evidence, resistance, and responsiveness) and device management procedures. If the device has any manufacturer keys loaded, compromise shall be both prevented and detected.

TM1.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should provide customers instruction on validation of the authenticity and integrity of the device, detailing the shipping process, tamper protection, instructions for receiving and inspection, as well as procedures for suspected tamper evidence. Alternatively, the approved documentation must detail how the device is shipped from the manufacturer's facility to the facility of initial deployment and stored en route under auditable controls that can account for the location of every device at every point in time.

TM1.2 The tester shall examine approved documentation detailing that where multiple parties are involved in organizing the shipping, it is the responsibility of each party to ensure that the shipping and storage they are managing is compliant with this requirement.

TM1.3 The tester shall examine a sample of documentation for tamper protection; inspection and transfer documentation, including inventory logs, shipping documentation; and device-validation procedures to ensure procedures are followed and describe how this shows compliance to this requirement.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM1.4 The tester shall interview those responsible for the shipping control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved documented procedures are known and understood by all affected parties.

TM1.5 The tester shall observe the shipping process to ensure that customers are provided documentation (both shipped with the product and available securely online) that provides instruction on validating the authenticity and integrity of the device; or alternatively, the device is shipped from the manufacturer's facility to the facility of initial deployment and stored en route under auditable controls.

DTR M2 Device-Accountability Transfer Procedures

Procedures are in place to transfer accountability for the device from the manufacturer to the facility of initial deployment. Where the device is shipped via intermediaries such as resellers, accountability will be with the intermediary from the time at which they receive the device until the time it is received by the next intermediary or the point of initial deployment. In the absence of defined agreements stipulating otherwise, the device vendor remains responsible.

Guidance

The product shall be accounted for at all points during the shipping process until transfer of responsibility. Documentation shall be maintained for each device.

TM2.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail the instructions for receiving and inspection, as well as procedures for the transfer of responsibility. Furthermore, the documentation should detail that where the device is shipped via intermediaries such as resellers, accountability will be with the intermediary from the time at which they receive the device until the time it is received by the next intermediary or the point of initial deployment.

TM2.2 The tester shall examine a sample of transfer documentation—including inventory logs and shipping documentation—to ensure procedures are followed

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM2.3 The tester shall interview those responsible for the shipping control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved documented procedures are known and understood by all affected parties.

TM2.4 The tester shall observe the shipping and transfer procedures to ensure that procedures are followed to transfer accountability for the device from the manufacturer to the facility of initial deployment.

TM2.5 The tester shall verify that where the device is shipped via intermediaries such as resellers, accountability is with the intermediary from the time at which they receive the device until the time it is received by the next intermediary or the point of initial deployment.

DTR M3 Shipping Security Procedures

While in transit from the manufacturer's facility to the initial key-loading facility, the device is:

- ***Shipped and stored in tamper-evident packaging; and/or***
- ***Shipped and stored containing a secret that:***
  - ***Is immediately and automatically erased if any physical or functional alteration to the device is attempted, and***
  - ***Can be verified by the initial key-loading facility but cannot feasibly be determined by unauthorized personnel.***

TM3.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail the shipping process, tamper protection, instructions for receiving and inspection, as well as procedures for suspected tamper evidence; and describe how this shows compliance to this requirement.

TM3.2 The tester shall examine a sample of documentation for tamper protection, inspection and transfer documentation—including inventory logs, shipping documentation, and device-validation procedures—to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM3.3 The tester shall interview those responsible for the shipping control processes—including engineers, software developers, line staff, supervisory staff, technical management, etc.—to verify that the approved documented procedures are known and understood by all affected parties.

TM3.4 The tester shall examine a sample of documentation for tamper protection using secret keys to ensure approved procedures for validation at the key-loading facility are followed.

TM3.5 The tester shall observe the shipping process to ensure that devices are shipped and stored containing a secret that is immediately and automatically erased if any physical or functional alteration to the device is attempted, that can be verified by the initial key-loading facility but that cannot feasibly be determined by unauthorized personnel.

DTR M4 Development-Security Documentation

The device's development-security documentation must provide means to the facility of initial deployment to assure the authenticity of the device's security-relevant components.

Guidance

The means to verify the authenticity of the device may include tools such as an SCD.

TM4.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail the device's development-security documentation to ensure the authenticity of the device's security-relevant components and describe how this shows compliance to this requirement.

TM4.2 The tester shall examine a sample of documentation for device-development security, including user documentation and component validation procedures to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM4.3 The tester shall interview those responsible for the initial key-loading facility—including engineers and line staff, supervisory staff, technical management, etc.—to verify that the approved documented procedures are known and understood by all affected parties.

TM4.4 The tester shall observe the secure development component control procedures to ensure security measures are followed during the initial key loading.

DTR M5 Authenticity of Device Security-Related Components

If the manufacturer is in charge of initial key loading, the manufacturer must verify the authenticity of the device's security-related components.

TM5.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail procedures for the manufacturer to ensure the authenticity of the device's security-relevant components.

TM5.2 The tester shall examine a sample of documentation, including user documentation, and component validation procedures to ensure procedures are followed.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM5.3 The tester shall interview those responsible for the initial key-loading facility—including engineers and line staff, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TM5.4 The tester shall observe the initial key-loading procedures to ensure the authenticity of the device's security-related components is verified.

DTR M6 Authenticity of Device Security-Related Components for Key-Loading Facility

If the manufacturer is not in charge of initial key loading, the manufacturer must provide the means to the facility of initial deployment to assure the verification of the authenticity of the device's security-related components.

TM6.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail procedures provided by the manufacturer to the initial key-loading facility to assure the authenticity of the device's security-relevant components for the initial key-loading facility.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM6.2 The tester shall interview those responsible for the initial key-loading facility—including engineers and line staff, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TM6.3 The tester shall examine a sample of documentation, including user documentation and component-validation procedures, to ensure procedures are followed at the facility of initial deployment.

DTR M7 Unique Visible Identifier

Each device shall have a unique visible identifier affixed to it—i.e., model name and hardware version—and shall be retrievable by a query using secure, cryptographically protected methods.

TM7.1 The tester shall examine and cite any supporting documentation submitted by the vendor. The documentation should detail how the change control procedures required in L1 are used in compliance with this requirement.

TM7.2 The tester shall verify that the model name and hardware version are retrievable from the device by a query using secure, cryptographically protected methods.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM7.3 The tester shall observe the manufacturing process to ensure the visible identifier is affixed to each device.

TM7.4 The tester shall sample the devices to ensure that each visible identifier is unique and is consistent with the identifiers checked with the change control procedure in as required in L1.

DTR M8 Operational Management Manual

The vendor must maintain a manual that provides instructions for the operational management of the device. This includes instructions for recording the entire lifecycle of the device's security-related components and of the manner in which those components are integrated into a single device—e.g.:

- ***Data on production and personalization***
- ***Physical/chronological whereabouts***
- ***Repair and maintenance***
- ***Removal from operation***
- ***Loss or theft***

Guidance

The operational management manual provides instructions and information concerning the identification, use, repair, updates, and configuration of hardware and firmware components of the device. This manual is in addition to user and application operation and configuration manuals.

TM8.1 The tester shall examine and cite any supporting documentation submitted by the vendor and describe how this shows compliance to this requirement. The documentation should include instructions for recording the entire life cycle of the device's security-related components and the manner in which those components are integrated into a single device. The operations management manual must be current and up to date.

Where required to validate via site inspection, the tester shall complete the following additional steps:

TM8.2 The tester shall interview those responsible for maintaining the operation management manual—including engineers and software developers, supervisory staff, technical management, etc.—to verify that the approved and documented procedures are known and understood by all affected parties.

TM8.3 The tester shall examine a sample of the operations management manual to ensure procedures are followed and recorded.

Appendix A: Physical Attack Potential Formula (Adopted from JIL)

Calculating Attack Potentials

This section presents factors that determine attack potentials and provides guidelines to help remove some of the subjectivity from this aspect of the evaluation process. Attack potential calculations in reports must accurately reflect the actual evaluated device's properties, coherently with this guidance. The approaches here should be adopted unless the tester determines that it would be inappropriate, in which case a strong rationale is required to justify the validity of the alternative approach. The examples shown here are generic descriptions and should not be assumed to apply to any specific device. Strong and detailed justification must be made for attributing high levels in ratings.

Identification and Exploitation

For an attacker wanting to exploit a vulnerability, the vulnerability must first be identified. This may appear to be a trivial separation, but it is an important one. To illustrate this, first consider a vulnerability that is uncovered following months of analysis by an expert, and a simple attack method published on the Internet. Compare this to a vulnerability that is well known but requires enormous expenditure of time and resources to exploit. Of course, factors such as time need to be treated differently in these cases.

In this document, "exploitation" and "initial exploitation" are alternatively used to designate "initial exploitation."

Factors to be Considered

The following factors should be considered for the analysis of the attack potentials required to exploit a vulnerability:

1. **Identification**
   a) Attack time for the various levels of expertise;
   b) Potential to acquire the required knowledge of the HSM's design and operation;
   c) Potential for the access to the HSM;
   d) Equipment required such as instruments, components, IT hardware and software required for the analysis;
   e) HSM-specific spare components.
2. **(Initial) Exploitation**
   a) Attack time for the various levels of expertise;
   b) Potential to acquire the required knowledge of the HSM's design and operation;
   c) Potential for the access to the HSM;
   d) Equipment required like instruments, components, IT hardware, software required for the analysis;
   e) HSM-specific spare components.

In many cases these factors don't depend on each other but might be substituted for each other in varying degrees. For example, expertise or hardware/software can be a substitute for time. A discussion of these factors follows. In most situations, the first identification phase shall include a complete simulation of the second phase—i.e., the (initial) exploitation phase. Therefore, it is not acceptable to allocate calculation factors (such as time, experience, equipment) of the simulation to an exploitation phase when these can be attributed to the identification phase.

The **attack time** is given in the time in hours taken by an attacker to identify or exploit an attack. If the attack consists of several steps, the attack time can be determined and added to achieve a total attack time for each of these steps. Actual labor time must be used instead of time expired as long as there is not a minimum attack time enforced by the attack method applied (for instance, the time needed for performing a side-channel analysis, data collections, or the time needed for an epoxy to harden). In those cases where attendance is not required during part of the attack time, the attack time is to be taken as expired time divided by 3.

Expertise refers to the level of generic capabilities including but not limited to knowledge, skills, experience, etc. of the application area or product type (e.g., UNIX operation systems and Internet protocols). Identified levels are as follows:

a) **Layman** is a person without professional or specialized knowledge in a particular subject. They are unknowledgeable compared to experts, proficient, or skilled persons, with no particular expertise but capable of implementing simple attack steps, or capable of implementing more complex attack steps under direction. For the purpose of exploitation, they can implement an attack based on a script or a written procedure, without requiring any particular skill.

b) **Skilled** persons are able to perform more complex tasks and attack steps without direction. They have or can demonstrate the ability and training to perform a specific task well.

c) **Proficient** persons are (highly) competent and have the necessary ability, knowledge, and skill to perform complex attacks successfully. They are familiar with the security functionalities and behavior of the product. For the purposes of exploitation, proficient persons qualify when specific skills are required to conduct an attack successfully. Proficient personnel can acquire capabilities equivalent to the skill sets of PTS lab evaluators.

c) **Experts** are extremely knowledgeable and skillful in one or more areas**.** They are very familiar with the underlying algorithms, protocols, hardware components, physical and logical architectures, etc., implemented in the device or system type and the principles and concepts of security employed. Experts are capable of quickly devising new attack paths that require specific competencies similar to those of experienced PTS lab personnel.

