Retired Draft

Warning Notice

The attached draft document has been RETIRED. NIST has discontinued additional development of this document, which is provided here in its entirety for historical purposes.

Retired Date March 17, 2023

Original Release Date August 6, 2014

Retired Document

Status	Initial Public Draft (IPD)
Series/Number	NIST Special Publication 800-85B-4
Title	PIV Data Model Test Guidelines
Publication Date	August 2014
Additional Information	See NIST SP 800-85B and https://csrc.nist.gov/projects/piv for current information about NIST's Personal Identity Verification (PIV) project.

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Draft NIST Special Publication 800-85B-4

PIV Data Model Test Guidelines

Ramaswamy Chandramouli
Hildegard Ferraiolo
Ketan Mehta
Jason Mohler
Lam Ly
Gregory Hollenbaugh

C O M P U T E R S E C U R I T Y

National Institute of
Standards and Technology
U.S. Department of Commerce

Draft NIST Special Publication 800-85B-4

PIV Data Model Test Guidelines

Ramaswamy Chandramouli
Hildegard Ferraiolo
Ketan Mehta
Computer Security Division
Information Technology Laboratory

Jason Mohler
Lam Ly
Gregory Hollenbaugh
Electrosoft Services, Inc.

August 2014

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Acting Under Secretary of Commerce for Standards and Technology and Acting Director

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Management Act of 2002 (FISMA), 44 U.S.C. § 3541 *et seq.*, Public Law 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-85B-4
Natl. Inst. Stand. Technol. Spec. Publ. 800-85B-4, 248 pages (August 2014)
CODEN: NSPUE2

Public comment period: *August 06, 2014* through *September 05, 2014*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: piv_comments@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

FIPS201 describes a variety of data model components as a part of the PIV logical credentials. Such components include biometric elements in the form of fingerprint information and facial imagery and security elements such as electronic keys, certificates, and signatures. FIPS201 incorporates by reference NIST Special Publication 800-73-4 (SP80073), which specifies elements related to the PIV card interface, NIST Special Publication 800-76 (SP80076), which specifies the biometric requirements, and NIST Special Publication 800-78 (SP80078), which specifies acceptable cryptographic algorithms and key sizes for PIV systems.

A robust testing framework and guidelines to provide assurance that a particular component or system is compliant with FIPS201 and supporting standards should exist to build the necessary PIV infrastructure to support common unified processes and systems for government-wide use. NIST developed test guidelines in two parts. The first part addresses test requirements for the interface to the PIV card, which are provided in NSIST Special Publication 800-85A (SP80085A). The second part provides test requirements for the PIV data model and is provided in this document. This document specifies the derived test requirements, and the detailed test assertions and conformance tests for testing the PIV data model.

Keywords

BER-TLV testing; biometrics; certificate conformance test; FIPS 201; identity credential; implementation under test (IUT); PIV data model; Personal Identity Verification (PIV); smart cards

Acknowledgements

The authors (Ramaswamy Chandramouli, Hildegard Ferraiolo, Ketan Mehta of NIST, Jason Mohler Lam Ly and Gregory Hollenbaugh of Electrosoft Services Inc., and Frazier Evans of Booz Allen Hamilton, Inc.) wish to thank their colleagues who reviewed this document and contributed to its development. The authors also gratefully acknowledge and appreciate the many contributions from the public and private sectors whose thoughtful and constructive comments to of previous revisions of this document improved the quality and accuracy of this publication.

Executive Summary

Homeland Security Presidential Directive 12 (HSPD-12) called for a new standard to be adopted governing the use of common identity credentials for physical and logical access to Federal government locations and systems. The Personal Identity Verification (PIV) standard for Federal Employees and Contractors, Federal Information Processing Standard 201 (FIPS201), was developed to establish government-wide identity credentials. Credentials are issued to individuals whose true identity has been verified and whose need for the credential has been established and authorized by proper authorities.

FIPS201 describes a variety of data model components as a part of the PIV logical credentials. Such components include biometric elements in the form of fingerprint information, facial and iris imagery, and security elements for electronic keys, certificates. FIPS201 incorporates by reference NIST Special Publication 800-73 (SP80073), which specifies elements related to the PIV card interface, NIST Special Publication 800-76 (SP80076), which specifies the biometric requirements, and NIST Special Publication 800-78 (SP80078) which details acceptable cryptographic algorithms and key sizes for PIV systems.

A robust testing framework and guidance to provide assurance that a particular component or system is compliant with FIPS201 and supporting standards should exist to build the necessary PIV infrastructure to support common unified processes and systems for government-wide use. NIST developed test guidance in two parts. The first part addresses test requirements to interface with the PIV card and is provided in SP80085A. The second part provides test requirements for the PIV data model of the PIV card and is provided in this document. This document specifies the derived test requirements, and the detailed test assertions and conformance tests for testing the PIV card's data model.

ALIGNING REVISION NUMBERS

WHAT HAPPENED TO SPECIAL PUBLICATION 800-85B REVISIONS 2 AND 3?

Revision numbers between NIST Special Publications 800-73 and 800-85B were misaligned from the start because the initial publication of SP 800-85B did not occur until after the publication of SP 800-73, Revision 1. When SP 800-73, Revision 4 was published, SP 800-85B was updated to Revision 1 for consistency with the updates to SP 800-73. This revision number mismatch created ongoing uncertainty and confusion regarding which revision of SP 800-85B was consistent with which revision of SP 800-73. To reduce this uncertainty going forward, revision numbers 1, 2 and 3 have been skipped for SP 800-85B, and this version of SP 800-85B has been given revision number 4 (SP 800-85B-4) since this version is consistent with the updates to SP 800-73, Revision 4. Future revisions of SPs 800-73 and 800-85A will maintain the revision number consistency.

Table of Contents

List of Appendices

1. Introduction

1.1 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This recommendation has been prepared for use by Federal agencies. It may be used by non-governmental organizations on a voluntary basis and is not subject to copyright. Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should this recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of OMB, or any other Federal official.

1.2 Purpose and Scope

The Federal Information Processing Standard 201 (FIPS201) establishes a system for verifying an individual employee or contractor's identity in a reliable, secure, and interoperable manner across the Federal government. Credentials are issued to individuals whose true identity has been verified and whose need for the credential has been established and authorized by proper authorities. FIPS201 also describes a variety of authentication mechanisms, including the use of cryptographic mechanisms and biometric data belonging to cardholders.

In order to build the necessary Personal Identity Verification (PIV) infrastructure to support common unified processes and systems for government-wide use, there must be a robust testing framework to provide assurance that a particular component or system is compliant with FIPS201 and companion specifications. This test guideline document specifies the derived test requirements, detailed test assertions, and conformance tests for testing the data elements of the PIV system as per specifications laid out in FIPS201, SP80073, SP80076, and SP80078.

This document does not provide conformance tests for any other software used in the PIV system such as the back-end access control software, card issuance software, and specialized service provider software. Specifically, this document does not provide test requirements for the PIV card interface, FIPS 140-2 validation, key generation and certificate binding, cryptographic algorithms, biometric enrollment and verification processes[1], performance of biometric products, and non-PIV aspects of external biometric standards and profiles.

This document provides technical guidelines on the methodology to be used during testing applicable PIV cards, but does not provide normative guidelines on which entities will execute the tests. Also, the

[1] Testing of biometric processing performance using measures such as False Accept Rate (FAR) is described in Section 7 of SP80076.

test methodologies defined in this document are not designed to test business processes or to verify compliance with external applicable standards. For example, this document does not provide test guidelines to determine how good a user's Personal Identification Number (PIN) choice is or how access rights are granted to employees.

1.3 Audience and Assumptions

This document is targeted at vendors and integrators of card management systems as well as the entities that will conduct tests on such systems. Readers are assumed to have a working knowledge of FIPS201, PIV guidelines, and applicable technologies.

This document will:

Enable developers of card management systems to design PIV card personalization modules to be testable for requirements specified in FIPS201, SP80073, SP80076, and SP80078.

Enable developers of these systems to develop self-tests as part of the development effort.

Enable testers to develop tests that cover the test suite provided in this document.

2. Conformance Test Overview

The conformance testing guidelines in this document applies to the testing of the PIV card's data model. The data model requirements are extracted from FIPS201, SP80073, SP80076, and SP80078. This overview section provides a high-level conformance test architecture for testing the PIV card's data model. The conformance test architecture is confined to a personalized PIV card. In other words, the conformance test approach views the card issuance system as a "black box," meaning that the interface of that system is opaque and its implementation details are not relevant to the testing. The PIV data model testing operates under the assumption that the PIV card being tested has already been personalized as described in Sections 2.8 and 2.9 of FIPS201. The following sections provide the details of data model testing, test architecture, test methodology, and test areas.

2.1 Test Architecture

The conceptual architecture for data model testing is shown in Figure 1. The conformance test in this document applies to the area highlighted with dashed lines. SP80085A addresses the writing and extracting of data from the card and subsequently, those processes are not addressed in this document.

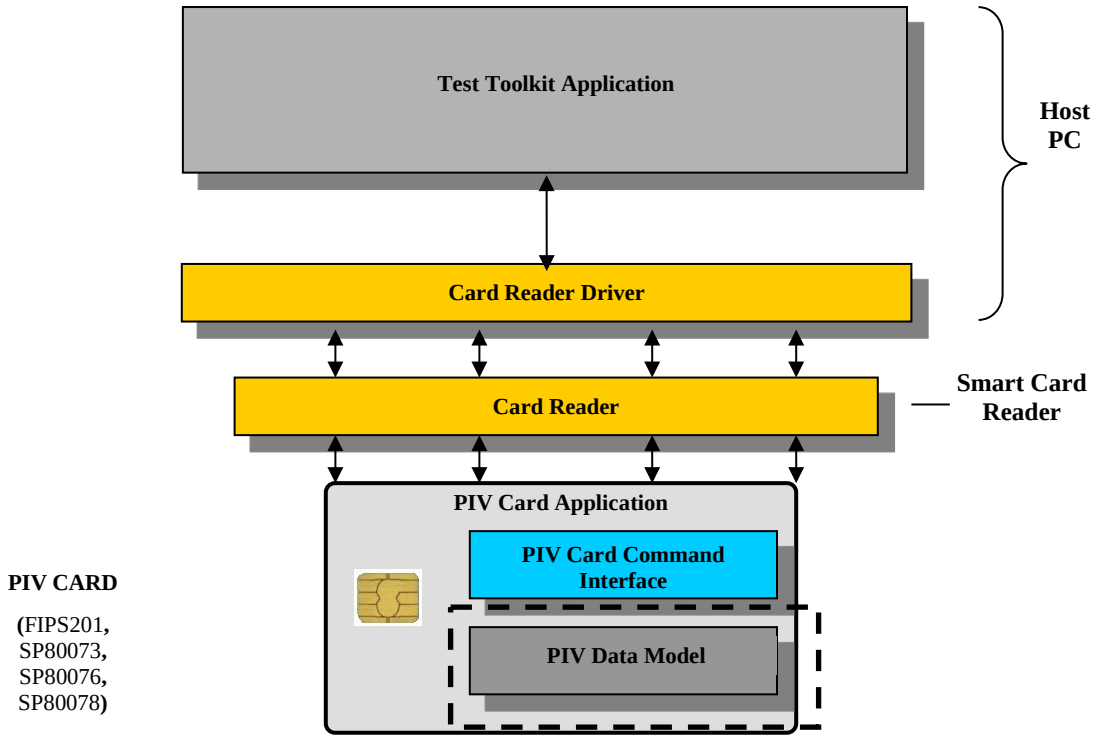

Figure 1: PIV Conformance Test Architecture

The PIV data model defines the logical use of the on-card application space including the SP80073 Part 1 required data objects and data elements along with the size and structure of each object.