If proficient expertise on various areas of technology is required for an attack—for example, on electrical engineering and cryptography—an expert level of expertise can be assumed. In most attack scenarios, the exploitation level of expertise is less than the required identification level of expertise.

Knowledge of the HSM refers to obtaining specific expertise in relation to the HSM. This is different from generic expertise but not unrelated to it. Identified levels are as follows:

a) **Public information** about the HSM (or no information): Information is considered public if it can be easily obtained by anyone (for example, from the Internet) or if it is provided by the vendor to any customer. Examples include open protocol specifications and electronic component datasheets. Information with automated access-control mechanisms (such as online account subscription) without human intervention classifies as Public.

b) **Restricted information** concerning the HSM (for example, as gained from vendor technical specifications): Information is considered restricted if it is distributed on request and the distribution is registered—for example, like the *PCI PTS HSM DTRs*. Restricted information is distributed upon request and is subject to human-based control mechanisms. Examples of restricted information are mechanical drawings for OEM device integration, external command API specifications, partial Gerber files, and secure processor datasheets available under NDA.

c) **Sensitive information** about the HSM (for example, knowledge of internal design, which may have to be obtained by “social engineering” or exhaustive reverse engineering). Distinction must be made between information required to identify the vulnerability and the information required to exploit it, especially in the area of sensitive information. Requiring restricted or sensitive information for exploitation would be unusual.

Access to the HSM is also an important factor. It is assumed here that the HSM would be purchased or otherwise obtained by the attacker and that beside other factors there is not any time limit in analyzing or modifying the HSM. Differences are defined in the status and functionality of the device to be analyzed/tested.

a) **Mechanical samples** are non-functional and are used merely to study the mechanical design or for supplying spare parts.

b) **Functional samples without working keys** might be used for the logical and electrical behavior of the device but are not loaded with working keys and are therefore not functional within a payment network or with real payment cards. Such devices might be regularly purchased. Please note that these also include devices fitted with test or dummy keys.

c) **Functional samples with working keys** are fully functional devices, which might be used to verify an attack method or to actually perform an attack.

If more than one sample is needed in any category, instead of multiplying the points by the number of samples, the following factors must be used:

Table A-1: Multiple Samples Factors

Number of Devices	Factor
1	1
2	1.5
3–4	2
5–10	4
> 10	5

Equipment refers to the equipment that is required to identify or exploit vulnerability. See Appendix B for details of equipment classification.

a) **Standard equipment** is equipment that is readily available to the attacker, either for the identification of vulnerability or for an attack. This equipment can be readily obtained—for example, from retail outlets or acquired/downloaded from the Internet.

b) **Specialized equipment** isn’t readily available to the attacker but could be acquired without undue effort.

c) **Bespoke equipment** is not readily available to the public, as it might need to be specially produced (for example, very sophisticated software), or because the equipment is so specialized that its distribution is controlled (for example, X-ray equipment), possibly even restricted. Bespoke equipment that can be rented must be treated as specialized equipment. Software that has been developed during the identification phase is considered as bespoke equipment and must not additionally be considered in the exploitation phase.

d) **Chip-level attack equipment**, necessary to implement chip-level attacks, is generally not widely available on the market and its access is restricted. It is also very expensive, and therefore is considered as a category on its own. Only the following equipment belong to this category:

- Focused Ion Beam
- Scanning Electron Microscope

If, for one phase (identification or exploitation), equipment from different levels is required, only the highest level shall be retained. Hence, only the highest level of equipment shall be counted in any one phase of an attack calculation.

Specialist expertise and equipment are concerned with there being an implicit relationship between an attacker's expertise and the ability to effectively make use of equipment in an attack. The weaker the attacker's expertise, the lower the potential to effectively use equipment. Likewise, the greater the expertise, the greater the potential for equipment to be used in the attack. Although implicit, this relationship between expertise and the use of equipment does not always apply—for instance, when environmental measures prevent an expert attacker's use of equipment; or when, through the efforts of others, attack tools requiring little expertise for effective use are created and freely distributed (for example, via the Internet).

Parts refer to components required to hide the signs of an attack; to otherwise replace components that have been broken during an attack, like a case part, a display, or a printer; to create a data-monitoring or communicating bug; or otherwise are needed to perform the attack. If the same part may be used for identification and exploitation, it must only be accounted for once.

a) **Standard parts** are readily available to the attacker, either by purchasing them from a supply store or by re-using parts from a mechanical sample of the same device.

b) **Specialized parts** are not readily available to the attacker but could be acquired without undue effort. These might be parts that are not readily available from a supply store but can be ordered from stock and require delivery time or a certain minimum component count for purchase.

c) **Bespoke parts** are not readily available and have to be specifically manufactured. It is very unlikely that an attack requires bespoke spare parts.

Parts used during the Identification phase that can be used in the initial exploitation must be counted fully in the Exploitation phase to equalize the attack potential value across all exploitations. While equipment readily lends itself to reuse for each exploitation, parts are typically a one-time use. Each exploitation should have the same attack potential value. Accounting for parts that are reused in the initial exploitation only in the Identification phase, or even splitting between the Identification and Exploitation phases, will result in the initial exploitation having a lower attack-potential value than the actual subsequent exploitations. Therefore, parts used during the Identification phase that can be used in the initial exploitation must be counted fully in the Exploitation phase to equalize the attack-potential value across all exploitations. If it is not readily reusable—the part once used in installation becomes unusable for exploitation because, for example, it is glued with epoxy and difficult to remove—it can be accounted for twice: once in the Identification phase and again in the Exploitation phase.

Rating Exploitation

It is important to note that only initial exploitation is considered, as further exploitations can reuse equipment and knowledge and are subject to optimization, which cannot be easily assessed through laboratory evaluations. Attacks are based on an attack performed on a single device. If the attack potential is met, the requirement is met regardless of whether or not applying the attack to additional devices is less than the attack potential.

The following items in calculation are typically subject to reuse in further exploitation phases:

- Equipment
- Knowledge

If several attack scenarios lead to the same potential in total and exploitation, the one that has the lowest cost in exploitation, exclusive of the items above, must be considered. Attack calculations must allocate ratings that assume the most conservative trade-offs between time, expertise, and equipment. Particularly, attack calculations shall not distribute ratings in a way that increases overall and/or exploitation minimum ratings above the most conservative approach.

An Approach to Calculation

The above section identifies the factors to be considered. The table below gives guidelines for the individual factors.

For a given attack it might be necessary to make several passes through the table for different attack scenarios (for example, trading off expertise for time or equipment). The lowest value obtained for these passes should be retained. In the case of a vulnerability that has been identified and is in the public domain, the identifying values should be selected for an attacker to uncover that attack scenario in the public domain, rather than to initially identify it.

Table A-2: Attack Potential Factors

Factor	Range	Identification Phase	Exploitation Phase
Attack time	≤ 1 hour	0	0
	≤ 2 hours	1	1
	≤ 4 hours	1.5	1.5
	≤ 6 hours	2	2
	≤ 8 hours	3	3
	≤ 12 hours	4	4
	≤ 16 hours	4.5	4.5
	≤ 24 hours	5	5
	≤ 40 hours	5.5	5.5
	≤ 60 hours	6	6
	≤ 100 hours	6.5	6.5
	≤ 160 hours	7	7
	Beyond 160 hours	7.5	7.5

Factor	Range	Identification Phase	Exploitation Phase
Expertise	Layman	0	0
	Skilled	1	1
	Proficient	3	3
	Expert	4	4
Knowledge of the HSM	Public	0	0
	Restricted	2	2
	Sensitive	3	3
Access to the HSM per device required for the attack. **Note:** *If more than one device is required, the values must be multiplied by the factors given above.*	Mechanical sample	1	1
	Functional samples without working keys	2	2
	Functional sample with working keys and software	4	4
Equipment required for the attack	None	0	0
	Standard	1	1
	Specialized	3	3
	Bespoke	5	5
	Chip-level attacks	7	7
Specific parts required	None	0	0
	Standard	1	1
	Specialized	3	3
	Bespoke	5	5

An approach such as this cannot take account of every circumstance or factor but should give a better indication of the attack potential. Other factors, such as the reliance on unlikely chance occurrences or the likelihood of detection before an attack can be completed, are not included in the basic model but can be used by a tester as justification for a rating other than those that the basic model might indicate.

Determining Applicable Time and Levels

For each phase, the testing laboratory shall document all necessary steps, including expertise, equipment, specific parts needed, and time required to operate (in hours).

This information is best summarized in a table containing all the items described above.

First Attack Example

The attack aims to penetrate through the casing of the HSM to expose and bypass casing switches. Once bypassed, the casing is removed and an internal mesh covering the HSM processing circuitry is abraded and shorted out, allowing for access to the bus carrying clear-text cryptographic keys.

Identification

1. Reverse-engineer the device to understand its design, including the tamper-detecting sensors and clear-text-key signal paths and storage locations. Expert level in electronics is required to understand the routing of security signals, as well as to understand the memory layout and specific locations for clear-text keys. Locations of tamper mechanisms are determined and

methods devised to bypass/deactivate them. It is therefore determined that 80 working hours, expert skills, standard equipment and one mechanical sample are required.

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Standard	Public	None	1 functional	80 hours

2. A method of attack is devised to bypass the tamper meshes and extract cryptographic keys from the system. In this example, multiple layers of mesh cover the sensitive areas and signals of the HSM, leading to a multi-stage attack. High speed memory is used in the system, requiring use of specialized equipment to capture the signals. The attack retrieves the main tamper key of the HSM, which is accessed on power on in this example, and therefore a bug is not required to be developed for installation into the HSM.
   Costings for this example are: Expert, 80 hours' work, specialized equipment, and reusing the same functional sample.

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Specialized	Public	None	None	80 hours

3. The attack is exercised on another functional device with test keys. Public knowledge of how to operate the HSM and load keys is assumed.

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Standard	Public	None	1 functional	40 hours

As a summary for the identification phase, the following would apply:

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Specialized	Public	None	2 functional	200 hours

<u>Exploitation</u>

Attacker uses a functional sample with keys and implements the attack. Several steps are necessary to perform the attack:

1. Attack the tamper-detection mechanisms to penetrate into the device. It was deemed during testing that the casing has tamper-detection switches, and each detection mechanism can be disabled with a success rate of 0.95 and requires 30 minutes per mechanism. In this example, there are eight tamper-detection casing switches, but only four of them need to be defeated. One additional hour is required for setup and finalization of this stage of the attack.

   While the attack is scripted, it nevertheless requires good mechanical skills and practice to successfully implement. Therefore, the level of expertise is considered as "proficient."

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Proficient	Standard (reused)	Public	None	1 working with keys	3 hours

2. Once inside, the attack needs to reach sensitive signals—e.g., clear-text cryptographic key signals—located behind tamper meshes within the device. The meshes need to be bypassed and disabled such that sufficient access is obtained to the internal memory subsystems.

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Standard (reused)	Public	None	1 working with keys	4 hours

3. Once the previous step is successfully achieved, the attacker will now attach the specialized equipment used to capture the cryptographic tamper key as the system is booted, and test the device. With the tamper key, no further use of the HSM is required and other keys can be retrieved and decrypted from the installed environment as required (or as previously captured).

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Proficient	Specialized (reused)	Public	None	1 working with keys	4 hours

As a summary for the exploitation phase, the following would apply:

Expertise	Equipment	Knowledge	Parts	Samples	Time needed
Expert	Specialized (reused)	Public	None	1 working with keys	11 hours

Table A-3: Attack Potential for Inserting a Keypad Signal-Disclosing Bug

Aspect	Identifying Value		Exploiting Value	
Attack time	> 160h	7.5	≤ 12 h	4
Expertise	Expert	4	Expert	4
Knowledge of the device	Public	0	Public	0
Access to HSM	2 functional samples	3	Functional sample with working keys	4
Equipment	Specialized	3	Specialized	3
Specific parts	None	0	None	0
Attack potential per phase		**17.5**		**15**
Total Attack Potential		**32.5**		

Second Attack Example

The attack aims at the determination of a TDES key used for encryption at the device using differential power analysis (DPA). It is assumed that:

- A function of the HSM is used which requires PIN translations using the cryptographic key under attack;
- The data used for DPA can be acquired at an external interface of the HSM module, e.g., the HSM needs not be further physically attacked to get the required test data; and that
- The HSM does not have effective countermeasures against DPA.

The attack would consist of the following steps:

1. Determine the method to run DPA on an HSM. This consists mostly of analyzing the electrical and logical interface. This step requires professional knowledge of electronic and computer engineering.
2. Develop the attack set-up including the control to run the HSM in an automated way. Since a large number of PIN translations are required. This is bespoke equipment, developed specially for this attack, which will be reused at Identification time.
3. Get an HSM and perform the measurement. We expect that at least 20,000 PIN translations and the associated decryptions/encryptions have to be observed. In the identification phase, this may have to be repeated several times. Since such an amount of translation operations cannot be performed surreptitiously in a real live environment, it must be possible to run the device off-line with a simulated host.
4. Analyze the data samples and retrieve the PIN-encrypting key.

The attack potentials are estimated within the following table:

Table A-4: Attack Potentials Example for DPA Analysis

Aspect	Identifying Value		Exploiting Value	
Attack time	Beyond 160h	7.5	≤ 8 h	3
Expertise	Expert	4	Expert	4
Knowledge of the device	Restricted	2	Public	0
Access to the device	Functional sample with trial keys	2	Functional sample with working keys	4
Equipment	Bespoke	5	Specialized	3
Specific parts	Standard	1	No further parts required	0
	Attack potential per phase	**21.5**		**14**
	Total Attack Potential	35.5		

Appendix B: Equipment Classification

Standard equipment

"Rule of thumb" followed: If it can be obtained easily (for example, over the Internet) for less than approximately US$1,000 or it can be reasonably expected that most people would have one or more (for example, a computer), it falls under this category. Most software is also to be considered standard, as it is highly likely that attackers will obtain illegal copies for free/little cost. Standard equipment would be expected for any attacks on the PCB or chip-carrier level, including exposure and attachment to chip bond-out wires. Unless otherwise indicated, the default classification for equipment is "Standard." Furthermore, as equipment classified above Standard becomes more readily available over time, it will be appropriately reclassified.

- Cutting blades (for example, scalpels, X-Acto knives, box cutters, axes, saws, etc.), including both conductive and non-conductive blades
- Dremel tool, attachments, extension cable, holder/mill jig, etc.
- Screwdrivers of any type—including custom tips, as they can be cut from another type of screwdriver simply enough if not able to be purchased
- Hammers, saws, pliers, clamps (G-type, screw type, and any others)
- Oscilloscopes with two channels and external trigger inputs—up to around 1GHz with some leeway either way depending on buffers, bit-depth of A/D, additional features, etc.
- Simple milling machine
- Drills and drill press, any drill bits down to 0.1mm
- Conductive and non-conductive ink
- PCB etching equipment
- PCB design software
- Simple side-channel software (capable of performing time-displacement alignment, correlation analysis, basic time-frequency conversion and filtering, etc., and having a basic functioning user interface)
- Acid of most types used for attacks, including for etching of chips
- Solvents, cleaning materials, plastic working tools
- Low-resolution 3D printers
- Soldering irons of any type, including with heated rework tweezers or temperature-controlled hot-air guns for SMT (re)work
- Mechanical probes, dentist's picks, tweezers (including non-magnetic), small mirrors, etc.
- Inspection microscopes up to 300x
- Multi-meters, continuity and resistance testers, IR and bi-metal temperature monitors
- Environmental chamber or equivalent heating/cooling tools (approximate range -80 C to 200 C)
- Current probes up to 1Ghz
- Simple EM probes (for example wire coils, antennas)
- Simple AM/FM receivers

- Custom jigs to hold devices/items during attacks (generally made from wood or 3D printed)
- Standard PC (either off the shelf or built from components)

PC-based instruments such as protocol sniffers, USB-attached oscilloscope adapters, and graphical multimeters are considered standard equipment, especially if they do not require dedicated hardware or adapters.

Specialized equipment

"Rule of thumb" followed: If it can be obtained for between approximately US$1,000 and US$50,000, it falls under this category. Specialized equipment would be expected for any side-channel work on a modern CPU (for example, 32/64 bits, clock frequency >100Mhz) or hardware implementation. The values stated below are indicative only.

- Expensive, high-resolution, high-frequency, deep-buffer oscilloscopes (> 1GS/s, 1GHz, 16-bit, etc.)
- High-resolution 3D printers
- High-resolution milling machines (for example, for chip planing)
- Complex side-channel software, able to perform complex alignment beyond linear shift techniques, and relatively advanced, noise removal, etc., in accordance with the tool capabilities described in Appendix F. It is expected that this will be run on a standard PC (Standard equipment)
- Micro-probes for attachment to die-level features such as bus lines on chips
- Glitching systems, for example EM fault injection
- High-frequency/high-bandwidth EM probes
- 16/32-bit high-speed logic analyzer for capture and analysis of bus traffic
- Dedicated electronic cards
- Specialized test benches
- Protocol analyzers
- Microprobe workstation
- Chemical workbench
- Laser abrading/cutting

Bespoke equipment

"Rule of thumb" followed: If it cannot be obtained for less than US$50000 or extensive customization is required for such an attack, it falls under this category. Bespoke equipment is expected to be required only in very rare circumstances.

- Plasma etching equipment
- Automated ("push-button") side-channel software to be used on a specific PED model to bypass strong countermeasures and leak keys from that specific model.
- Sophisticated X-ray 3-D imaging equipment

Chip-level equipment

"Rule of thumb" followed: Absolutely required for attacks at the feature level on chips (not bus level).

- Focused Ion Beam
- Electron tunneling microscope/Scanning Electron Microscope

Appendix C: Configuration and Use of the STS Tool

The NIST STS (Statistical Test Suite) is a reference implementation of the statistical tests described in *NIST SP 800-22 Revision 1a.*

The tester shall use NIST's STS tool, version 2.1.2 or later, or its mathematical equivalent. The tester shall verify that the compiled instance of the STS tool is operating correctly on the testing device by testing the NIST-provided sample data and comparing the results with those found in *NIST SP 800-22 Revision 1a* (SP800-22r1a), Appendix B. This configuration guidance is for use with STS version 2.1.2, though it will likely continue to be applicable to future versions.

A note on STS versions: *Prior versions of STS include bugs that have been fixed in the current version. Previous versions must not be used unless the critical fixes present in the current NIST tool have been backported. At a minimum, prior versions must disable the Lempel-Ziv compression test [Hamano 2009] and include fixes to the DFT (Spectral) test [Kim 2004], the Overlapping Template test [Hamano 2007], the Non-Overlapping test [NIST 2014], and the "Proportion of Sequences Passing a Test" test interpretation.*

The tester should request and obtain a sample of 2^{30} bits from the vendor. The tester should exercise care to verify that the vendor-supplied data is interpreted correctly by the STS tool (the STS tool assumes that binary data is in big-endian formatting on all devices).

The STS testing on the data shall be judged as a "pass" if it passes all of the tests, for both the "Proportion of Sequences Passing a Test" and "Uniform Distribution of P-Values" interpretation approaches. If the data does not pass all tests and the failure is marginal, the tester should acquire additional data from the vendor and repeat the testing using both the initial data and the additional vendor-supplied data.

The STS tool should be configured as per guidance provided in SP 800-22 Revision 1a, which is summarized below.

The following settings are consistent with the SP 800-22 Revision 1a document:

Configuration Item	Setting	Reference in Key Below
Length of bit streams (*n*)	1,000,000	[1]
Number of bit streams (sample size) *(M)*	1,073	[2]
Block Frequency block length	20,000	[3]
Non-Overlapping Templates template length	9	[4]
Overlapping Template template length	9	[5]
Universal block length (*L*), number of initialization steps (*Q*)	*L*=7, *Q*=1,280	[6]
Approximate Entropy block length	8	[7]
Serial block length	16	[8]
Linear Complexity block length	1,000	[9]

Key to Configuration Item Table Above

[1] n must be selected to be consistent with the requirements of all of the tests to be run. The Overlapping Templates, Linear Complexity, Random Excursions, and Random Excursions Variant tests all require n to be greater than or equal to 10^6 in order to produce meaningful results. The Discrete Fourier Transform (Spectral) test requires n to equal 10^6. (See SP800-22r1a Sections 2.8.7, 2.10.7, 2.14.7, 2.15.7, and [NIST 2010].)

[2] The number of bit sequences (sample size) must be 1,000 or greater in order for the "Proportion of Sequences Passing a Test" result to be meaningful. (See SP800-22r1a Section 4.2.1.) This value will be 1,073 for the first test, but any additional testing (for example, further testing to resolve test failures) will necessarily include more bit sequences.

[3] For the Block Frequency test, if $n=10^6$, the test block size should be set between 10^4 and 10^6. (See SP800-22r1a Section 2.2.7.)

[4] The Non-Overlapping test requires selection of a template length of 9 or 10 in order to produce meaningful results. (See SP800-22r1a Sections 2.7.7 and 2.8.7.) For a template length of 10, the MAXNUMOFTEMPLATES constant (in defs.h) should be set to at least 284 prior to compiling STS, otherwise most 10-bit aperiodic templates with a leading 1 bit are discarded.

[5] The Overlapping test requires selection of a template length of 9 or 10 in order to produce meaningful results. When $n=10^6$, the template size of 9 comes closest to fulfilling the parameter selection criteria. (See SP800-22r1a Section 2.8.7.)

[6] The Universal test block length (L) and initialization steps (Q) must be consistent with the table in SP800-22r1a Section 2.9.7. For $n=10^6$, the only acceptable values are ($L=6$, $Q=640$) and ($L=7$, $Q=1280$). Note: Any parameters passed into this test are discarded, and reasonable values are internally set. For $n=10^6$, STS automatically uses the parameters recommended here.

[7] For the Approximate Entropy (ApEn) test, SP800-22r1a Section 2.12.7 requires the block length to be less than $[\log_2 n] - 5$. Other analysis [Hill 2004] has shown that for n=1,000,000, block lengths greater than 8 can cause failures more often than expected for large scale testing.

[8] The Serial Test block length is also set based on n. If $n=10^6$, the block length must be less than 17. (See SP800-22r1a Section 2.11.7.)

[9] The Linear Complexity test block length is required to be set to between 500 and 5,000 (inclusive) and requires that $\frac{n}{M} \geq 200$. (See SP800-22r1a Section 2.10.7.)

References

[Rukhin 2010] Rukhin, Andrew, et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST SP 800-22, Revision 1a.

[Kim 2004] Kim, Song-Ju, et al., "Corrections of the NIST Statistical Test Suite for Randomness."