The PIV data model test includes the testing of the following aspects of a PIV card's data:

- Basic Encoding Rules Tag-Length-Value (BER-TLV) Format Conformance as per Appendix A of SP80073 Part 1 for all objects.
- Conformance of the Signature Block to Cryptographic Message Syntax (CMS) Signature format for all signed objects.
- Conformance of the two mandatory biometric objects (Card Holder Fingerprints and Facial Image), and the optional biometric iris object to the Common Biometric Exchange Formats Framework (CBEFF) Header format as well as to ANSI/INCITS 378, ANSI/INCITS 385, and ISO/IEC 19794-6 profiles respectively. The OCC BIT Group Template shall conform to Table 7 of SP80076.
- Conformance to Federal Public Key Infrastructure Policy Authority (FPKIPA) profiles for all Public Key Infrastructure (PKI) Certificates.

2.2 Test Methodology

The data model testing was developed through the following two-step process:

- **Create Derived Test Requirements (DTRs)** — These are constructed from the data format and content requirements in FIPS201, SP80073, SP80076, and SP80078 specifications.
- **Develop test assertions** — These provide the tests that need to be performed to test each of the DTRs. The test assertions will include testing of data formats, values in the individual fields, relationship among values in multiple fields and validate the computations. Also, the test assertions include testing of the optional fields when they are present.

Figure 2 depicts the test methodology adopted to provide complete guidelines for testing PIV conformant products. SP80085A provides the DTRs and test assertions for the interfaces to the PIV smart card and the PIV middleware. This document provides DTRs and test assertions for the identity credentials stored on the PIV card.

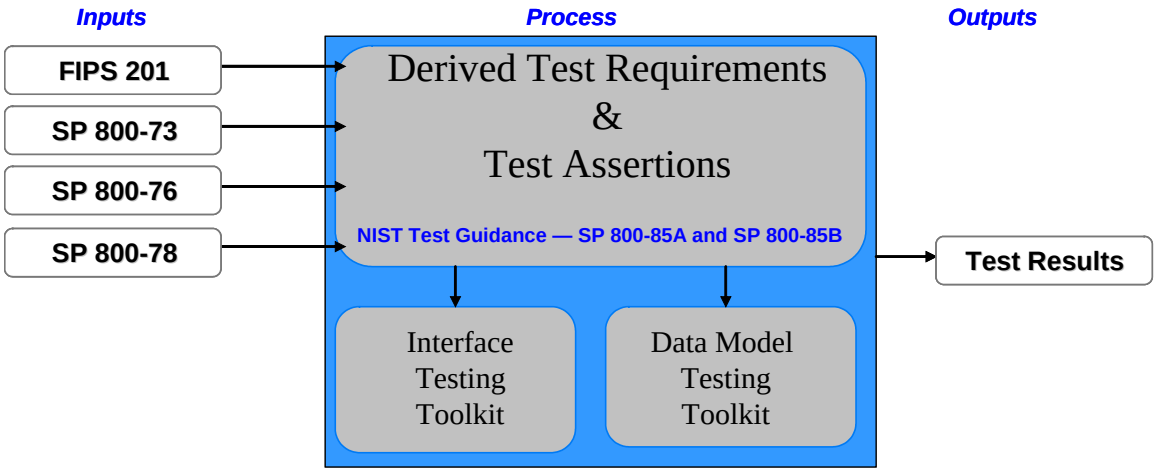

Figure 2: PIV Test Methodology

2.3 Test Set-up

The test system consists of the following components:

- A test toolkit application software that resides on a personal computer.
- An ISO7816 and Personal Computer / Smart Card (PC/SC) compliant contact-based smart card reader.
- A mechanism to input the PIN that can be transmitted to the smart card reader. Examples of such mechanisms are a PIN pad or a keyboard.
- A set of test personalized PIV cards whose applications and interfaces are compliant with SP80073. All personalized biometric information is assumed to be collected and processed by template generation and matching implementations that have been tested against minimum performance qualification criteria established by NIST, OMB, and by Federal agencies, as appropriate.

2.4 Test Areas

The test assertions in this document will validate that all PIV data objects conform to their respective requirements. Conformance criteria include correct formatting and, when appropriate, context specific content. Additionally, conformance will be based on correct computation of content such as digital signatures. The DTRs and test assertions are designed to validate each PIV data object such that the following three statements are true for the objects:

- PIV containers are formatted correctly,
- Field values are in accordance with the specifications, and
- Data consistency and value computations such as signatures are accurate.

Again, these requirements are not designed to test business processes or to verify external compliance with applicable standards. For further clarification of the document scope, refer to Section 1.2, Purpose and Scope.

2.4.1 BER-TLV Format Conformance

The tags and lengths in various data objects shall conform to specifications in Appendix A of SP80073 Part 1. In addition some data objects may be tested for the valid values of data elements.

2.4.2 Biometric Data Objects Conformance

The two mandatory biometric objects, (Card Holder Fingerprints and Facial Image), and the optional biometric iris object shall conform to the common CBEFF Header format as well as to ANSI/INCITS 378, ANSI/INCITS 385, and ISO/IEC 19794-6 profiles respectively. The OCC BIT Group Template shall conform to Table 7 of SP80076.

2.4.3 Digital Signature Blocks Conformance

For all signed objects the fields in the signature block shall conform to the CMS syntax specified in SP80073. In addition some data objects may be tested for the valid values of data elements.

2.4.4 PKI Certificate Profile Conformance

The mandatory PIV Authentication Certificate, Digital Signature, Key Management, Card Authentication Certificate, and the Content Signing Certificate(s) shall conform to the certificate profiles as specified in the X.509 Certificate and Certificate Revocation List (CRL) Extensions Profile for the Shared Service Providers (SSP) Program (X509 Extensions).

3. Test Methodology

3.1 Derived Test Requirements

DTRs show the type of tests required based on the normative specifications in FIPS201 and supporting special publications. Each DTR consists of the following:

Actual statements taken/derived from the specification — these include explicit statements using the words "shall," "must," and other normative deliminators in the standard. The condition statements are identified by codes starting with 'AS' followed by a running sequence.

Required Vendor Information — these include information that card vendors, card management system vendors agencies or integrators are mandated to provide in their documentation. The Required Vendor Information is identified by codes starting with 'VE' followed by a running sequence.

Required Test Procedures — these are actions that the tester has to perform in order to satisfy the requirements stated in actual condition statements. These include verifying the information mandated in the "Required Vendor Information" for the condition as well as performing software-based tests. Some of the required test procedures do not call explicitly for verification of information in the associated "Required Vendor Information." In these instances it is implicitly assumed that such information is provided by the vendor and verified by the tester. The Required Test Procedures are identified by codes starting with 'TE' followed by a running sequence that denotes the section in this document where they occur.

Validations of some DTRs are not covered by the test assertions provided in this document. These DTRs require compliance of a component with an external specification or standard such as the Electronic Biometric Transmission Specification. No required test procedures are provided for these DTRs, and a note is added to indicate that "this assertion is externally tested." The tester is required to check the vendor documentation for claimed compliance with such requirements or confirm the presence of an external test/compliance certificate obtained from the test organization, when applicable.

In some instances, testing of DTRs may not be feasible using the test methodologies described in this document. For example, a test tool built on these methods cannot test the procedure by which fingerprints are taken at an agency PIV installation. Most of these DTRs, however, can be tested by inspection of the system description document. Where this is the case, an adequate description is generally required by the VE section of the DTR. Where testing is not feasible, a note is added to indicate that "this assertion is not separately tested."

3.2 Test Assertions

Test assertions are statements of behavior, action, or condition that can be measured or tested. They provide the procedures to guide the tester in executing and managing the test. They include purpose of the test, starting conditions and prerequisites, success criteria, and post-test conditions, when applicable.

The following four sets of test assertions are included in this document —

- BER-TLV (Section 8)
- Biometric data object (Section 9)
- Signed data element (Section 10)
- PKI certificate profile (Section 11)

All the test assertions provided in this document come under the PIV card's data model testing and are based on DTRs in Sections 4, 5, 6, and 7. Specifically, each test assertion makes specific references to the related DTR sections. Overall there is a many-to-many relationship from the test assertions to the DTRs (i.e., one test can map to many DTRs and one DTR can map to many tests). To narrow the search space for cross references, Table 3-1 presents a cross-referencing guide showing the relevant DTR sections and test assertion sections with respect to tests.

Category/Classes of Test	DTR Section(s)	Test Assertion Section(s)
(1) BER-TLV	Section 4 (Derived from SP80073 Part 1)	Section 8
(2) Biometric Data	Section 5 (Derived from SP80076)	Section 9
(3) Signed Data Elements	Section 6 (Derived from FIPS201 and SP80078)	Section 10
(4) PKI Certificate Profile	Section 7 (Derived from FIPS201, SP80073, and SP80078)	Section 11

Table 3-1. Cross-referencing Guide

4. BER-TLV DTRs

4.1 BER-TLV Testing

AS04.01.01: Conformant cards shall return all the Tag-Length-Value (TLV) elements of a container in the physical order listed for that container in this data model.

VE04.01.01.01: The vendor shall specify in its documentation the format (TLV) and the content of all the elements in each data container on the card.

VE04.01.01.02: The vendor shall specify in its documentation that the information provided conforms to SP80073 Part 1.

TE04.01.01.01: The tester shall validate that the formatting, encoding and the content of all the elements in each data container conforms to SP80073 Part 1.

4.2 Card Capability Container (CCC)

AS04.02.01: The data model of the PIV card Application shall be identified by data model number 0x10.

VE04.02.01.01: The vendor shall specify in its documentation the tags and associated values in the CCC container.

VE04.02.01.02: The vendor shall specify presence of the optional fields and highlight if any of the deprecated data fields (Extended Application CardURL and Security Object Buffer data) are present. Vendor shall also specify if 1) all mandatory data elements of the CCC, except for the data model number, have a length value set to zero bytes (i.e., no value field will be supplied) and 2) if optional data elements are present or absent.

TE04.02.01.01: The tester shall validate the format and the content of all the elements in CCC data container on the card. Data read from the card will be validated against the vendor provided data.

TE04.02.01.02: The tester shall validate that the registered data model value is `0x10`.

4.3 Card Holder Unique Identifier (CHUID)

AS04.03.01: The CHUID on a PIV card shall meet the following requirements:

The Federal Agency Smart Credential Number (FASC-N) shall be in accordance with Technical Implementation Guidance: Smart Card Enabled Physical Access Control Systems (TIG SCEPACS). The Agency Code, System Code, and Credential Number of the FASC-N shall be present. The credential series, individual credential issue, person identifier, organizational category, organizational identifier, and person/organization association category of the FASC-N shall be populated or contain all zeros.