[Hill 2004] Hill, Joshua (InfoGard Labs), "ApEn Test Parameter Selection."

[Hamano 2007] Hamano, K. and Kaneko, T., "Correction of Overlapping Template Matching Test Included in NIST Randomness Test Suite," IEICE Trans. Fundamentals, vol. E90-A, no. 9, pp. 1788-1792, Sept. 2007.

[Hamano 2009] Hamano, Kenji, "Analysis and Application of the T-complexity." Ph.D. thesis, The University of Tokyo.

[NIST 2010] STS Software Revision History.
URL: http://csrc.nist.gov/groups/ST/toolkit/rng/revision_history_software.html.
Internet Archive:
http://web.archive.org/web/20150520193625/http://csrc.nist.gov/groups/ST/toolkit/rng/revision_history_software.html

[NIST 2014] Current STS Release Notes.
URL: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html.
Internet Archive:
http://web.archive.org/web/20150103230340/http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

Appendix D: Minimum and Equivalent Key Sizes and Strengths for Approved Algorithms

Approved Algorithms in connection with the requirements in this document are based on the approved algorithms listed in *NIST SP 800-57 Part 1 Rev. 4,* Section 4.

- **Hash functions:** Only algorithms from the SHA2 and SHA3 family are allowed, with output size > 255.[1] MD5 and SHA-1 are not allowed for use.
- **Symmetric-key algorithms used for encryption and decryption:** AES must be used, with key size >= 128 bits or TDES with keys size >= 112 bits.
- **Message authentication codes (MACs):** CMAC or GMAC can be used with AES, as well as HMAC with an approved hash function and a key size >= 128. TDES MACs must use *ISO 16609* MAC algorithm 1 using padding method 3, or MAC algorithm 5 using padding method 4.
- **Signature algorithms:** DSA, RSA (with PKCS1-v1.5 or PSS) and ECDSA with key sizes specified below.
- **Approved key-establishment schemes** are described in *NIST SP800-56A* (ECC/FFC[2]-based key agreement), *NIST SP800-56B* (IFC-based key agreement), and *NIST SP800-38F* (AES-based key encryption/wrapping).

The following are the minimum key sizes[3] and parameters for the algorithm(s) in question that must be used in connection with key transport, exchange, or establishment and for data protection in connection with these requirements. Other key sizes and algorithms may be supported for non-PCI payment-brand-relevant transactions:

	Algorithm				
	TDES	IFC (RSA)	ECC (ECDSA, EdDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Minimum key size in number of bits:	112	2048	224	2048/224	128

[1] Except as noted, the use of SHA-1 is prohibited for all digital signatures used on the device in connection with meeting PCI Security Requirements. This includes certificates used by the device that are non-device specific and part of a vendor PKI, up to and including a vendor root certificate. The only exception to this is that the initial code on ROM that initiates upon the device start may authenticate itself using SHA-1, but all subsequent code must be authenticated using SHA-2.

SHA-2 or higher is recommended for other usages, but SHA-1 may be used in conjunction with the generation of HMAC values and surrogate PANs (with salt), for deriving keys using key-derivation functions (i.e., KDFs) and random number generation. Where applicable, appropriate key-length minimums as delineated in the Derived Test Requirements are also required.

[2] IFC: Integer Factorization Cryptography; ECC: Elliptic Curve Cryptography; FFC: Finite Field Cryptography.

[3] Other key sizes and algorithms specified in this appendix may be supported for non-PCI payment-brand-relevant transactions. They are also not applicable to the EMV kernel; cryptographic requirements for EMV Contactless transactions are set by EMVCo and/or the Payment Brands.

Key-encipherment keys shall be at least of equal or greater strength than any key that they are protecting. This applies to any key-encipherment keys used for the protection of secret or private keys that are stored or for keys used to encrypt any secret or private keys for loading or transport. For purposes of this requirement, the following algorithms and keys sizes by row are considered equivalent.

	Algorithm				
	TDES	IFC (RSA)	ECC (ECDSA, EdDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Key size in number of bits:	112	1024	160	1024/160	–
	168	2048	224	2048/224	–
	–	3072	256	3072/256	128
	–	7680	384	7680/384	192
	–	15360	512	15360/512	256

TDES refers to TDES keys with non-parity bits. The RSA key size refers to the size of the modulus. The Elliptic Curve key size refers to the minimum order of the base point on the elliptic curve; this order should be slightly smaller than the field size. The DSA key sizes refer to the size of the modulus and the minimum size of a large subgroup.

TLS implementations must prevent the use of cipher suites that do not enforce the use of cryptographic ciphers, hash functions, and key lengths as defined in the Technical FAQs.

For implementations using FFC or ECC:

- **FFC implementations entities** must securely generate and distribute the system-wide parameters: generator *g,* prime number *p,* and parameter *q,* the large prime factor of (*p* – 1). Parameter *p* must be at least 2048 bits long, and parameter *q* must be at least 224 bits long. Each entity must generate a private key *x* and a public key *y* using the domain parameters (*p, q, g*).

- **ECC implementations entities** must securely generate and distribute the system-wide parameters. Entities may generate the elliptic curve domain parameters or use a recommended curve (see *NIST SP 800-186*). The elliptic curve specified by the domain parameters must be at least as secure as P-224. Each entity must generate a private key *d* and a public key *Q* using the specified elliptic curve domain parameters. (See *NIST SP 800-186* for methods of generating *d* and *Q*.)

- Each private key must be statistically unique, unpredictable, and created using an approved random number generator as described in this document.

- Entities must authenticate the FFC or ECC public keys using DSA, ECDSA, a certificate, or a MAC. (If using a MAC, see *ISO 16609 – Banking – Requirements for message authentication using symmetric techniques*. One of the following shall be used: MAC algorithm 1 using padding method 3, or MAC algorithm 5 using padding method 4.)

IFC, FFC, and ECC are vulnerable to attacks from large-scale quantum computers. In 2017, NIST initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms, planned to end with a selection of new algorithms by 2023-2025.

Because of rapid progress in the field of quantum computing, it is advised to become informed/aware of this specific threat and its potential mitigations.

Appendix E: Side-Channel Analysis Standards for PCI-PTS Evaluations

***Note:** This appendix is for use in conjunction with Requirement A5, "Non-Invasive Attacks for Cryptographic Keys."*

Objectives

Attackers' objectives are to compromise high-value cryptographic keys in a device by finding leakage. Testers' objectives are to search for paths to key leakage to be able to validate resistance to attacks. A test performed in accordance with the DTRs can result in no detectable leakage, may show how a key can be fully exposed but at an acceptable level of difficulty, or may produce a partial result. In every case, a test determining compliance will demonstrate that an attack in the field is infeasible with respect to PTS Security Requirements' pass/fail thresholds.

This appendix defines baseline expectations for PTS-recognized Laboratory competencies on side-channel analysis. The PCI PTS standard assigns the highest rating level for attack calculations for the protection of PIN-security-related keys; side-channel analysis is often a feasible method to compromise these assets. This appendix shall be used to provide consistency so that for all evaluations, all laboratories assess the effort of the side-channel component of an attack on any device similarly. The summary information below outlines expectations of PTS-recognized laboratories' competencies—that is, what laboratories must possess; what evaluations need to do; what reports should demonstrate. This information is also a reference for laboratory recognition/maintenance, assessing the quality of evaluation test reports, and assessing laboratories' competencies.

Introduction

Side-channel analysis (SCA) is an important activity in PCI PTS evaluations, relevant to tests where the highest levels of confidence in security assurance are needed, as part of DTR A5 In these tests, it is necessary to derive an accurate measure of the effort to defeat a device using SCA techniques, distinctly from any physical penetration/modification.

Evaluations do not have to fully replicate SCA as would be applied in the field; a targeted and focused "quick-scan" approach based on optimized practices capable of assessing key leakage is sufficient. The information below provides guidance and a framework for testing. In no circumstances should this be interpreted as a formula or checklist of necessary procedures.

PCI SSC expects all evaluation test reports to consistently provide good evidence of valid test results. Results must derive from robust testing and appropriate evaluation methodologies. The necessary foundation for a PCI PTS-recognized Laboratory's capability to deliver this includes the laboratory's core competencies in performing SCA: equipment, skills, experience, and capacity.

PCI SSC expects these competencies to be maintained and applied at an equivalent level to attackers capable of exploiting devices (including approved devices) using contemporary SCA-related methods and techniques.

This appendix provides an outline of SCA criteria to indicate PCI SSC's expectations regarding PTS-recognized Laboratory capabilities. This does not define a complete scope of the application of SCA and should not be interpreted as the boundary of what is adequate. This appendix provides supplementary information that can be referred to by evaluators, but it is not a tutorial. This memorandum may be used as a guideline for structuring tests and providing test evidence in reports.

The Laboratory General Requirements document includes relevant information on laboratory maintenance (competencies, audit processes, laboratory assessment of test artifacts, etc.). Other relevant documents include (but are not restricted to): DTRs, FAQs, Program Guide, Delta Evaluations Scoping and Guidance, and Report Templates.

Resources

All PCI PTS-recognized Laboratories must be capable of performing SCA with an attack potential that can distinguish compliant from non-compliant devices. This capability relies on a strong foundation of combined competencies that must be available for any test: skills, knowledge, and experience; collection and analysis equipment (both hardware equipment and software tools).

Laboratories' SCA resources must be maintained at levels consistent with industry "state-of-the-art," capabilities, including (but not restricted to): cryptanalysis of all relevant algorithms, higher-order DPA, and template attacks. Laboratories' resources must be improved regularly to reflect updates to the PCI PTS program, for example as version changes occur and/or as new attacks are published. Labs must have available a sufficient volume of personnel and equipment resources to be capable of performing SCA at any capacity of evaluations the lab undertakes.

The best SCA tools, even implementing advanced automation, are useless without a proficient user. PCI SSC expects laboratory personnel to generally participate in SCA-related activities such as (but not restricted to): training, teaching, research and development, contribution to new attack paths and techniques, and participation in relevant industry events. The following list outlines minimum expectations for evaluators.

Minimum expectations for evaluators

- Personnel performing SCA have a thorough understanding of known attack methods and techniques.
- Personnel performing SCA are sufficiently skilled and experienced to devise and implement new approaches.
- Personnel are experienced and capable of analyzing various types of devices, to perform SCA efficiently and optimally, adapted to different situations.
- Personnel performing SCA have a proven track record of being able to defeat devices, including successful removal of SCA countermeasures to extract unknown cryptographic keys.

PCI SSC expects laboratory hardware equipment and software tools to be fit for purpose for performing SCA. These resources must be sufficiently well developed to enable testers to determine devices' strengths and/or vulnerabilities, and to be able produce evidence of this. The following list outlines minimum expectations for test resources.

Minimum expectations for test resources

- Configurable collection hardware and control software are capable of acquiring and storing high-resolution power end electromagnetic side-channel recordings.
- Triggering and noise-removal collection capabilities are able to overcome typical obstacles to acquiring good recordings, to achieve stable timing and good signal quality.
- SCA toolsets are demonstrably effective in analyzing, modifying, and extracting hidden information from side-channel recordings, including the capability to identify and remove SCA countermeasures.
- SCA toolsets are adaptable and extensible according to variable test situations, providing effective user interfaces and user-configurable functionalities.

- SCA toolsets are updated incrementally to improve performance, increase attack potential, remove bugs, and identify and resolve usability issues.

Laboratory capabilities related to any of the resources outlined above must correspond closely to information expressed in current versions of PTS documents and to information releases from PCI SSC.