A subset of FASC-N, the FASC-N Identifier, shall be the unique identifier as described in [TIG SCEPACS, 6.6]: “The combination of an Agency Code, System Code, and Credential Number is a fully qualified number that is uniquely assigned to a single individual”.

The Global Unique Identifier (GUID) field must be present, and shall include a Card Universally Unique Identifier (UUID). The value of the GUID data element of the CHUID data object shall be a 16-byte binary representation of a valid UUID [RFC4122]. The UUID should be version 1, 4, or 5, as specified in [RFC4122, Section 4.1.3]. The previously deprecated Authentication Key Map data element shall not be present in the CHUID.

The optional Cardholder UUID field shall map to RFU tag 0x36. When present, the Cardholder UUID shall be a 16-byte binary representation of a valid UUID, and it shall be version 1, 4, or 5, as specified in [RFC4122, Section 4.1.3].

The Expiration Date shall be mapped to the reserved for future use (RFU) tag `0x35`, keeping that within the existing scope of the TIG SCEPACS specification. This field shall be 8 bytes in length and shall be encoded in ASCII as YYYYMMDD. The expiration date shall be the same as printed on the card.

VE04.03.01.01: The vendor shall specify in its documentation the format (TLV) and the content of all the elements in CHUID container on the card.

VE04.03.01.02: The vendor shall specify presence of the optional fields and highlight any of the deprecated data fields (Buffer Length, DUNS and Organization Identifier) that are present. The vendor shall also indicate that the retired key map field is absent.

TE04.03.01.01: The tester shall validate the format and the content of all the elements in CHUID data container on the card.

4.4 Off-Card Comparison Biometric Fingerprint

AS04.04.01: The fingerprint data object on a PIV card is preceded with the tag value 0xBC and the FASC-N shall be present in the CBEFF header as well as in the CBEFF signature block. The Card UUID shall be present in the CBEFF signature block.

There are no vendor requirements.

TE04.04.01.01: The tester shall validate that the fingerprint data follows the tag value `0xBC` within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.

4.5 Biometric Facial Image

AS04.05.01: The facial image on a PIV card is preceded with the tag value 0xBC and the FASC-N shall be present in the CBEFF header as well as in the CBEFF signature block. The Card UUID shall be present in the CBEFF signature block.

There are no vendor requirements.

TE04.05.01.01: The tester shall validate that the facial image follows the tag value `0xBC` within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.

4.6 Security Object

AS04.06.01: The message digest produced as a result of a hash function on the contents of a data object shall be identical to that data object's message digest contained in the security object.

There are no vendor requirements.

TE04.06.01.01: The tester shall validate that all unsigned data objects, such as the Printed Information data object, are included in the Security Object if present and that the message digests for the various data objects present in the security object are identical to the message digest of the data object itself.

4.7 Biometric Iris

AS04.07.01: The iris image on a PIV card is preceded with the tag value 0xBC and the FASC-N shall be present in the CBEFF header as well as in the CBEFF signature block. The Card UUID shall be present in the CBEFF signature block.

VE04.07.01.01: The vendor shall specify if the iris image is stored on the card.

TE04.07.01.01: The tester shall validate that the iris image follows the tag value `0xBC` within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.

4.8 Key History Object

AS04.08.01: The Key History Object shall meet the following requirements:

The Key History object shall be present in the PIV Card Application if the PIV Card Application contains any retired key management private keys, but may be present even if no such keys are present in the PIV Card Application.

The Key History object includes two mandatory fields, keysWithOnCardCerts and keysWithOffCardCerts, and one optional field, offCardCertURL.

The offCardCertURL field shall be present if the keysWithOffCardCerts value is greater than zero and shall be absent if the values of both keysWithOnCardCerts and keysWithOffCardCerts are zero. The offCardCertURL field may be present if the keysWithOffCardCerts value is zero but the keysWithOnCardCerts value is greater than zero.

VE04.08.01.01: The vendor shall specify presence of the optional fields.

TE04.08.01.01: The tester shall validate the format and the contents of all the elements in Key History data container on the card.

4.9 Discovery Object

AS04.09.01: The Discovery Object shall meet the following requirements:

PIV Card Applications that implement the pairing code shall implement the Discovery Object with the first byte of the PIN Usage Policy set to 0x50, 0x58, 0x70, or 0x78.

PIV Card Applications for which both the PIV Card Application PIN and the Global PIN is implemented shall have a Discovery Object with the PIN Usage Policy set to 0x60 zz, 0x68 zz, 0x70 zz, or 0x78 zz where zz is either 0x10 or 0x20.

PIV Card Applications for which OCC is implemented shall have a Discovery Object with the first byte of the PIN Usage Policy set to 0x48, 0x58, 0x68, or 0x78.

If the first byte is set to 0x40, 0x48, 0x50, or 0x58, then the second byte is RFU and shall be set to 0x00.

VE04.09.01.01: The vendor shall specify the card's PIN usage policy (Global PIN, PIV Card Application PIN, Pairing Code, and OCC) in its vendor documentation.

TE04.09.01.01: The tester shall validate that both tag 0x4F (PIV Card Application AID) and tag 0x5F2F (PIN Usage Policy) data elements are present in the Discovery Object. The tester shall validate the format and the contents of these data elements in the Discovery object data container on the card.

4.10 Biometric Information Templates (BIT) Group Template Object

AS04.10.01: The BIT Group Template Object shall meet the following requirements:

VE04.10.01.01: The Vendor shall specify the presence of optional fields.

TE04.10.01.01: The tester shall ensure that encoding of the BIT group template is in accordance with Table 7 of SP80076.

AS04.10.02: The BIT Group Template Object shall meet the following requirements:

When OCC satisfies the PIV ACRs for PIV data object's access and command execution, both the Discovery Object and the BIT Group Template data object shall be present, and bit 4 of the first byte of the PIN Usage Policy shall be set.

There are no vendor requirements.

TE04.10.02.01: When the BIT Group Template is present, the tester shall ensure that bit 4 of the first byte of the PIN Usage Policy is set.

4.11 Secure Messaging Certificate Signer Object

AS04.11.01: The Secure Messaging Certificate Signer Object shall meet the following requirements:

The Secure Messaging Certificate Signer data object, which shall be present if the PIV card supports secure messaging for non-card-management operations, contains the certificate(s) needed to verify the signature of the secure messaging card verifiable certificate (CVC), as specified in 800-73-4 Part 2, Section 4.1.5.

VE04.11.01.01: The vendor shall specify presence of the optional Intermediate CVC.

TE04.11.01.01: If PIV card secure messaging for non-card-management operations is enabled, the tester shall ensure that the Secure Messaging Certificate Signer data object is present and

contains the certificate (and Intermediate CVC, if applicable) needed to verify the signature on secure messaging CVC.

4.12 Pairing Code Reference Data Container Object

AS04.12.01: The Pairing Code Reference Data Container Object shall meet the following requirements:

The Pairing Code Reference Data Container shall be present if the PIV card supports the virtual contact interface (VCI) with the pairing mechanism. This object includes a copy of the PIV card's pairing code.

There are no vendor requirements.

TE04.12.01.01: If the PIV card supports VCI with pairing, then the tester shall ensure that the Pairing Code Reference Data Container is present and includes a copy of the PIV card's pairing code.

4.13 Unused Data Objects on the PIV Card

AS04.13.01: Data containers that are created but not used shall be set to zero-length value.

VE04.13.01.01: Vendor shall state in its documentation the unused data containers on the PIV card.

TE04.13.01.01: The tester shall confirm that all unused data containers on the card are set to a zero length by performing GET DATA on the unused data objects and ensure that the length of the returned objects is equal to zero (i.e., the value field is absent).

5. Biometric Data DTRs

5.1 Common Header for PIV Biometric Data

The assertions in this section apply to the fingerprint template, facial image, and iris image stored on the PIV card. The iris image is an optional element on the PIV card and must be tested if present.

AS05.01.01: The CBEFF structure must comply with SP80076 Table 13, "CBEFF concatenation structure."

VE05.01.01.01: The vendor shall specify the optional biometrics stored on the PIV card.

TE05.01.01.01: The tester shall verify that the CBEFF structure is implemented in accordance with Table 13 of SP80076.

AS05.01.02: The CBEFF header must comply with SP80076 Table 14, "Patron format PIV specification."

No requirements for vendor.

TE05.01.02.01: The tester shall verify the length of the Patron Format header.

TE05.01.02.02: The tester shall verify the values are consistent with Table 14 "Patron format PIV specification" requirements of SP80076.

AS05.01.03: Multi-byte integers in the CBEFF headers shall be in big-endian byte order.

VE05.01.03.01: The vendor shall document the values of the CBEFF header fields.

TE05.01.03.01: The tester shall compare value provided against the stored data.

AS05.01.04: The Patron Header Version of the CBEFF Patron Format shall be 0x03.

No requirements for vendor.

TE05.01.04.01: The tester shall verify that the Patron Header Version value is 0x03.

AS05.01.05: The biometric data block is digitally signed but not encrypted, and this shall be reflected by setting the value of the Signature Block Header (SBH) security options field to b00001101.

No requirements for vendor.

TE05.01.05.01: The tester shall verify that the SBH security option value is b00001101.

AS05.01.06: For fingerprint and facial records, the Biometric Data Block (BDB) Format Owner shall be 0x001B denoting M1, the INCITS Technical Committee on Biometrics and for iris images, the BDB Format Owner shall be 0x0101 denoting ISO/IEC JTC 1/SC37 Biometrics.

No requirements for vendor.

TE05.01.06.01: The tester shall verify that the BDB Format Owner field contains 0x001B for fingerprint and facials records and contains 0x0101 for iris images.

AS05.01.07: For the mandatory fingerprint template on the PIV card, the BDB Format Type value shall be 0x0201. For the mandatory facial image on the PIV card, the BDB Format Type value shall be 0x0501. For the optional iris image on the PIV card the BDB Format Type value shall be 0x0009.

No requirements for vendor.

TE05.01.07.01: The tester shall verify that the BDB Format Type field is 0x0201 for fingerprint template, 0x0501 for facial image, and 0x0009 for the optional iris image if present.

AS05.01.08: The Creation Date in the PIV Patron Format (see Row 7 in Table 14 of SP80076) shall be the date of acquisition of the parent sample, encoded in eight bytes using a binary representation of "YYYYMMDDhhmmssZ". Each pair of characters (for example, "DD") is coded in 8 bits as an unsigned integer where the last byte is the binary representation of the ASCII character Z which is included to indicate that the time is represented in Coordinated Universal Time (UTC). The field "hh" shall code a 24 hour clock value.

No requirements for vendor.

TE05.01.08.01: The tester shall verify the date field is in compliance with the assertion.

AS05.01.09: The Validity Period in the PIV Patron Format (Row 8 in Table 14 of SP80076) contains two dates.

No requirements for vendor.

TE05.01.09.01: The tester shall verify that the headers contain two dates in compliance with the assertion.

AS05.01.10: Biometric Type field within the PIV Patron Format shall be 0x000008 for the fingerprint templates, 0x000002 for facial images, and 0x000010 for the iris images. The value for other biometric modalities shall be that given in CBEFF, 5.2.1.5. For modalities not listed there the value shall be 0x0.

No requirements for vendor.

TE05.01.10.01: The tester shall verify that the Biometric Type field contains 0x000008 for fingerprint templates, 0x000002 for facial images, and 0x000010 for the iris images.

AS05.01.11: The Biometric Data Type field within the PIV Patron Format shall be b100xxxxx for the fingerprint templates, b001xxxxx for the facial images, and b010xxxxx for the iris images.

No requirements for vendor.

TE05.01.11.01: The tester shall verify that the Biometric Data Type field within the PIV Patron Format is b100xxxxx for the fingerprint templates, b001xxxxx for the facial images, and b010xxxxx for the iris images.

AS05.01.12: For all biometric data stored on a PIV Card the quality value shall be a signed integer between -2 and 100 per the text of INCITS 358. A value of -2 shall denote that assignment was not supported by the implementation; a value of -1 shall indicate that an attempt to compute a quality value failed. Values from 0 to 100 shall indicate an increased expectation that the sample will ultimately lead to a successful match. The zero value required by FACESTD shall be coded in this CBEFF field as -2.

VE05.01.12.01: The vendor shall provide the quality values for the biometric data.

TE05.01.12.01: The tester shall verify that the value of Biometric Data Quality is between -2 and 100 for the fingerprint templates, facial images, and iris images.

AS05.01.13: The Creator field in the PIV Patron Format contains 18 bytes of which the first K ≤ 17 bytes shall be printable ASCII characters, and the first of the remaining 18-K shall be a null terminator (zero).

VE05.01.13.01: The vendor shall provide the value of Creator field.

TE05.01.13.01: The tester shall verify that the Creator field in the PIV Patron Format contains 18 bytes of which the first K ≤ 17 bytes is printable ASCII characters, and the first of the remaining 18-K is a null terminator (zero).

AS05.01.14: The FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier.

VE05.01.14.01: The vendor shall provide the value for FASC-N.

TE05.01.14.01: The tester shall verify that the FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier.

Note: This field may be filled with zeroes in the one exceptional case where PIV registration images are being stored before a FASC-N has been assigned. In such instances, the digital signature shall be regenerated once the FASC-N is known.

AS05.01.15: The "Reserved for future use" field in the PIV Patron Format shall contain 0x00000000.

No requirement for vendor.

TE05.01.15.01: The tester shall verify the "Reserved for future use" field is 0x00000000.

5.2 Off-Card Comparison Fingerprint Template Stored on PIV Card

AS05.02.01: Both fingers' template records shall be wrapped in a single CBEFF structure prior to storage on the PIV card.

VE05.02.01.01: The vendor shall specify in its documentation the CBEFF structure is constructed in accordance with this assertion.

TE05.02.01.01: The tester shall parse the biometric data container to verify this assertion.

Note: The CBEFF structure itself is tested in later assertions.

AS05.02.02: The fingerprint templates stored on the card are compliant to the MINUSTD profile specified in SP80076, Table 6, INCITS 378 profile for PIV card templates.

VE05.02.02.01: The vendor shall specify in its documentation that the template generator generates templates in accordance with MINUSTD.

TE05.02.02.01: The tester shall verify that the resultant template is in compliance with the assertion.

AS05.02.03: The Format Identifier of the General Header Record shall be 0x464D5200.

No requirements for vendor.

TE05.02.03.01: The tester shall verify that the Format Identifier value is 0x464D5200.

AS05.02.04: The Version Number of the General Record Header shall be 0x20323000.

No requirements for vendor.

TE05.02.04.01: The tester shall verify that the Version Number is 0x20323000.

AS05.02.05: The Record Length of the General Record Header shall be 26≤L≤1574.

VE05.02.01 The vendor shall specify the length of the entire container which includes CBEFF wrapped record.

TE05.02.05.01: The tester shall verify that the Record Length of the General Record Header is $26 \leq L \leq 1574$.

AS05.02.06: Both of the two fields ("Owner" and "Type") of the CBEFF Product Identifier shall be > 0 (non-zero).

VE05.02.06.01: The vendor shall provide the Owner and Type of CBEFF product identifier.

TE05.02.06.01: The tester shall verify that the both of the two fields ("Owner" and "Type") of the CBEFF Product Identifier are > 0 (non-zero).

AS05.02.07: The Capture Equipment Compliance of the General Record Header shall be 1000b.

No requirements for vendor

TE05.02.07.01: The tester shall verify the Capture Equipment Compliance value is 1000b.

AS05.02.08: The Capture Equipment ID of the General Record Header is > 0.

VE05.02.08.01: The vendor shall specify the Capture Equipment ID value.

TE05.02.08.01: The tester shall verify the Capture Equipment ID of the General Record Header is > 0.

AS05.02.09: The width on Size of Scanned Image in X Direction shall be the larger of the widths of the two input images. Similarly, the height on Size of Scanned Image in Y Direction shall be the larger of the heights of the two input images.

VE05.02.09.01: The vendor shall report width and height of the images whose fingerprint templates are stored on the card.

TE05.02.09.01: The tester shall verify that the width on Size of Scanned Image in X Direction is the larger of the widths of the two input images and the height on Size of Scanned Image in Y Direction is the larger of the heights of the two input images.

AS05.02.10: The X (horizontal) and Y (vertical) resolutions shall be 197.

No requirements for vendor

TE05.02.10.01: The tester shall verify that the X (horizontal) and Y (vertical) resolutions are 197.

AS05.02.11: The Number of Views of the General Header Record shall be 2.

No requirements for vendor

TE05.02.11.01: The tester shall verify the Number of Views value is 2.

AS05.02.12: The Reserved Byte of the General Header Record shall be 0.

No requirements for vendor

TE05.02.12.01: The tester shall verify the Reserved Byte value is 0.

AS05.02.13: The View Number of the Single Finger View Record shall be 0.

No requirements for vendor

TE05.02.13.01: The tester shall verify the View Number value of the Single Finger View Record is 0.

AS05.02.14: The Impression Type of the Single Finger View Record shall be either 0 or 2.

VE05.02.14.01: The vendor shall specify if the live or non-live scan images were used.

TE05.02.14.01: The tester shall verify the value is either 0 or 2 and is consistent with vendor reporting.

AS05.02.15: The quality value of captured fingerprint images shall be 20, 40, 60, 80, 100, 254, or 255.

VE05.02.15.01: The vendor shall specify the procedure used to calculate the quality value. The vendor shall also specify the value in the Finger Quality field.

TE05.02.15.01: The tester shall verify that the quality value of captured fingerprint images shall be 20, 40, 60, 80, 100, 254, or 255.

Note: A value of "255" shall be assigned when fingerprints are temporarily unusable for matching. A value of "254" shall be assigned when the fingerprints are permanently unusable.

AS05.02.16: The Number of Minutiae of Single Finger View Record is between 0 and 128.

No requirements for vendor.

TE05.02.16.01: The tester shall verify the Number of Minutiae is between 0 and 128.

AS05.02.17: Fingerprint templates shall be limited to minutiae of types "ridge ending" and "ridge bifurcation" unless it is not possible to reliably distinguish between a ridge ending and a bifurcation, in which case the category of "other" shall be assigned and encoded as 00b.

No requirements for vendor.

TE05.02.17.01: The tester shall verify that the Minutiae Type is 00b, 01b, or 10b.

AS05.02.18: All coordinates and angles for fingerprint minutiae shall be recorded with respect to the original finger image. They shall not be recorded with respect to any image processing sub-image(s) created during the template creation process.

VE05.02.18.01: The vendor shall specify in its documentation that the template generator generates templates in accordance with this assertion.

Note: This assertion is externally tested.

AS05.02.19: The mandatory value for Extended Data Block Length for MINUSTD template shall be zero.

No requirements for vendor.

TE05.02.19.01: The tester shall verify that the value of Extended Data Block Length is zero.

5.3 Facial Image Stored on PIV Card

AS05.03.01: All facial images must conform to the requirements in SP80076 Table 12, "INCITS 385 Profile for PIV Facial Images."

VE05.03.01.01: The vendor shall include documentation of the procedure by which facial images are enrolled.

TE05.03.01.01: The tester shall review the documentation to verify compliance with the assertion.

AS05.03.02: The Format Identifier of the Facial Header shall be 0x46414300.

No requirements for vendor.

TE05.03.02.01: The tester shall verify that the Format Identifier value is 0x46414300.

AS05.03.03: The Version Number of the Facial Header shall be 0x30313000.

No requirements for vendor.

TE05.03.03.01: The tester shall verify that the Version Number is 0x30313000.

AS05.03.04: The Record Length of the Facial Header shall fit within the container size limits specified in [800-73].

No requirements for vendor.

TE05.03.04.01: The tester shall verify that the Record Length of the Facial fits within the container size limits specified in [800-73].

AS05.03.05: The Number of Facial Images of the Facial Header shall be ≥1 and the most recent image shall appear first and serve as the default provided to the application.

No requirements for vendor.

TE05.03.05.01: The tester shall verify that the Number of Facial Images of the Facial Header is ≥1 and that the most recent image appears first and serve as the default provided to the application.

AS05.03.06: The Number of Feature Points for the facial image shall be ≥0.

No requirements for vendor.

TE05.03.06.01: The tester shall verify that the Number of Feature Points for the facial image is ≥0.

AS05.03.07: The Facial Image Type shall be 1.

No requirements for vendor.

TE05.03.07.01: The tester shall verify that the Facial Image Type is 1.

AS05.03.08: The Image Data Type shall be 0 or 1.

No requirements for vendor.

TE05.03.08.01: The tester shall verify that the Image Data Type is 0 or 1.

Note: Both whole-image and single-region-of-interest (ROI) compression are permitted. 800-76 recommends that newly collected facial image should be compressed using ISO/IEC 15444 (i.e., JPEG 2000).

AS05.03.09: The Image Color Space of the facial image shall be 1 and shall be converted to the sRGB color space.

No requirements for vendor.

TE05.03.09.01: The tester shall verify that the Image Color Space of the facial image is 1 and is converted to the sRGB color space.

AS05.03.10: The Source Type of the facial image shall be 2 or 6.

No requirements for vendor.

TE05.03.10.01: The tester shall verify that the Source Type of the facial image is 2 or 6.

AS05.03.11: Facial images shall be compressed using a compression ratio no higher than 15:1. However, when facial images are stored on PIV cards, JPEG 2000 shall be used with

ROI compression in which the innermost region shall be centered on the face and compressed at no more than 24:1.

VE05.03.11.01: The vendor shall include documentation of the procedure by which facial images are enrolled.

TE05.03.11.01: This assertion is externally tested.

5.4 On-Card Comparison

AS05.04.01: The Biometric Information Templates (BITs) Group Template shall conform to the specifications in SP80076 section 5.5.1.

VE05.04.01.01: The vendor shall specify in its documentation the process flow of how the Biometric Information Templates (BITs) are generated.

TE05.04.01.01: The tester shall verify that the resultant BITs is in compliance with the assertion.

AS05.04.02: The Number of BITs (tag 0x02) (corresponding to the number of fingers that follow) in the BIT Group Template shall be 2, one for each finger.

No requirements for vendor.