Principles and methods for testing

PCI SSC expects that the overall quality and effort required for SCA testing to conclude device compliance is comparable to the criteria described here.

SCA must not be hindered by lack of basic resources PCI SSC considers to be essential:

- Notwithstanding physical protections (such as those validated by DTR A1), approved devices must not contain cryptographic implementations which, under analysis, can be straightforwardly compromised through SCA.
- Devices provided for evaluations must be adequately prepared by the vendor for efficient lab testing.

SCA tests must not be hindered by the types of obstacles that experienced attackers having contemporary tools can overcome straightforwardly. The following, non-exhaustive list gives examples of factors related to PCI SSC's expectations in testing.

Factors related to testing expectations – examples

- Relevant information from the device vendor, such as: source code, functional documents, and other explanations of functionality, including accurate identification of hardware and software.
- Relevant assistance from the vendor, such as: devices adapted as necessary to support collection interfacing, signal acquisition access, collection triggers, etc.
- Availability of oscilloscopes, interface tools, workstations, RAM, data storage, etc.
- Signal-analysis capabilities such as GUIs that can display, manipulate, synchronize, superimpose, and compare multiple waveforms, including waveforms representing recordings and results derived from analysis of recordings. *
- Signal-processing capabilities to identify and overcome obstacles to SCA such as timing variation or other characteristics of recorded or derived waveforms, including deliberate countermeasures or other noise that may obscure information. *
- Waveform-alignment capabilities that can accurately and efficiently synchronize specific sensitive events, or operations, occurring in algorithms under analysis.
- Waveform-leakage investigation capabilities, such as correlation, that can accurately and efficiently plot the dependencies of various data values processed in algorithms under analysis. *
- Configurable libraries for attacks on known or unknown cryptographic keys, implementing DPA and related cryptanalysis methods, and implementing robust analysis methods for: different leakage models, results interpretation, and graphing. *
- SCA toolset user interfaces and features such as: scripting, automation, process batching, tabulation of results, precise graphical rendering of recorded/filtered/results waveforms, information handling, and software-development capabilities, allowing evaluators to adapt analysis techniques and create new analysis methods. *
- Appropriate skills, experience, training, and available time for evaluators.

- Validation of any claimed SCA countermeasures as being both active and effective. Without these validations, tests cannot claim strong reliance on countermeasures
- Validation of appropriate test methodologies sufficient to give a high degree of confidence that the device cannot be compromised using SCA methods similar to or different than those used in testing.

** Examples of specific attributes of SCA toolsets satisfying these expectations are listed below*

Critical steps in SCA scenarios

Even in straightforward scenarios—for example, in the absence of SCA countermeasures—several fundamental activities are necessary for SCA to succeed. Critical steps that should occur in most SCA scenarios are identified below.

- **Understanding of the algorithm(s) to be attacked and how they are implemented:** Determining the appropriate side channel(s) to acquire, collection method(s) and analysis tools, and their configuration. Next, validating that the device behaves as expected before proceeding further—if necessary, gathering additional information and/or reconfiguring the test setup as required. This includes validation of hardware/software identifiers and claimed countermeasures.
- **Preparing the device:** While many of the considerations for the assessment on device physical penetration or modification are relevant to performing SCA, PCI SSC expects that in most cases the vendor's assistance must significantly ease the operation of the device and/or a specially prepared subset of the device (such as the security processor abstracted from the device). An important part of the evaluation is to determine which of these possibilities is optimum to assess leakage of the cryptographic implementation (hardware/software).
- **Acquiring stable (well-triggered) recordings at an optimum sampling rate for a particular analysis task:** The sampling rate should not be too low as to introduce aliasing, particularly if recording waveforms to input to attacks such as DPA/DEMA. Acquisitions should be set up to optimize SNR, amplitude, and time resolution. The evaluation should aim for well-timed, low-noise, high-resolution acquisitions appropriate to a particular task and commensurate to resources and techniques available to expert attackers.
- **Characterizing side-channel behavior through approaches such as SPA/SEMA:** A critical goal of this phase is to identify events in the algorithm under attack, to localize sensitive operations. Specific checks are necessary for certain test methods—for example, for EMA the relative location of the EM probe to the device's components is a highly sensitive variable factor. In many situations no useful features can be determined upon first inspection. In this case, it is necessary to explore either whether there are obstructions to SPA that can be removed and/or how the collection can be improved.
- **Alignment:** Time synchronization is a crucial step in almost all SCA attacks. For example, DPA attacks can often succeed straightforwardly after a small number of CPU clock cycles (or sample points) have been well synchronized, but not without achieving this. Moreover, many aspects of timing variation in waveforms that outwardly appear to obstruct analysis—for example randomly occurring delays—can in practice be removed straightforwardly. Hence, the evaluation should utilize the types of various alignment techniques available to expert attackers. A single application of alignment processing is often insufficient. In most situations, when a simple, approximate, or limited-scope alignment test is performed, that approach is insufficient unless highly effective obstructions to alignment can be demonstrated.

- **Correlation:** Computations such as (but not restricted to) the Pearson product-moment coefficient are a crucial element in a test's capability to achieve alignment. Effective correlation capabilities are also crucial for identification of leakage that may often be obscured, for example to identify security-significant patterns in waveforms, test a device's relative susceptibility to leak internally processed data, localize sensitive operations for making well-targeted attacks, launch and develop attacks to infer secret data values. The evaluation should use correlation-analysis techniques equivalent to those available to expert attackers. If no correlation between a device's side-channel behavior and any non-sensitive/clear-text data values can be found, a possible explanation for this is that the test setup is not configured appropriately—for example, I/O data values unassociated directly with sensitive operations can be expected to leak unless there is a valid explanation otherwise. In most situations a test finding no leakage of any kind is insufficient unless valid reasons for this are identified.

  Establishing definite correlation of non-sensitive I/O data is important to (1) validate that the collection setup and test methodology are not flawed; (2) localize sensitive cryptographic operations in developing attacks; and (3) demonstrate that cryptographic key data is more resistant to leakage than other data. The expectation is for most tests to be capable of establishing definite correlation of non-sensitive I/O data. Very strong justification for null correlation results is needed otherwise, in asserting device compliance.

- **Signal analysis and processing:** All side-channel recordings will contain a certain degree of noise in acquired waveforms—for example, ambient noise sources in the test apparatus and environment, including a device's physical architecture. Any noise source can be expected to introduce artifacts into side channels that obstruct an attacker seeking to identify exploitable signals such as: SPA/SEMA patterns, correlation leakage pinpointing sensitive operations, and key-dependent leakage. Many types of hardware-generated and algorithmic SCA countermeasures exist. When implemented appropriately, individual countermeasures that obscure sensitive signals can significantly delay or prevent SCA attacks. Combinations of well-implemented countermeasures therefore provide greater defense-in-depth. Nevertheless, unintentional leakage may become straightforwardly detectable when noise is removed. It is possible to utilize design information and source-code review, to some extent, to infer the likely effectiveness of obstructive noise. However, there is always a possibility that obstructions may be removed by analyzing and modifying recordings to reveal hidden signals. The degree of success for an attacker to achieve this can rarely be determined from simply viewing the appearance of initially acquired waveforms; obstacles may be trivial to overcome for an experienced attacker prepared to thoroughly investigate noise removal. Therefore, evaluations should usually include several activities such as signal filtering to explore and demonstrate the quality of claimed countermeasures, and to optimize testing.

- **Cryptanalysis:** Attacks such as DPA succeed by discriminating a few inferences of guessed key values from among many other possible values. One specific algorithm's key-leakage potential is unlikely to be similar to another algorithm or implementation. Detectable leakage is often highly dependent on the options an attacker pursues. Hence, a simplistic implementation of a published attack, alone, is often insufficient to validate a device's compliance and the evaluation should make use of attack-modeling options available to expert attackers. Examples of these options include but are not restricted to selection functions (the calculation event in an algorithm under attack), statistical discriminants (the bit value(s) or Hamming weights, etc. being searched for), correlation functions (e.g., Pearson product-moment coefficient, differencing, and others), and options for ranking key guesses and modeling known key value leakage. In most situations, when a small number of options for key searching are attempted, it cannot be assumed that attempting other options would not succeed, without strong justification.

- **Optimization:** There are many approaches that will improve the chance of SCA success, considering the application of critical steps such as those outlined above. Depending on the discoveries made early in analysis, it is likely that some of the steps performed should be adapted, branched, and repeated—for example by iteratively re-applying well-chosen processes. Large data size may become problematic in some situations; however expert attackers are skilled in overcoming this unless robust obstacles are present—for example, analysis time may be significantly reduced by waveform compression filtering. To minimize elapsed time, in many situations analysis should be optimized by performing pilot tests, before launching SCA on larger data sets, through analysis of appropriately chosen sample point ranges, etc. Additionally, there are many computationally expensive analytical tasks that may be parallelized and run efficiently in the background (utilizing elapsed time) while appropriately focused analysis (directed effort) occurs.
- **Validation:** It is necessary for evaluations to check that critical steps performed in tests have the potential to succeed. It is often difficult to distinguish between a device's apparently robust resistance to attacks versus results deriving from a flawed or naive testing approach. For example, the absence of significant correlation leakage may be due to several situations that produce similar outcomes: errors in correlation calculations, bugs in analysis tools, inappropriate choice of analysis parameters, poor alignment, under-sampling or poor noise filtering, electronic noise introduced by the collection unintentionally, electronic noise deliberately generated by designed hardware countermeasures, obstructive noise from algorithmic signal-randomizing countermeasures, countermeasures for mathematical masking of specific sensitive data values, etc.. Hence, tests should include processes to differentiate between valid results or null results and demonstrate this.

Testing rationale and reporting

The outline information above defines many important aspects of testing. This should not be interpreted as an exhaustive list or as a definitive formula. PCI SSC expects SCA to be well targeted; that is, focused analysis considering the device's characteristics and making best use of adequate evaluation resources. It is not expected that all of these examples of techniques should be applied in any particular evaluation. More exactly, evaluations should demonstrably produce results deriving from well-balanced decisions corresponding to appropriate aspects of best practices.

In some situations it is likely that testing from one evaluation can be reused straightforwardly in another evaluation. This situation occurs commonly when two devices having similar characteristics are evaluated by the same laboratory in parallel or in close succession. Other situations exist where SCA-test reuse, or partial reuse, is appropriate, including reuse of third-party testing. PCI SSC encourages optimized reuse of SCA by incrementally extending tests on similar devices evaluated sequentially. In these cases, enhancements and improvements in each new evaluation must be proportionate to these criteria and must be evident. In all circumstances where testing is reused, it is very important that the equivalent validity of reused testing to an entirely full-scope test can be demonstrated.

The outcome for an evaluation's SCA testing is to either defeat the device emulating a realistic attack scenario or derive a definitive rating showing that the effort for an attack in the field is clearly above the necessary threshold. If the latter is not achieved in a test, detailed evidence is necessary to demonstrate compliance otherwise. The evaluation test report is the principal resource for demonstrating compliance to Security Requirements. The report must be self-consistent and adequate to justify compliance on its own. Therefore, the report must provide good-quality evidence related to this appendix. This includes (but is not restricted to):

- Accurate identification of the device's cryptographic implementation (hardware and/or software) and accurate identification of any reused testing.
- Descriptions and justifications of valid test methodologies.

- Presentation of results including details of the testing performed and the device's behavior under test.
- Strong justification of conclusions showing compliance.