TE05.04.02.01: The tester shall verify that the value for Number of Fingers is 2 in tag 0x02.

AS05.04.03: The Reference data qualifier used by VERIFY (tag 0x83) for the first finger shall be '96'.

No requirements for vendor.

TE05.04.03.01: The tester shall verify that the Reference data qualifier used by VERIFY (tag 0x83) for the first finger is '96'.

AS05.04.04: The Reference data qualifier used by VERIFY (tag 0x83) for the second finger shall be '97'.

No requirements for vendor.

TE05.04.04.01: The tester shall verify that the Reference data qualifier used by VERIFY (tag 0x83) for the second finger is '97'.

AS05.04.05: The Biometric type (tag 0x81) for the first and second finger shall be 08.

No requirements for vendor.

TE05.04.05.01: The tester shall verify that the Biometric type (tag 0x81) for the first and second finger is 08.

AS05.04.06: The BHT's Biometric subtype (tag 0x82) uses values for the first and second finger from ISO/IEC 19784-3:2007 (not CARD-MIN).

No requirements for vendor.

TE05.04.06.01: The tester shall verify that the subtype values for the first and second finger match the values identified in SP80076 Table 8.

AS05.04.07: The CBEFF BDB format owner (tag 0x87) for the first and second finger shall be set to 0101.

No requirements for vendor.

TE05.04.07.01: The tester shall verify that the CBEFF BDB format owner (tag 0x87) for the first and second finger is set to 0101.

AS05.04.08: The CBEFF BDB format type (tag 0x88) for the first and second finger shall be set to 0005.

No requirements for vendor.

TE05.04.08.01: The tester shall verify that the CBEFF BDB format type (tag 0x88) for the first and second finger is set to 0005.

AS05.04.09: Biometric Matching algorithm parameters [CARD-Min table 14] tag 83 shall NOT be present for the first and second finger.

No requirements for vendor.

TE05.04.09.01: The tester shall verify that TAG 83 is not present in the Biometric Matching parameters for the first and second finger.

5.5 Iris Image Stored on PIV Card

AS05.05.01: All iris images must conform to the requirements in SP80076 Table 9, "ISO/IEC 19794-6 profile for iris images stored on PIV Cards"

VE05.05.01.01: The vendor shall include documentation of the procedure by which iris images are enrolled.

TE05.05.01.01: The tester shall review the documentation to verify compliance with the assertion.

AS05.05.02: The Format identifier of the Iris General Header shall be 0x49495200.

No requirements for vendor.

TE05.05.02.01: The tester shall verify that the Format identifier of the Iris General Header is 0x49495200.

AS05.05.03: The Version number of the Iris General Header shall be 0x30323000.

No requirements for vendor.

TE05.05.03.01: The tester shall verify that the Version number of the Iris General Header is 0x30323000.

AS05.05.04: The Length of record of the Iris General Header must be less than or equal to size specified in 800-73-4 and JPEG 2000 compressed iris image implementations shall be executed with a bit rate input value that corresponds to the 3Kilobyte target result.

No requirements for vendor.

TE05.05.04.01: The tester shall verify that the Length of record of the Iris General Header is less than or equal to size specified in 800-73-4 and JPEG 2000 compressed iris image implementations are executed with a bit rate input value that corresponds to the 3Kilobyte target result.

AS05.05.05: The Number of iris representations of the Iris General Header shall be 1 or 2.

No requirements for vendor.

TE05.05.05.01: The tester shall verify that the Number of iris representations of the Iris General Header is 1 or 2.

AS05.05.06: The Certification flag of the Iris General Header shall be 0x00.

No requirements for vendor.

TE05.05.06.01: The tester shall verify that the Certification flag of the Iris General Header is 0x00.

AS05.05.07: The Number of eyes represented shall be 1 or 2.

No requirements for vendor.

TE05.05.07.01: The tester shall verify that the Number of eyes represented is 1 or 2.

AS05.05.08: The Capture date and time shall be 2011 onwards.

No requirements for vendor.

TE05.05.08.01: The tester shall verify that the Capture date and time is 2011 onwards.

AS05.05.09: The Representation number shall be 1 and then, optionally 2.

No requirements for vendor.

TE05.05.09.01: The tester shall verify that the Representation number is 1 and then, optionally 2.

AS05.05.10: The Eye label shall be 1 for left eye and 2 for right eye. If camera does not estimate automatically, then these shall be manually assigned.

No requirements for vendor.

TE05.05.10.01: The tester shall verify that the Eye label is 1 for left eye and 2 for right eye. If camera does not estimate automatically, then these are manually assigned.

AS05.05.11: The Image type shall be image type 7 with (0,6R 0,2R) margins.

No requirements for vendor.

TE05.05.11.01: The tester shall verify that the Image type is type 7 with (0,6R 0,2R) margins.

AS05.05.12: The Image format shall be 10 = 0x0A and the compression algorithm and encoding shall be mono JPEG 2000.

No requirements for vendor.

TE05.05.12.01: The tester shall verify that the Image format is 10 = 0x0A and the compression algorithm and encoding is mono JPEG 2000.

AS05.05.13: The Iris image properties bit field shall be (Bits 1-2: 01 or 10), (Bits 3-4: 01 or 10), (Bits 5-6: 01 and scan type shall be progressive), and (Bits 7-8: 01 and the compression history shall be "none").

No requirements for vendor.

TE05.05.13.01: The tester shall verify that the Iris image properties bit field is (Bits 1-2: 01 or 10), (Bits 3-4: 01 or 10), (Bits 5-6: 01 and scan type shall be progressive), and (Bits 7-8: 01 and the compression history shall be "none").

Note: Bit 1 is the least significant bit and Bit 8 is the most significant.

AS05.05.14: The Image width (image width, W) shall be $288 \leq W \leq 448$.

No requirements for vendor.

TE05.05.14.01: The tester shall verify that the Image width (image width, W) is $288 \leq W \leq 448$.

AS05.05.15: The Image height (image height, W) shall be $216 \leq H \leq 336$.

No requirements for vendor.

TE05.05.15.01: The tester shall verify that the Image height (image height, W) is $216 \leq H \leq 336$.

AS05.05.16: The Bit depth shall be 8.

No requirements for vendor.

TE05.05.16.01: The tester shall verify that the Bit depth is 8.

Note: Bit depth is in bits per pixel and shall not be used to indicate compression level.

AS05.05.17: The Iris centre, lowest X shall be (W/2 for W odd, else), highest X shall be (W/2+1 for W even), lowest Y shall be (H/2 for H odd, else), and highest X shall be (W/2+1 for H even).

No requirements for vendor.

TE05.05.17.01: The tester shall verify that the Iris centre, lowest X is (W/2 for W odd, else), highest X is (W/2+1 for W even), lowest Y is (H/2 for H odd, else), and highest Y is (W/2+1 for H even).

AS05.05.18: The Iris diameter, lowest shall be $D \geq 160$ and the highest shall be $D \leq 280$.

No requirements for vendor.

737 TE05.05.18.01: The tester shall verify that the Iris diameter, lowest is $D \geq 160$ and the highest is
738 $D \leq 280$.

6. Signed Data Elements DTRs

6.1 Card Holder Unique Identifier

6.1.1 Asymmetric Signature Conformance

AS06.01.01: The CHUID data object shall contain an Asymmetric digital signature of the CHUID object, which has been encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652.

No requirement for vendor.

TE06.01.01.01: The tester shall validate that the CHUID data object contains a digital signature and has been formatted correctly as a CMS external signature as defined in RFC 5652.

AS06.01.02: The digital signature is implemented as a SignedData Type.

No requirement for vendor.

TE06.01.02.01: The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.

AS06.01.03: The value of the version field of the SignedData content type shall be v3.

No requirement for vendor.

TE06.01.03.01: The tester shall validate the version of the SignedData type is version 3.

AS06.01.04: The digestAlgorithms field of the SignedData content type shall be in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.01.04.01: The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.

AS06.01.05: The eContentType of the encapContentInfo shall be id-PIV-CHUIDSecurityObject (OID = 2.16.840.1.101.3.6.1).

No requirement for vendor.

TE06.01.05.01: The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-CHUIDSecurityObject OID.

AS06.01.06: The encapContentInfo of the SignedData content type shall omit the eContent field.

No requirement for vendor.

TE06.01.06.01: The tester shall validate that the eContent field has been omitted from the encapContentInfo.

AS06.01.07: The certificates field shall include only a single X.509 certificate which is used to verify the signature in the SignerInfo field.

No requirement for vendor.

TE06.01.07.01: The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.

AS06.01.08: The crls field from the SignedData content type shall be omitted.

No requirement for vendor.

TE06.01.08.01: The tester shall validate that the crls field has been omitted from the SignedData.

AS06.01.09: The SignerInfos in the SignedData content type shall contain only a single SignerInfo type.

No requirement for vendor.

TE06.01.09.01: The tester shall validate that only a single SignerInfo exists in the SignedData.

AS06.01.10: The SignerInfo type shall use the issuerAndSerialNumber choice for the SignerIdentifier and this shall correspond to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the CHUID.

No requirement for vendor.

TE06.01.10.01: The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier and it corresponds to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the CHUID.

AS06.01.11: The SignerInfo type shall specify a digestAlgorithm in accordance with Table 3-2 of SP 800-78.

No requirement for vendor.

TE06.01.11.01: The tester shall validate that the digest algorithm is in accordance with Table 3-2.

AS06.01.12: The signedAttrs of the SignerInfo shall include the MessageDigest (OID = 1.2.840.113549.1.9.4) attribute containing the hash computed over the concatenated content of the CHUID, excluding the asymmetric signature field and the optional Buffer Length field (if present).

No requirement for vendor.

TE06.01.12.01: The tester shall validate the presence of a MessageDigest attribute in the signed attributes.

TE06.01.12.02: The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated content of the CHUID, excluding the asymmetric signature field and the optional Buffer Length field (if present).

AS06.01.13: The signedAttrs of the SignerInfo shall include the pivSigner-DN (OID = 2.16.840.1.101.3.6.5) attribute containing the subject name that appears in the X.509 certificate for the entity that signed the CHUID.

No requirement for vendor.

TE06.01.13.01: The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.

TE06.01.13.02: The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the CHUID.

AS06.01.14: The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding, the signatureAlgorithm field shall specify the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

No requirement for vendor.

TE06.01.14.01: The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.

AS06.01.15: The SignedData content type shall include the digital signature.

No requirement for vendor.

TE06.01.15.01: The tester shall validate that the SignedData content type includes the digital signature corresponding to the CHUID.

6.1.2 Certificate that signs the CHUID

Note: This requirement is not tested separately and is tested as part of Section 7.7: X.509 Certificate for Content Signing.

6.2 Biometric Fingerprint for Off-Card Comparison

6.2.1 Asymmetric Signature Conformance

AS06.02.01: The CBEFF_SIGNATURE_BLOCK shall be encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652.

No requirement for vendor.

TE06.02.01.01: The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.

AS06.02.02: The digital signature is implemented as a SignedData Type.

No requirement for vendor.

TE06.02.02.01: The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.

AS06.02.03: The value of the version field of the SignedData content type shall be v3.

No requirement for vendor.

TE06.02.03.01: The tester shall validate the version of the SignedData type is version 3.

AS06.02.04: The digestAlgorithms field of the SignedData content type shall be in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.02.04.01: The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.

AS06.02.05: The eContentType of the encapContentInfo shall be id-PIV-biometricObject (OID = 2.16.840.1.101.3.6.2).

No requirement for vendor.

TE06.02.05.01: The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.

AS06.02.06: The encapContentInfo of the SignedData content type shall omit the eContent field.

No requirement for vendor.

TE06.02.06.01: The tester shall validate that the eContent field has been omitted from the encapContentInfo.

AS06.02.07: If the signature on the biometric fingerprint was generated with a different key as the signature on the CHUID, the certificates field shall include only a single certificate in the SignerInfo field which can be used to verify the signature; else the certificates field shall be omitted.

VE06.02.07.01: The vendor shall state whether or not the same certificate was used to sign the CHUID and the Biometrics.

TE06.02.07.01: The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.

TE06.02.07.02: If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.

AS06.02.08: The crls field from the SignedData content type shall be omitted.

No requirement for vendor.

TE06.02.08.01: The tester shall validate that the crls field has been omitted from the SignedData.

AS06.02.09: The signerInfos in the SignedData content type shall contain only a single SignerInfo type.

No requirement for vendor.

TE06.02.09.01: The tester shall validate that only a single SignerInfo exists in the SignedData.

AS06.02.10: The SignerInfo type shall use the issuerAndSerialNumber choice for the SignerIdentifier and this shall correspond to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

No requirement for vendor.

TE06.02.10.01: The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier and it corresponds to the to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

AS06.02.11: The SignerInfo type shall specify a digestAlgorithm in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.02.11.01: The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.

AS06.02.12: The signedAttrs of the SignerInfo shall include the MessageDigest (OID = 1.2.840.113549.1.9.4) attribute containing the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

No requirement for vendor.

TE06.02.12.01: The tester shall validate the presence of a MessageDigest attribute in the signed attributes.

TE06.02.12.02: The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

AS06.02.13: The signedAttrs of the SignerInfo shall include the pivSigner-DN (OID = 2.16.840.1.101.3.6.5) attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometric fingerprint data.

No requirement for vendor.

TE06.02.13.01: The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.

TE06.02.13.02: The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.

AS06.02.14: The signedAttrs of the SignerInfo shall include the pivFASC-N (OID = 2.16.840.1.101.3.6.6) attribute containing the FASC-N of the PIV card.

No requirement for vendor.

TE06.02.14.01: The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.

TE06.02.14.02: The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in the CHUID.

AS06.02.15: The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding, the signatureAlgorithm field shall specify the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

No requirement for vendor.

TE06.02.15.01: The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP**80078**.

AS06.02.16: The SignedData content type shall include the digital signature.

No requirement for vendor.

TE06.02.16.01: The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.

AS06.02.17: The signedAttrs of the SignerInfo shall include an entryUUID (OID = 1.3.6.1.1.16.4) attribute [FRC4530] containing the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

No requirement for vendor.

TE06.02.17.01: The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.

TE06.02.17.02: The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

6.2.2 Certificate that signs the biometric fingerprint

Note: This requirement is not tested separately and is tested as part of Section 7.7: X.509 Certificate for Content Signing.

6.3 Biometric Facial Image

6.3.1 Asymmetric Signature Conformance

AS06.03.01: The CBEFF_SIGNATURE_BLOCK shall be encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652.

No requirement for vendor.

TE06.03.01.01: The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.

AS06.03.02: The digital signature is implemented as a SignedData Type.

No requirement for vendor.

TE06.03.02.01: The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.

AS06.03.03: The value of the version field of the SignedData content type shall be v3.

No requirement for vendor.

TE06.03.03.01: The tester shall validate the version of the SignedData type is version 3.

AS06.03.04: The digestAlgorithms field of the SignedData content type shall be in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.03.04.01: The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.

AS06.03.05: The eContentType of the encapContentInfo shall be id-PIV-biometricObject (OID = 2.16.840.1.101.3.6.2).

No requirement for vendor.

TE06.03.05.01: The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.

AS06.03.06: The encapContentInfo of the SignedData content type shall omit the eContent field.

No requirement for vendor.

TE06.03.06.01: The tester shall validate that the eContent field has been omitted from the encapContentInfo.

AS06.03.07: If the signature on the biometric facial image was generated with a different key as the signature on the CHUID, the certificates field shall include only a single certificate in the SignerInfo field which can be used to verify the signature; else the certificates field shall be omitted.

No requirement for vendor.

TE06.03.07.01: The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.

TE06.03.07.02: If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.

AS06.03.08: The crls field from the SignedData content type shall be omitted.

No requirement for vendor.

TE06.03.08.01: The tester shall validate that the crls field has been omitted from the SignedData.

AS06.03.09: The signerInfos in the SignedData content type shall contain only a single SignerInfo type.

No requirement for vendor.

TE06.03.09.01: The tester shall validate that only a single SignerInfo exists in the SignedData.

AS06.03.10: The SignerInfo type shall use the issuerAndSerialNumber choice for the SignerIdentifier and this shall correspond to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

No requirement for vendor.

TE06.03.10.01: The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier and it corresponds to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

AS06.03.11: The SignerInfo type shall specify a digestAlgorithm in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.03.11.01: The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.

AS06.03.12: The signedAttrs of the SignerInfo shall include the MessageDigest (OID = 1.2.840.113549.1.9.4) attribute containing the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

No requirement for vendor.

TE06.03.12.01: The tester shall validate the presence of a MessageDigest attribute in the signed attributes.

TE06.03.12.02: The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

AS06.03.13: The signedAttrs of the SignerInfo shall include the pivSigner-DN (OID = 2.16.840.1.101.3.6.5) attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometric data.

No requirement for vendor.

TE06.03.13.01: The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.

TE06.03.13.02: The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.

AS06.03.14: The signedAttrs of the SignerInfo shall include the pivFASC-N (OID = 2.16.840.1.101.3.6.6) attribute containing the FASC-N of the PIV card.

No requirement for vendor.

TE06.03.14.01: The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.

TE06.03.14.02: The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in the CHUID.

AS06.03.15: The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding, the signatureAlgorithm field shall specify the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

No requirement for vendor.

TE06.03.15.01: The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.

AS06.03.16: The SignedData content type shall include the digital signature.

No requirement for vendor.

TE06.03.16.01: The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.

AS06.03.17: The signedAttrs of the SignerInfo shall include an entryUUID (OID = 1.3.6.1.1.16.4) attribute [FRC4530] containing the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

No requirement for vendor.

TE06.03.17.01: The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.

TE06.03.17.02: The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

6.3.2 Certificate that signs the biometric facial image

Note: This requirement is not tested separately and is tested as part of Section 7.7: X.509 Certificate for Content Signing.

6.4 Security Object

6.4.1 Data Integrity Check

AS06.04.01: The message digest produced as a result of a hash function on the contents of a data object buffer shall be identical to that data object's message digest contained in the security object.

There are no vendor requirements.

TE06.04.01.01: The tester shall validate that the message digests for the various data objects present in the security object are identical to the message digest of the data object itself.

6.4.2 Asymmetric Signature Conformance

AS06.04.02: The security object buffer shall contain an asymmetric digital signature as specified in RFC (5652).

There are no vendor requirements.

TE06.04.02.01: The tester shall validate that the digital signature has been formatted correctly as a CMS signature as defined in RFC (5652).

AS06.04.03: The digital signature is implemented as a SignedData Type.

There are no vendor requirements.

TE06.04.03.01: The tester shall validate that the CMS digital signature has been implemented as a SignedData type.

AS06.04.04: The value of the version field of the SignedData content type shall be v3.

There are no vendor requirements.

TE06.04.04.01: The tester shall validate the version of the SignedData type is version 3.

AS06.04.05: The digestAlgorithms field of the SignedData content type shall be in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.04.05.01: The tester shall validate that the digest algorithm value is in accordance with Table 3-2 of SP80078.

AS06.04.06: The eContentType of the encapContentInfo shall be id-icao-ldsSecurityObject (OID = 1.3.27.1.1.1).

No requirement for vendor.

TE06.04.06.01: The tester shall validate that eContentType of the encapContentInfo asserts the id-icao-ldsSecurityObject OID.

AS06.04.07: The eContent of the encapContentsInfo field shall contain the encoded contents of the ldsSecurity object.

No requirement for vendor.

TE06.04.07.01: The tester shall validate that eContent of the encapContentInfo contains the contents of the ldsSecurity object.

AS06.04.08: The certificates field shall be omitted since it is included in the CHUID.

No requirement for vendor.

TE06.04.08.01: The tester shall validate that the certificates field has been omitted from the SignedData.

AS06.04.09: The digestAlgorithm field specified in the SignerInfo field is the same as in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.04.09.01: The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.

AS06.04.10: The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding, the signatureAlgorithm field shall specify the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

No requirement for vendor.

TE06.04.10.01: The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.

AS06.04.11: The SignedData content type shall include the digital signature.

No requirement for vendor.

TE06.04.11.01: The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed security object.

6.4.3 Certificate that signs the Security Object

Note: This requirement is not tested separately and is tested as part of Section 7.7: X.509 Certificate for Content Signing.

6.5 Biometric Iris

6.5.1 Asymmetric Signature Conformance

AS06.05.01: The CBEFF_SIGNATURE_BLOCK shall be encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652.

No requirement for vendor.

TE06.05.01.01: The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.

AS06.05.02: The digital signature is implemented as a SignedData Type.

No requirement for vendor.

TE06.05.02.01: The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.

AS06.05.03: The value of the version field of the SignedData content type shall be v3.

No requirement for vendor.

TE06.05.03.01: The tester shall validate the version of the SignedData type is version 3.

AS06.05.04: The digestAlgorithms field of the SignedData content type shall be in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.05.04.01: The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.

AS06.05.05: The eContentType of the encapContentInfo shall be id-PIV-biometricObject (OID = 2.16.840.1.101.3.6.2).

No requirement for vendor.

TE06.05.05.01: The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.

AS06.05.06: The encapContentInfo of the SignedData content type shall omit the eContent field.

No requirement for vendor.

TE06.05.06.01: The tester shall validate that the eContent field has been omitted from the encapContentInfo.

AS06.05.07: If the signature on the biometric iris was generated with a different key as the signature on the CHUID, the certificates field shall include only a single certificate in the SignerInfo field which can be used to verify the signature; else the certificates field shall be omitted.

VE06.05.07.01: The vendor shall state whether or not the same certificate was used to sign the CHUID and the Biometrics.

TE06.05.07.01: The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.

TE06.05.07.02: If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.

AS06.05.08: The crls field from the SignedData content type shall be omitted.

No requirement for vendor.

TE06.05.08.01: The tester shall validate that the crls field has been omitted from the SignedData.

AS06.05.09: The signerInfos in the SignedData content type shall contain only a single SignerInfo type.

No requirement for vendor.

TE06.05.09.01: The tester shall validate that only a single SignerInfo exists in the SignedData.

AS06.05.10: The SignerInfo type shall use the issuerAndSerialNumber choice for the SignerIdentifier and this shall correspond to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

No requirement for vendor.