Overall scope of SCA testing

Generally, all the following secret and private keys, and related algorithms, are to be considered in scope of SCA testing:

- Encryption of cryptographic keys or sensitive data
- Remote key loading,
- Local key protection (internal and external memory encryption),
- Authentication (e.g., firmware), and
- Software loading.

Determining test applicability

Valid statements of compliance cannot be made based on documentation/source code alone; applied testing is needed. Therefore, in principle, all algorithms used for protecting the assets listed above must be analyzed using SCA to some extent. At a minimum, this means recording the algorithms' side-channel profiles and validating that the profiles agree with the algorithms' expected characteristics.

The evaluation should determine at least one algorithm to analyze thoroughly, based on a rigorous assessment of asset value versus feasible attacks. Reasoning for not testing any algorithm has to be explained. This reasoning should include reliable assumptions made in the vulnerability analysis based on asset value and attack complexity—for example, limits on collections such as delay insertions or key-usage counters, and any additional countermeasures.

Prerequisites

To facilitate testing, the vendor must provide "open" samples (with a wire or pin attached for triggering and, if applicable, attachments to locations agreed with the lab for power measurements). The vendor must provide specially adapted firmware to allow cryptographic operations' input/outputs to be collected. For both EMA and power consumption (if possible), the measurement shall be performed directly at the chip/processor.

Effort to complete the overall device side-channel analysis

The descriptions below outline typical parameters that are acceptable for testing. Good attackers are creative and will not limit their efforts to below pre-defined limits such as disk storage size or an exact number of acquisitions, etc. PCI SCC expects the degree of any key leakage to be quantified using effective techniques (e.g., correlation, key attacks related to DPA, etc.) that are in agreement with these typical parameters but without reliance on parameter values as the primary justification for compliance verdicts.

For evaluating cryptographic implementations developed by the device vendor, the overall side-channel analysis effort should be expert-level work, assuming that well-prepared samples are available for analysis with effective tools, and assuming that testers are skilled in applying robust methodologies using these tools. Elapsed time is four weeks in most situations—for example, to allow collections and computationally expensive processes to run automatically. EMA is generally expected to provide better results than other side channels; it is sufficient for a lab to rely on EM analysis only, if this decision is well-reasoned. Data-collections size acquiring at least 100,000 high quality, high SNR traces is sufficient.

It is very likely that a significant part—even the majority—of the test effort will involve processing data (e.g., iteratively collecting, analyzing, filtering, aligning, performing correlation trials) following initial collections to optimize inputs to key attacks.

Testing from separate evaluations meeting these best practices criteria can be reused, provided that (1) testing is no longer than from the previous major version of the standard (e.g., v3.x work can be reused if appropriate as part of a v4.x review; and (2) a complete chain of trust is demonstrated validating why previous testing is wholly applicable to a newly evaluated device. For “System on Chip” (SoC) type cryptographic implementations, where expert-level work, multiple effective SCA countermeasures, and collection sizes in excess of 1 million traces, etc., are demonstrated, test reuse will generally be straightforward to apply.

A good test may discover total, partial, or zero key leakage, feasible in a field attack scenario, and/or feasible in the white-box context of the evaluation. In situations where a feasible attack is known to be capable of exposing the key, the evaluation shall explain definitively the effort needed to extract the key and how compliance is assessed considering obstacles to key-leakage attacks. In assessing this, the evaluation must consider and justify how any degree of key leakage is acceptable considering that successful attacks may require a lower degree of effort than the effort discovered through testing.

Reusing results

Tests identifying little or no correlation and/or algorithmic structure are generally insufficient for reuse. Since multiple devices commonly use same security processor or System on Chip for cryptographic operations, it can be useful to reuse results from other evaluations to decrease the efforts for side-channel analysis. Nevertheless, to ensure the transferability of results, the following points have to be considered:

- SCA results reused from other reports must not contain less information than required by the guidance
- The measurement setup of the reused analysis has to be described in a way that it can be justified that this setup also applies for the device under evaluation.
- The identification and configuration of the cryptographic component has to be compared to ensure that the results also apply to the device under evaluation—e.g., registers for enabling/disabling counter measures, etc.
- If results from third parties are reused, these third parties have to be approved PCI-labs. It is mandatory that the reports, respectively the SCA part of the reports, are provided to the evaluation lab that wants to reuse the results.
- It has to be justified that the results remain valid. For example, no new attacks or measurement techniques are known which could impact on compliance.
- It should never be assumed that the use of the same chip will yield the same results given various implementation options across devices—for example, usage differences in the software libraries, compilers, hardware-versus-software engines, power filtering, PCBs, etc.

Examples of appropriate SCA tool capabilities

- Acquisition and storage for waveforms and associated data values recorded through interfaces to the test device and oscilloscopes.
- Automation and control for x-y-z positioning and scanning and configuration where EMA is being investigated.
- Scripting capabilities for waveforms' acquisition, analysis, processing, cryptanalysis, including factors such as command/response I/O, storage, and data editing.
- Display and rendering of multiple waveforms (both acquired waveforms and derived waveforms and derived graphs of results) at high resolutions with functions for: scaling, zooming, overlapping, etc.
- Alignment functions capable of matching and displacing waveforms against a reference pattern, including configurable acceptance/rejection criteria.
- Alignment functions capable of adapting/modifying waveforms in situations where straightforward matching and displacement is ineffective.
- Alignment functions capable of generating, storing, and applying waveform displacements and using references for matching derived from separately filtered waveforms.
- Correlation functions capable of calculating and plotting leakage of internally processed data (single and multiple bits) values across time-series and spectral waveform sets.
- Correlation functions capable of calculating and plotting matches between selected sample-point ranges within a waveform and across time-series and spectral waveform sets.
- Multiple cryptanalysis libraries such as first-order and higher-order DPA time-series sets and spectra sets, applicable to various algorithms used by evaluated devices, including configurable parameters to model different selection functions and discriminants, and capable of producing, analyzing, and graphing results.
- Waveform analysis and filtering capable of producing and plotting frequency spectra derived from time-series waveforms.
- Functions capable of combining, removing, deriving, or otherwise operating on waveforms and spectra within one set and between different sets.
- Waveform and spectrum filters capable of selecting, counting, removing, summing, or otherwise operating on sample points, amplitude ranges, or data values against configurable thresholds.
- Waveform and spectrum filters capable of calculating and plotting statistical analysis of data set, such as variance, standard deviation, average, absolute or values, etc.
- Waveform and spectrum filters capable of producing and plotting frequency-filtered outputs such as: high-pass, low-pass, band-pass/reject, harmonics series pass/reject, etc.
- Compression functions capable of reducing waveform storage size with minimal reduction in SNR.
- Ability to apply processing operations to signals during waveform acquisition.
- Ability to apply operations on signals in batches/sequences, and to parallelize and otherwise automate time-consuming processes efficiently.
- Ability for the user to configure the functionalities described above, including the ability to configure through GUI(s).
- Ability for the user to extend the functionalities described above, including the ability to develop innovative software.
- Ability for the user to output information, including graphical information, of sufficient detail and quality to demonstrate analysis findings, for example in evaluation test reports.

Glossary of abbreviations related to side-channel analysis

DEMA	Differential electromagnetic analysis
DPA	Differential power analysis
EMA	Electromagnetic analysis
SCA	Side-channel analysis
SEMA	Simple electromagnetic analysis
SNR	Signal-to-noise ratio
SPA	Simple power analysis

Appendix F: Domain-Based Asset Flow Analysis

Introduction

Modern IT solutions, including PCI HSM, have reached a level of complexity that sometimes renders seemingly obvious questions as incomprehensible. A central question focuses on which components should be protected against which attack potential. In current solutions this often is hardly obvious and sometimes even surprises the developers.

The general idea is to determine when a certain asset is available at a certain location. Any hardware component or software module dealing with the asset will be virtually marked with the domain that the asset belongs to. We call this imaginary process **"tagging."** If a component or module is already tagged, it will keep the domain with the higher attack potential associated. Once the entire Asset Flow Analysis is performed, the appropriate domain will be assigned for each software module, hardware component, and even PCB track. This allows the tester to apply the appropriate DTRs for the specific domain.

It is therefore expected that the developer of the PTS device will perform this Domain-Based Asset Flow Analysis and provide the results and a proper explanation to the testers. The test lab will verify the analysis and use the effective domain rating as input for the evaluation.

For the vulnerability analysis—e.g., performed during attack costing according to Appendix B—it is furthermore vital to understand whether and when which asset may be attacked in a certain location. Obviously, the Asset Flow Analysis also yields this information.

Assets and Domains

"Domains" is used herein as a shortened term for "security domains group assets" with similar protection requirements. Such domains are easily reflected by similar attack cost thresholds. While in general domains with the same requirements concerning attack potential might have to be kept segregated to reflect different administrative requirements, such a structure is as yet not foreseen in PCI HSM. Therefore, the domain hierarchy is linear—i.e., any domain can always be represented by another domain with higher attack resistance. In effect, a developer can easily declare the entire hardware and software to belong in the Key domain. This will save the developer from performing an analysis; on the other hand, some modules or components are likely to fail the protection requirements imposed by the Key domain.

The PCI HSM criteria defines various assets with corresponding protection requirements. Assets are defined by data and which kind of access it shall be protected against. Integrity means that any modification to the asset, whether spurious or induced, shall prohibit its being processed any longer. Confidentiality means that clear text of the data must not be disclosed. In Table F1 below, four Security Domains are defined from higher-protection requirements to lesser requirements:

Table F-1: Security Domains Defined

Protection Level	Domain	Description
Higher Protection	**Key**	▪ PIN-related cryptographic keys are protected to resist an attack potential of 35 points. All keys must maintain integrity; secret keys also require confidentiality. This pertains to both storage and usage of these keys. ▪ Although account-data protection keys require a lesser attack potential—i.e., 26 points—this is still in excess of the related security, requiring 20 points or less. Therefore, keys constitute a separate asset class in this environment, too.
↕	**PIN**	▪ The cardholder PIN is protected to resist an attack potential of 26 points in DTR A1. PINs must not be disclosed. Passwords meet the same category.
	Data	▪ This class covers all other transaction data—e.g., PAN— The DTRs protect these assets at 20 points or less.
Lower Protection	**Vendor**	Any other data or functionality that is not related to the PCI HSM DTRs. Compromising these assets cannot compromise PCI HSM security. The corresponding security policy is largely at the discretion of the vendor.

It is vital that any lower domain cannot access any higher domain other than by dedicated, confined, and well-specified interfaces provided by the higher domain or any of its parents. Explicitly, the Vendor domain has no access beyond itself, meaning that it is not possible by any change of the code of this domain to gain any access to assets in that domain or change the behavior of that domain. It is not sufficient that the access is not foreseen, not implemented, or somehow prohibited by the code of the domain itself. If, for example, an application of the Vendor domain requires user input, it must interface with the Data domain. The latter will enforce that this access cannot compromise the security requirements of PCI HSM.

If confidential assets are encrypted using keys of at least the next higher domain, the encrypted asset is not a confidential asset. Keys from the Key domain may encrypt other keys of the Key domain. Note that the encrypting keys must at least have the same cryptographic strength as the keys they encrypt.

Assets that require integrity protection may be accompanied by integrity/authenticity tokens—e.g., MAC, HMAC, or a signature. The critical time for an attack is the time span from the token validation to the processing of the asset. Additionally, unbundling must be prevented—i.e., it must not be possible to use a foreign token with the asset.

For example, it is valid to store keys in a domain of lesser security—if the keys are encrypted and MAC'd, the MAC contains the key name, and the keys and algorithms used for encryption and MAC are adequately strong. The modules and components performing the cryptography and processing the validated or clear-text assets necessarily belong to the Key domain. This represents the idea of firmware in previous PCI HSM requirements.