TE06.05.10.01: The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier and it corresponds to the to the issuer and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.

AS06.05.11: The SignerInfo type shall specify a digestAlgorithm in accordance with Table 3-2 of SP80078.

No requirement for vendor.

TE06.05.11.01: The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.

AS06.05.12: The signedAttrs of the SignerInfo shall include the MessageDigest (OID = 1.2.840.113549.1.9.4) attribute containing the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

No requirement for vendor.

TE06.05.12.01: The tester shall validate the presence of a MessageDigest attribute in the signed attributes.

TE06.05.12.02: The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.

AS06.05.13: The signedAttrs of the SignerInfo shall include the pivSigner-DN (OID = 2.16.840.1.101.3.6.5) attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometric iris data.

No requirement for vendor.

TE06.05.13.01: The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.

TE06.05.13.02: The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.

AS06.05.14: The signedAttrs of the SignerInfo shall include the pivFASC-N (OID = 2.16.840.1.101.3.6.6) attribute containing the FASC-N of the PIV card.

No requirement for vendor.

TE06.05.14.01: The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.

TE06.05.14.02: The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in the CHUID.

AS06.05.15: The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding, the signatureAlgorithm field shall specify the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

No requirement for vendor.

TE06.05.15.01: The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.

AS06.05.16: The SignedData content type shall include the digital signature.

No requirement for vendor.

TE06.05.16.01: The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.

AS06.05.17: The signedAttrs of the SignerInfo shall include an entryUUID (OID = 1.3.6.1.1.16.4) attribute [FRC4530] containing the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

No requirement for vendor.

TE06.05.17.01: The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.

TE06.05.17.02: The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

6.5.2 Certificate that signs the biometric iris

Note: This requirement is not tested separately and is tested as part of Section 7.7: X.509 Certificate for Content Signing.

7. PKI Certificate Profile DTRs

7.1 PIV Authentication Certificate

7.1.1 Certificate Profile Conformance

AS07.01.01: The signature field in the certificate shall specify an algorithm from Table 3-3 of SP80078 in the AlgorithmIdentifier (other than sha1WithRSAEncryption).

VE07.01.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.01.01.01: The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.

AS07.01.02: If Rivest Shamir Adleman (RSA) with Probabilistic Signature Scheme (PSS) padding is used, the parameters field of the AlgorithmIdentifier type shall assert Secure Hash Algorithm (SHA) 256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For Elliptic Curve Digital Signature Algorithm (ECDSA), the parameters field is absent.

VE07.01.02.01: The vendor shall specify in its documentation the permitted values of the AlgorithmIdentifier field based on the signature algorithm used to sign certificates issued.

TE07.01.02.01: The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

AS07.01.03: The subjectPublicKeyInfo field shall assert an algorithm in the AlgorithmIdentifier in accordance with Table 3-4 of SP80078.

VE07.01.03.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate PIV authentication keys.

TE07.01.03.01: The tester shall validate that the algorithm used to generate PIV authentication keys are in accordance with Table 3-4 of SP80078.

AS07.01.04: If the public key algorithm is Elliptic Curve, then the parameters field contains the namedCurve choice populated with the appropriate OID from Table 3-5 of SP80078.

VE07.01.04.01: The vendor shall specify in its documentation the allowed values of the parameters field of the algorithm of the subjectPublicKeyInfo field as part of the PIV authentication certificate profile. These values shall be based on the algorithm used to generate the key pair.

TE07.01.04.01: The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the PIV authentication certificate issued by the vendor.

AS07.01.05: The keyUsage extension shall assert only the digitalSignature bit. No other bits shall be asserted.

VE07.01.05.01: The vendor shall specify in its documentation the assertion of the digitalSignature bit in the keyUsage extension as part of the PIV authentication certificate profile.

TE07.01.05.01: The tester shall validate the assertion of the digitalSignature bit in the keyUsage extension in the PIV authentication certificate issued by the vendor.

AS07.01.06: The policyIdentifier field in the certificatePolicies must assert id-fpki-common-authentication (OID = 2.16.840.1.101.3.2.1.3.13).

VE07.01.06.01: The vendor shall specify in its documentation the inclusion of the certificatePolicies extension which asserts the id-fki-common-authentication OID as part of the PIV authentication certificate profile.

TE07.01.06.01: The tester shall validate the presence of the id-fki-common-authentication OID in the certificatePolicies extension in the PIV authentication certificate issued by the vendor.

AS07.01.07: The authorityInfoAccess field shall contain an id-ad-ocsp accessMethod. The access location uses the Uniform Resource Identifier (URI) name form to specify the location of a Hypertext Transfer Protocol (HTTP) accessible Online Certificate Status Protocol (OCSP) Server distributing status information for this certificate.

VE07.01.07.01: The vendor shall specify in its documentation the inclusion of an id-ad-ocsp accessMethod in the authorityInfoAccess extension as part of the PIV authentication certificate profile. Additionally, the accessLocation for this accessMethod uses the URI name form to specify the location of an HTTP accessible OCSP server.

TE07.01.07.01: The tester shall validate the presence of an id-ad-ocsp accessMethod in the authorityInfoAccess extension in the PIV authentication certificate issued by the vendor. The tester shall also validate that the accessLocation for this accessMethod uses the URI name form and points to an HTTP accessible OCSP server.

AS07.01.08: The FASC-N and Card UUID shall be populated in the subjectAltName extension using the pivFASC-N attribute (OID = 2.16.840.1.101.3.6.6) for the FASC-N and URI for the Card UUID.

No requirements for vendor.

TE07.01.08.01: The tester shall validate that the FASC-N and Card UUID in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. The tester shall validate that no other name forms appear in the subjectAltName extension.

AS07.01.09: The piv-interim extension (OID = 2.16.840.1.101.3.6.9.1) shall be present and contain an interim_indicator field which is populated with a Boolean value. This extension is not critical.

VE07.01.09.01: The vendor shall specify in its documentation the use of this extension as part of the PIV authentication certificate profile.

TE07.01.09.01: The tester shall validate that the piv-interim extension is present in the PIV authentication certificate issued by the vendor.

AS07.01.10: The cRLDistributionPoints field shall contain a URI that uses HTTP and must point to a file that has an extension of ".crl" that contains the DER encoded CRL (see RFC 2585) for status information about the certificate.

VE07.01.10.01: The vendor shall specify in its documentation the inclusion of a URI in the cRLDistributionPoints field that uses HTTP.

TE07.01.10.01: The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the PIV authentication certificate.

AS07.01.11: The authorityInfoAccess field shall contain an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location shall use a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

No requirements for vendor.

TE07.01.11.01: The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

7.1.2 Key Pair and Certificate Conformance

AS07.01.12: The size of the public key for PIV authentication certificate shall be in accordance with Table 3-1of SP80078.

VE07.01.12.01: The vendor shall specify in its documentation the allowable public key size to be used while generating PIV authentication keys.

TE07.01.12.01: The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.

AS07.01.13: The public key present in the PIV authentication certificate correspond to the PIV authentication private key.

No requirement for vendor.

TE07.01.13.01: The tester shall validate that the public key present in the PIV authentication certificate is part of the key pair corresponding to the private key on the PIV card.

AS07.01.14: The FASC-N in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N present in the CHUID and the Card UUID is the same value as in the GUID, present in the CHUID.

No requirement for vendor.

TE07.01.14.01: The tester shall validate that the FASC-N in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N present in the CHUID in the PIV card.

TE07.01.14.02: The tester shall validate that the Card UUID present in the subjectAltName field is the same as the Card UUID in the CHUID in the PIV card.

AS07.01.15: The expiration of the PIV authentication certificate is not beyond the expiration of the CHUID.

No requirement for vendor.

TE07.01.15.01: The tester shall validate that the expiration of the PIV authentication certificate is not beyond the expiration of the CHUID in the PIV card.

AS07.01.16: If the public key algorithm is RSA, the exponent shall be equal to 65,537.

VE07.01.16.01: The vendor shall specify in its documentation the size of the exponent permitted while generating an RSA key pair for PIV authentication.

TE07.01.16.01: The tester shall validate that the RSA public key exponent size is equal to 65,537.

7.2 Digital Signature Certificate

7.2.1 Certificate Profile Conformance

AS07.02.01: The signature field in the certificate shall specify an algorithm from Table 3-3 of SP80078 in the AlgorithmIdentifier (other than sha1WithRSAEncryption).

VE07.02.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.02.01.01: The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.

AS07.02.02: If RSA with PSS padding is used, the parameters field of the AlgorithmIdentifier type shall assert SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

VE07.02.02.01: The vendor shall specify in its documentation the permitted values of the AlgorithmIdentifier field based on the signature algorithm used to sign certificates issued.

TE07.02.02.01: The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

AS07.02.03: The subjectPublicKeyInfo field shall assert an algorithm in the AlgorithmIdentifier in accordance with Table 3-4 of SP80078.

VE07.02.03.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate digital signature keys.

TE07.02.03.01: The tester shall validate that the algorithm used to generate digital signature keys are in accordance with Table 3-4 of SP80078.

AS07.02.04: If the public key algorithm is Elliptic Curve, then the parameters field contains the namedCurve choice populated with the appropriate OID from Table 3-5 of SP80078.

VE07.02.04.01: The vendor shall specify in its documentation the allowed values of the parameters field of the algorithm of the subjectPublicKeyInfo field as part of the digital signature certificate profile. These values shall be based on the algorithm used to generate the key pair.

TE07.02.04.01: The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the digital signature certificate issued by the vendor.

AS07.02.05: The keyUsage extension shall assert both the digitalSignature and nonRepudiation bits. No other bits shall be asserted.

VE07.02.05.01: The vendor shall specify in its documentation the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension as part of the digital signature certificate profile.

TE07.02.05.01: The tester shall validate the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension in the digital signature certificate issued by the vendor.

AS07.02.06: The policyIdentifier field in the certificatePolicies must assert one of the following: id-fpki-common-hardware (OID = 2.16.840.1.101.3.2.1.3.7) or id-fpki-common-High (OID = 2.16.840.1.101.3.2.1.3.16).[2]

VE07.02.06.01: The vendor shall specify in its documentation the inclusion of the certificatePolicies extension which asserts one of the following OIDs as part of the Digital Signature certificate profile: the id-fpki-common-hardware or id-fpki-common-High.

TE07.02.06.01: The tester shall validate the presence of one of the following OIDs in the certificatePolicies extension in the Digital Signature certificate issued by the vendor: the id-fpki-common-hardware or id-fpki-common-High.

AS07.02.07: The cRLDistributionPoints field shall contain a URI that uses HTTP and must point to a file that has an extension of ".crl" that contains the DER encoded CRL (see RFC 2585) for status information about the certificate.

VE07.02.07.01: The vendor shall specify in its documentation the inclusion of a URI in the cRLDistributionPoints field that uses HTTP.

TE07.02.07.01: The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Digital signature certificate.

AS07.02.08: The authorityInfoAccess field shall contain an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location shall use a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

No requirements for vendor.

TE07.02.08.01: The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

7.2.2 Key Pair and Certificate Conformance

AS07.02.09: The size of the public key for digital signature certificate shall be in accordance with Table 3-1 of SP80078.