Asset data that is properly protected by encryption and/or validation token, will be called an **"asset container"** in the remainder of this appendix.

Firmware and Process Spaces

While hardware tagging is quite simple—i.e., any single component may or may not process or store an asset—the situation is more complex with software. In this section we define some terms.

Any code that could potentially endanger any asset—e.g., in case of malfunction or even complete replacement—**is considered "firmware" in the sense of the DTR**. We define "**code"** as any instructions for a processing hardware, including microcode or netlists for programmable hardware, and any kind of data that may affect the processing by these instructions. Such data can be as simple as configuration bits, whether or not a certain security function is enforced. In PCI HSM such data often is interpreted code, from simple access control rules represented as data for an engine up to large amounts of code—e.g., Java byte code in Android-based systems. As soon as data has the potential to compromise any of the DTRs, this data is considered firmware—i.e., code.

We define a process space as a container for code. Within the same process space there is no mechanism to ensure that any deliberate code change in one place cannot affect processing in another place. There are three established methods to segregate process spaces:

1. Segregation of process spaces by hardware support is well known from operating systems. Standard concepts comprise separated processors or CPU with memory management and execution modes of graded privileges. The proper configuration of the segregation hardware may not be overly simple, but in general it should be feasible to verify its effectiveness.
2. Segregation of process by simulation leads to a virtual hardware support. Standard concepts are interpreted programs like Java. Since such solutions often allow calls to native libraries, i.e., which are not interpreted, and the interpreter itself may be quite complex, in general it is much harder to verify that such segregation is actually effective.
3. Mechanisms must exist such that any memory released from a process space and reused or re-acquired by another does not contain any asset data.

Obviously, this implies that any loaders or managers for process spaces are effectively executed in the same process space as their children. Eventually any initial boot loader is the parent of all process spaces on the same hardware.

Hardware Tagging

All of the assets discussed above enter the PCI HSM at some time using external interfaces, i.e., the devices are not produced with the assets built-in already.

- Consider the device in operational state and model each transaction type.
- Identify all interfaces involved in conveying assets or asset containers.
- Follow the path of asset containers until these are decrypted or validated.
- Consider explicitly how assets are conveyed internally between components.
- Define logical groups of hardware, including components, tracks, and wires carrying an asset with the corresponding domain.
- Define for logical interfaces a set of electrical signals and the passive components attached.
- Present all logical interfaces in a block diagram.

Software Tagging

A proven method to identify the domain for any piece of code is the following asset flow analysis algorithm:

1. Consider the software in operational state and perform all transactions—e.g., key loading, online PIN translation, key export, etc. Follow the incoming data from each external interface until messages are sent on external interfaces. Consider where assets or parts of them pass through the firmware. Tag each module with the corresponding asset domain.

2. For each module tagged, identify modules in the same process space and tag them with the highest asset domain found for the current module. The result is the domain structure in operational state.

3. Consider how the tagged modules are loaded during boot and updated during operation. Since modification of the module's code might compromise the corresponding assets, consider the authentication token of that code as an asset and assign it the same domain as the module itself.

   Since authentication tokens like digital signatures are self-protective, the relevant code of the loader might reduce to the code verifying that the image loaded is authentic. This includes any keys required to perform verification. This code may be active in non-operational phases, too. Furthermore, identify all firmware that manages the separation of process spaces. Tag these modules with the highest asset domain of the code authenticated or managed.

4. Repeat from Step 2 until no new modules come into scope and no tagging has been changed.

Whatever software remains untagged by this algorithm is in the Vendor domain—i.e., no firmware in the sense of the DTR. All other software parts are then assigned a proper firmware domain.

A description of this analysis should be in the software architecture delivered by the vendor. It should make clear on the implementation level where the assets are produced, transferred, stored, and destroyed, giving a focused design description. It should furthermore provide a rationale how the process spaces in Step 2 are segregated. It will describe the authentication data required for code management and naturally describe the boot stack of the system.

Tag Coherence Verification

Identify which hardware components execute the modules tagged during software tagging. Tag the hardware with the highest domain tagged to any module executed. During the analysis it may happen that some hardware components have a lower domain assigned by initial hardware tagging than by software tagging. Identify the relevant assets that showed up in software tagging but were invisible during hardware tagging. Then do hardware tagging for these assets.

If both hardware and software tagging have been performed correctly, the effective domain for each hardware component should be identical. As a result, the domain of each hardware component, down to PCB tracks and for each piece of code—and even what is considered code in the first place—should be clear to the developer and the testers.

A description of this analysis should be in the hardware architecture delivered by the vendor. It should make clear on implementation level where the assets are produced, transferred, stored, and destroyed, giving a focused design description.

The vendor shall provide a block diagram at domain level that clearly identifies how the domains interact with one another. The corresponding programming interfaces or, if applicable, hardware interfaces shall be uniquely identified and documented. The diagram shall clearly identify whether software is executed on the same hardware or on separate CPU, memory, etc.

Applications

An application in the sense of a PCI HSM device is code that is for any reason not considered firmware. The developer produces guidance on how such code shall be developed and installs procedures to verify compliance with this guidance as a precondition for signing the code. Application code must be authentic to be installed.

PCI HSM may allow applications in the Data domain and the Vendor domain. Applications therefore can never handle clear-text PIN or keys of the Key domain. The installation and use of applications for Cloud HSM Processing elements is not permitted.

Applications shall be segregated from other firmware, including firmware of the same domain and other applications at least from different issuers. This requires applications to run in unique process spaces. The tester shall verify that the separation of process spaces is effective for application sandboxing, which obviously exceeds the requirements for domain separation.

Asset Tagging Guidance

In this section, an incomplete list of typical assets and their respective domain is presented. Assets not listed here should be assigned similarly.

Table F-2: Asset Tagging Examples

Asset	Protection	Domain
PIN encryption key	Confidentiality / integrity	Key domain
Key decryption key	Confidentiality / integrity	Key domain
Code validation public key	Integrity	Key domain
PIN	Confidentiality	PIN domain
PAN	Confidentiality / integrity	Data domain
Mag-stripe data	Confidentiality / integrity	Data Domain
Authentication codes / passwords	Confidentiality / integrity	Data Domain
Secure channel cryptographic keys	Confidentiality / integrity	Data Domain

Structure and Contents

The developer shall provide an Asset Flow Analysis as evidence for the entire scoping of the PCI HSM testing. The asset flow must be complete and unambiguous. Verification of the asset flow is a critical task in the assessment. The following requirements describe the content of the analysis and test requirements for verification.

Identification of Interfaces

- The vendor shall list all interfaces accessible without tampering the device through normal access—e.g., casing removal. This explicitly includes interfaces that are not visible on the PCB but are available after removing any covers that do not signal a tamper event.
- The tester shall visually inspect the device to verify that all accessible interfaces have been listed.
- If interfaces connect to hardware subsystems that support multiple process spaces, the vendor shall clearly state which process space handles the low-level communication with the respective interface. Note that in most cases this will pertain to some kind of operating system.
- The tester shall consider any relevant documentation to verify these claims.
- The vendor shall describe the purpose of each interface listed. The tester shall reference and provide complete documentation describing all protocol layers. The documentation shall contain all commands, parameters, signal levels and their potential responses including all error messages/states.
- The tester shall check that the documentation is complete, comprehensive, and adequately detailed.

Identification of Components

- The vendor shall list all components—both hardware and software—used in his asset flow description. The vendor shall describe the boundaries of each component.
- The tester shall examine design documentation such as but not limited to schematics, architecture descriptions, and source code, to verify whether the boundaries of each component are clearly defined.
- The tester shall examine the design documentation to verify that components do not intersect—i.e., that no element from hardware or software design appears in more than one component.
- For hardware components, the tester shall verify that no boundary intersects a single chip. Exceptions for SoC with dedicated firewalling though uncommon may exist.
- For software components, the tester shall verify that no boundary intersects a process space.

Identification of Operations

- The vendor shall describe which kinds of transactions and other PCI-related communications or key management are supported by the device. Typical transactions are online PIN with MK/SK, online PIN with DUKPT, remote key loading, etc. Transactions that require a different key management shall be listed separately. "Other communication" refers to any other security-related communication by firmware or applications. At a minimum, it relates to firmware updates and key loading.

- The tester shall verify that all possible transactions for the device type and the claimed key-management schemes are listed.
- The tester shall verify that there are key-loading communications for each component storing keys.
- The tester shall verify that there is at least one operation to load any loadable key from the key table.
- The tester shall verify that there are firmware-loading communications for each component capable of software updates.
- If the device supports loadable prompts, the tester shall verify that there are mechanisms for loading prompts.
- If the device supports applications, the tester shall verify that there are mechanisms for loading applications.

Asset Flow Analysis

- For each operation, the vendor shall describe how any assets or asset containers enter or exit at the listed interfaces, and how they are conveyed between subsystems. The description shall clearly state where and when asset containers are unwrapped or assets are wrapped into containers. This may be illustrated by flow charts overlaid to block diagrams indicating assets by red arrows and asset containers in black. Note that wrapping or unwrapping requires further assets. The description shall clearly identify all assets.
- For hardware components that support multiple process spaces, the tester shall provide a logical view of how the data is conveyed between process spaces. Mechanisms changing software privilege level shall be detailed.
- The tester shall verify that all operations listed are described.
- For each operation, the tester shall verify that all required assets exist in a meaningful combination. For example, it is not possible to translate PINs if the required key exists only as an asset container.
- For each cryptographic key, the tester shall verify that the key is listed in the key table supplied by the vendor.
- The vendor shall assign a security domain for each hardware subsystem or interface and for each piece of code executed on the device.
- The tester shall perform asset tagging to verify the assigned security domains.
- If the device supports applications, the vendor shall describe how the applications are separated from firmware and from each other. If the applications may communicate with interfaces not considered firmware, these interfaces shall be described at the same level as those for firmware.
- The tester shall verify that the process spaces for applications are clearly defined.
- The tester shall verify that the process spaces for applications are distinct from those of firmware and other applications.
- The tester shall perform asset tagging on each application process space and verify that it is in the Data domain or Vendor domain.

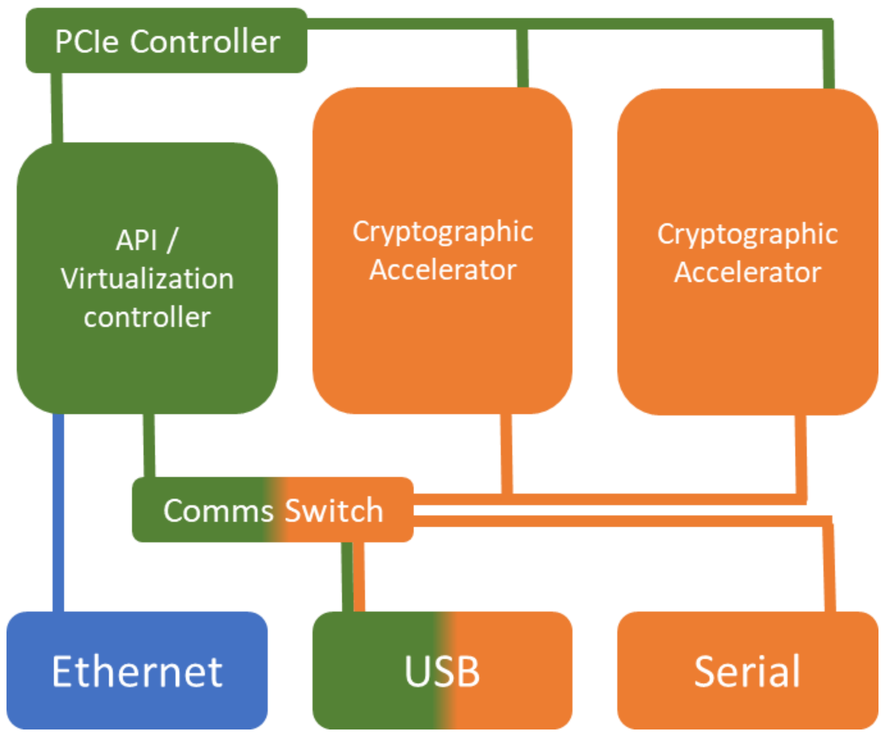
Scoping the Tests

- All hardware and software tagged as Vendor domain may be taken out of scope. As such, the boundary of the device is clearly defined. It may also be used as guidance for the security officer of the vendor to decide which modifications may clearly be covered by wildcards.
- The domain assigned to any subsystem defines the appropriate attack potential to the subsystem.
- The Asset Flow Analysis shall describe all cryptographic operations performed with relevant keys. This shall be used as an input to scope which side-channel attacks are feasible. Furthermore, it defines which components should be investigated for side channels.

Asset Flow Analysis Example

In this section we will provide a brief example of an asset flow analysis for an imaginary HSM. It should be noted that this is an example only and does not imply that this is the only possible configuration for an HSM, or even that this configuration would be found to be compliant to all requirements in all cases. All references to data flows and operation are for this example only and should not be construed as requirements or even suggestions as to how a compliant HSM may operate.

Figure F-1: Asset Flow Analysis Example


PCIe Controller
API / Virtualization controller
Cryptographic Accelerator
Cryptographic Accelerator
Comms Switch
Ethernet
USB
Serial

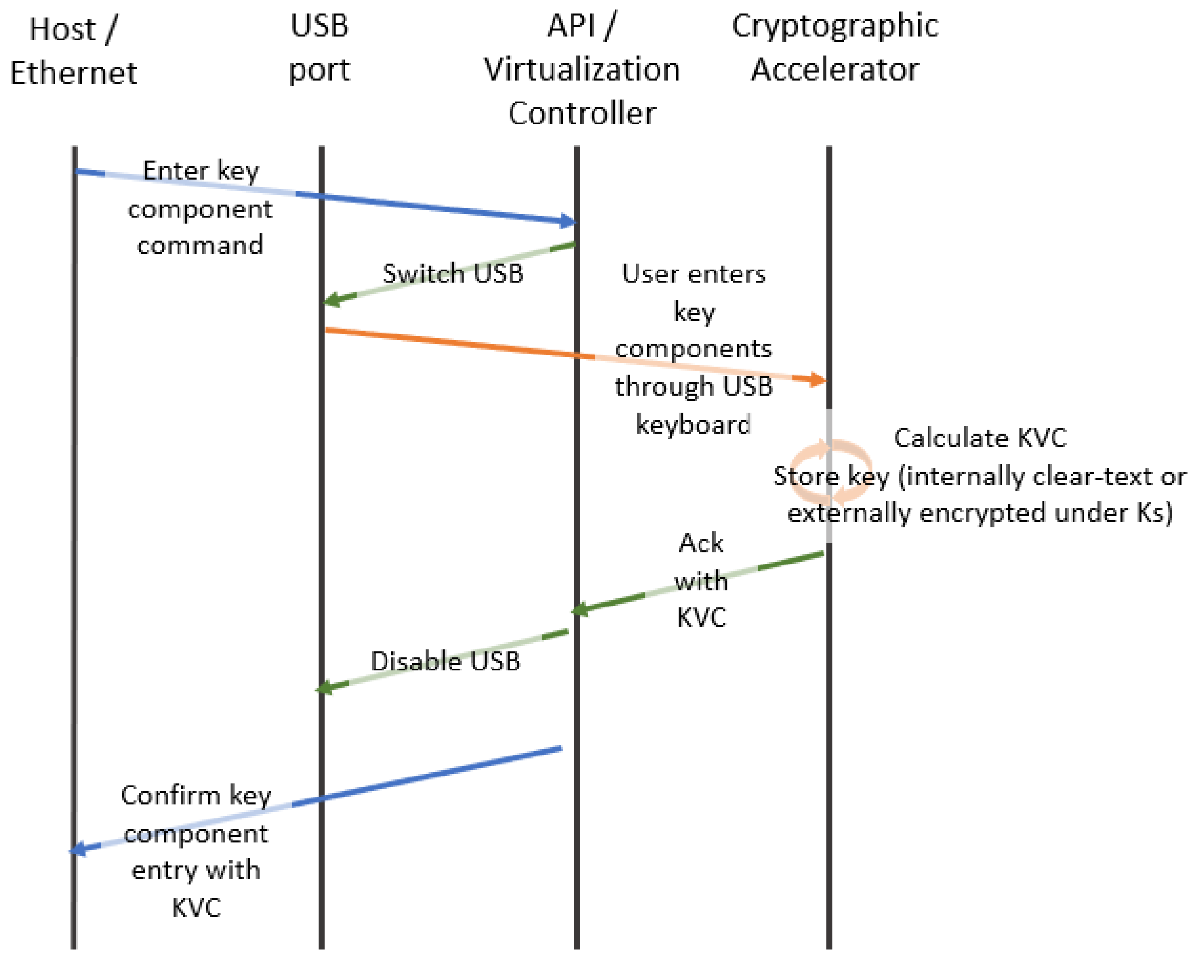
In the above diagram, the orange color represents the key domain, the green color is the data domain, and the blue color is the vendor domain.

The HSM has two main processing areas, the API/Virtualization controller and the two Cryptographic Accelerators. The API/Virtualization controller is used to provide the interface to the network which receives commands from the host/user(s). It terminates the secure channel used to transmit these commands and routes the commands to the cryptographic accelerators for processing using the PCIe connections. The API/Virtualization controller never has access to clear-text user or storage cryptographic keys. Console access to the API/Virtualization controller can be achieved remotely across the Ethernet using a mutually authenticated SSH connection, or locally through the USB interface.

The Cryptographic Accelerators store and process clear-text user cryptographic keys, as well as the HSM tamper keys used to secure the user keys. These keys are entered into the accelerators encrypted under another (already known) key or entered as key components through the serial or USB interfaces. Clear-text cryptographic keys and components can be output from the Cryptographic Accelerator(s) via the serial connection only (clear-text key output is provided on this system for key-loading purposes). A communications switch, under the control of the API/Virtualization controller, is able to route the USB or serial traffic to the Cryptographic Accelerators as required, with the USB interface also able to be switched to the API/Virtualization controller.

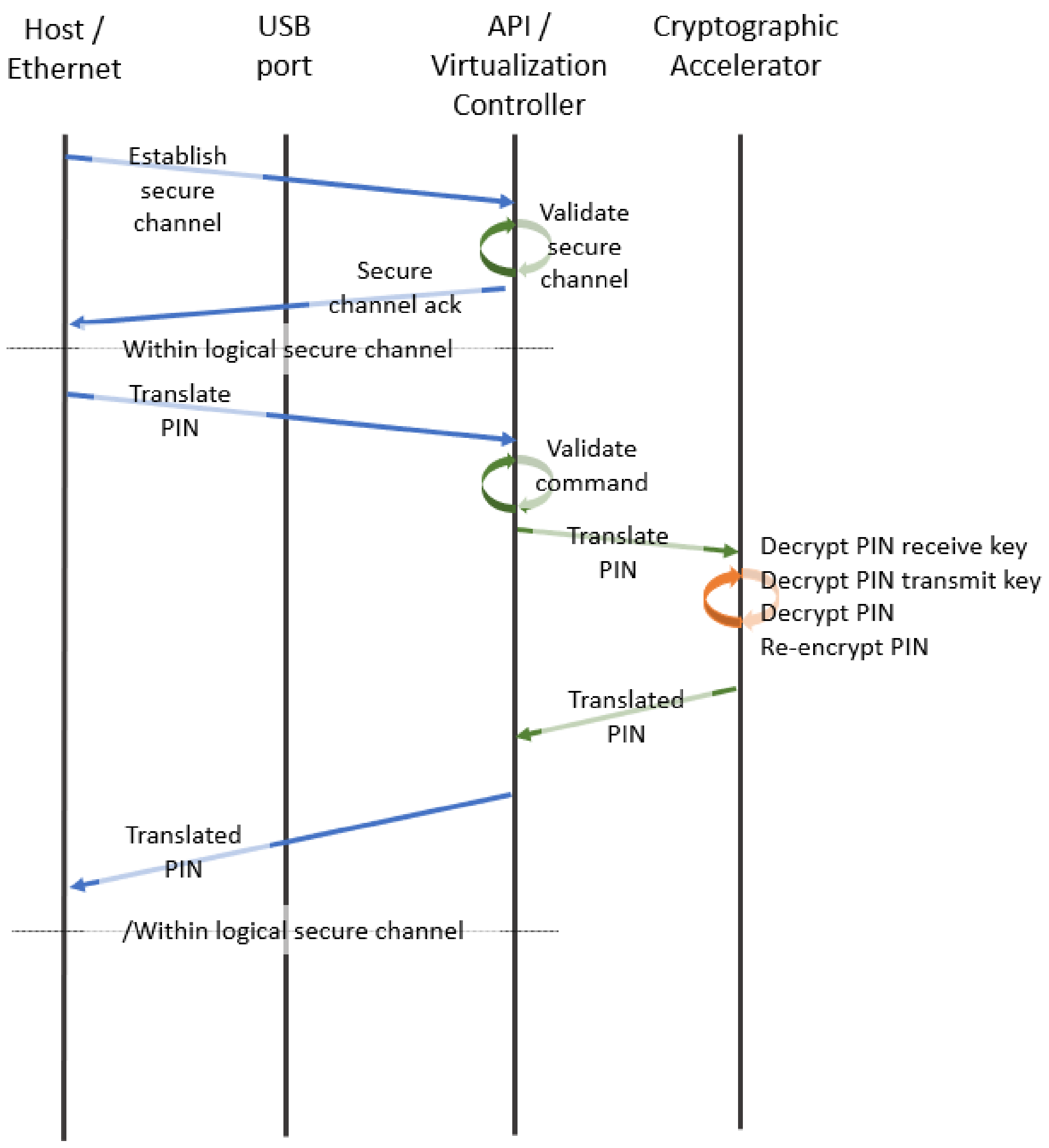
Sample data flows for key component entry and for cryptographic operations are provided below.

Figure F-2: Data Flow for Key-component Loading from Example HSM


Host / Ethernet
USB port
API / Virtualization Controller
Cryptographic Accelerator
Enter key component command
Switch USB
User enters key components through USB keyboard
Calculate KVC
Store key (internally clear-text or externally encrypted under Ks)
Ack with KVC
Disable USB
Confirm key component entry with KVC

.

Figure F-3: Data Flow for PIN-translation Command from Example HSM


Host / Ethernet
USB port
API / Virtualization Controller
Cryptographic Accelerator
Establish secure channel
Validate secure channel
Secure channel ack
Within logical secure channel
Translate PIN
Validate command
Translate PIN
Decrypt PIN receive key
Decrypt PIN transmit key
Decrypt PIN
Re-encrypt PIN
Translated PIN
Translated PIN
/Within logical secure channel