VE07.02.09.01: The vendor shall specify in its documentation the allowable public key size to be used while generating digital signature keys.

TE07.02.09.01: The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.

[2] This test is not applicable to legacy PKIs.

AS07.02.10: The public key present in the digital signature certificate corresponds to the digital signature private key.

No requirement for vendor.

TE07.02.10.01: The tester shall validate that the public key present in the digital signature certificate is part of the key pair corresponding to the private key on the PIV card.

AS07.02.11: The expiration of the digital signature certificate is not beyond the expiration of the CHUID.

No requirement for vendor.

TE07.02.11.01: The tester shall validate that the expiration of the digital signature certificate is not beyond the expiration of the CHUID in the PIV card.

AS07.02.12: If the public key algorithm is RSA, the exponent shall be equal to 65,537.

VE07.02.12.01: The vendor shall specify in its documentation the size of the exponent permitted while generating an RSA key pair for digital signatures.

TE07.02.12.01: The tester shall validate that the RSA public key exponent size is equal to 65,537.

7.3 Key Management Certificate

7.3.1 Certificate Profile Conformance

AS07.03.01: The signature field in the certificate shall specify an algorithm from Table 3-3 of SP80078 in the AlgorithmIdentifier (other than sha1WithRSAEncryption).

VE07.03.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.03.01.01: The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.

AS07.03.02: If RSA with PSS padding is used, the parameters field of the AlgorithmIdentifier type shall assert SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

VE07.03.02.01: The vendor shall specify in its documentation the permitted values of the AlgorithmIdentifier field based on the signature algorithm used to sign certificates issued.

TE07.03.02.01: The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

AS07.03.03: The subjectPublicKeyInfo field shall assert an algorithm in the AlgorithmIdentifier in accordance with Table 3-4 of SP80078.

VE07.03.03.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate key management keys.

TE07.03.03.01: The tester shall validate that the algorithm used to generate key management keys are in accordance with Table 3-4 of SP80078.

AS07.03.04: If the public key algorithm is Elliptic Curve, then the parameters field contains the namedCurve choice populated with the appropriate OID from Table 3-5 of SP80078.

VE07.03.04.01: The vendor shall specify in its documentation the allowed values of the parameters field of the algorithm of the subjectPublicKeyInfo field as part of the key management certificate profile. These values shall be based on the algorithm used to generate the key pair.

TE07.03.04.01: The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the key management certificate issued by the vendor.

AS07.03.05: If the public key algorithm is RSA, then the keyUsage extension shall only assert the keyEncipherment bit.

VE07.03.05.01: The vendor shall specify in its documentation that certificates corresponding to RSA keys assert only the keyEncipherment bit in the keyUsage extension.

TE07.03.05.01: The tester shall validate that certificates corresponding to RSA keys assert only the keyEncipherment bit in the keyUsage extension.

AS07.03.06: If the public key algorithm is Elliptic Curve, then the keyUsage extension shall only assert the keyAgreement bit.

VE07.03.06.01: The vendor shall specify in its documentation that certificates corresponding to elliptic curve keys assert only the keyAgreement bit in the keyUsage extension.

TE07.03.06.01: The tester shall validate that certificates corresponding to elliptic curve keys assert only the keyAgreement bit in the keyUsage extension.

AS07.03.07: The policyIdentifier field in the certificatePolicies must assert one of the following: id-fpki-common-policy (OID = 2.16.840.1.101.3.2.1.3.6), id-fpki-common-hardware (OID = 2.16.840.1.101.3.2.1.3.7), or id-fpki-common-High (OID = 2.16.840.1.101.3.2.1.3.16).[3]

VE07.03.07.01: The vendor shall specify in its documentation the inclusion of the certificatePolicies extension which asserts one of the following OIDs as part of the Key Management certificate profile: the id-fpki-common-policy, id-fpki-common-hardware, or id-fpki-common-High.

TE07.03.07.01: The tester shall validate the presence of one of the following OIDs in the certificatePolicies extension in the Key Management certificate issued by the vendor: the id-fpki-common-policy, id-fpki-common-hardware, or id-fpki-common-High.

AS07.03.08: The cRLDistributionPoints field shall contain a URI that uses HTTP and must point to a file that has an extension of ".crl" that contains the DER encoded CRL (see RFC 2585) for status information about the certificate.

VE07.03.08.01: The vendor shall specify in its documentation the inclusion of a URI in the cRLDistributionPoints field that uses HTTP.

TE07.03.08.01: The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Key management certificate.

AS07.03.09: The authorityInfoAccess field shall contain an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location shall use a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

No requirements for vendor.

TE07.03.09.01: The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

[3] This test is not applicable to legacy PKIs

7.3.2 Key Pair and Certificate Conformance

AS07.03.10: The size of the public key for key management shall be in accordance with Table 3-1 of SP80078.

VE07.03.10.01: The vendor shall specify in its documentation the allowable public key size to be used while generating key management keys.

TE07.03.10.01: The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.

AS07.03.11: The public key present in the key management certificate corresponds to the key management private key.

No requirement for vendor.

TE07.03.11.01: The tester shall validate that the public key present in the key management certificate is part of the key pair corresponding to the private key on the PIV card.

AS07.03.12: If the public key algorithm is RSA, the exponent shall be equal to 65,537.

VE07.03.12.01: The vendor shall specify in its documentation the size of the exponent permitted while generating an RSA key pair for key management.

TE07.03.12.01: The tester shall validate that the RSA public key exponent size is equal to 65,537.

7.4 Card Authentication Certificate

7.4.1 Certificate Profile Conformance

AS07.04.01: The signature field in the certificate shall specify an algorithm from Table 3-3 of SP80078 in the AlgorithmIdentifier (other than sha1WithRSAEncryption).

VE07.04.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.04.01.01: The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.

AS07.04.02: If RSA with PSS padding is used, the parameters field of the AlgorithmIdentifier type shall assert SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

VE07.04.02.01: The vendor shall specify in its documentation the permitted values of the AlgorithmIdentifier field based on the signature algorithm used to sign certificates issued.

TE07.04.02.01: The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

AS07.04.03: The subjectPublicKeyInfo field shall assert an algorithm in the AlgorithmIdentifier in accordance with Table 3-4 of SP80078.

VE07.04.03.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate card authentication keys.

TE07.04.03.01: The tester shall validate that the algorithm used to generate card authentication keys are in accordance with Table 3-4 of SP80078.

AS07.04.04: If the public key algorithm is Elliptic Curve, then the parameters field contains the namedCurve choice populated with the appropriate OID from Table 3-5 of SP80078.

VE07.04.04.01: The vendor shall specify in its documentation the allowed values of the parameters field of the algorithm of the subjectPublicKeyInfo field as part of the card authentication certificate profile. These values shall be based on the algorithm used to generate the key pair.

TE07.04.04.01: The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the card authentication certificate issued by the vendor.

AS07.04.05: The keyUsage extension shall assert only the digitalSignature bit. No other bits shall be asserted.

VE07.04.05.01: The vendor shall specify in its documentation the assertion of the digitalSignature bit in the keyUsage extension as part of the card authentication certificate profile.

TE07.04.05.01: The tester shall validate the assertion of the digitalSignature bit in the keyUsage extension in the card authentication certificate issued by the vendor.

AS07.04.06: The policyIdentifier field in the certificatePolicies must assert id-fpki-common-cardAuth (OID = 2.16.840.1.101.3.2.1.3.17).

VE07.04.06.01: The vendor shall specify in its documentation that the policyIdentifier field in certificatePolicies asserts the id-fpki-common-cardAuth OID.

TE07.04.06.01: The tester shall validate the policyIdentifier field in certificatePolicies has asserted the id-fpki-common-cardAuth OID.

AS07.04.07: The extKeyUsage extension shall assert id-PIV-cardAuth (OID = 2.16.840.1.101.3.6.8). This extension is critical.

VE07.04.07.01: The vendor shall specify in its documentation that the extKeyUsage extension asserts the id-PIV-cardAuth OID.

TE07.04.07.01: The tester shall validate the extKeyUsage is present, is marked as critical, asserts the id-PIV-cardAuth OID, and does not assert any other OIDs.

AS07.04.08: The authorityInfoAccess field shall contain an id-ad-ocsp accessMethod. The access location uses the URI name form to specify the location of an HTTP accessible OCSP Server distributing status information for this certificate.

VE07.04.08.01: The vendor shall specify in its documentation the inclusion of an id-ad-ocsp accessMethod in the authorityInfoAccess extension as part of the card authentication certificate profile. Additionally, the accessLocation for this accessMethod uses the URI name form to specify the location of an HTTP accessible OCSP server.

TE07.04.08.01: The tester shall validate the presence of an id-ad-ocsp accessMethod in the authorityInfoAccess extension in the card authentication certificate issued by the vendor. The tester shall also validate that the accessLocation for this accessMethod uses the URI name form and points to an HTTP accessible OCSP server.

AS07.04.09: The FASC-N and Card UUID shall be populated in the subjectAltName extension using the pivFASC-N attribute OID = 2.16.840.1.101.3.6.6) for the FASC-N and URI for the Card UUID.

No requirements for vendor.

TE07.04.09.01: The tester shall validate that the FASC-N and Card UUID in the subjectAltName field in the Card authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. The tester shall validate that no other name forms appear in the subjectAltName extension.

AS07.04.10: The piv-interim extension (OID = 2.16.840.1.101.3.6.9.1) shall be present contain an interim_indicator field which is populated with a Boolean value. This extension is not critical.

VE07.04.10.01: The vendor shall specify in its documentation the use of this extension as part of the card authentication certificate profile.

TE07.04.10.01: The tester shall validate that the piv-interim extension is present in the card authentication certificate issued by the vendor.

AS07.04.11: The cRLDistributionPoints field shall contain a URI that uses HTTP and must point to a file that has an extension of ".crl" that contains the DER encoded CRL (see RFC 2585) for status information about the certificate.

VE07.04.11.01: The vendor shall specify in its documentation the inclusion of a URI in the cRLDistributionPoints field that uses HTTP.

TE07.04.11.01: The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Card authentication certificate.

AS07.04.12: The authorityInfoAccess field shall contain an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location shall use a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

No requirements for vendor.

TE07.04.12.01: The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

7.4.2 Key Pair and Certificate Conformance

AS07.04.13: The size of the public key for card authentication shall be in accordance with Table 3-1 of SP80078.

VE07.04.13.01: The vendor shall specify in its documentation the allowable public key size to be used while generating card authentication keys.

TE07.04.13.01: The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.

AS07.04.14: The public key present in the card authentication certificate correspond to the card authentication private key.

No requirement for vendor.

TE07.04.14.01: The tester shall validate that the public key present in the card authentication certificate is part of the key pair corresponding to the private key on the PIV card.

AS07.04.15: The FASC-N in the subjectAltName field in the Card authentication certificate is the same as the FASC-N present in the CHUID and the Card UUID is the same value as in the GUID, present in the CHUID.

No requirement for vendor.

TE07.04.15.01: The tester shall validate that the FASC-N in the subjectAltName field in the Card authentication certificate is the same as the FASC-N present in the CHUID in the PIV card.

TE07.04.15.02: The tester shall validate that the Card UUID present in the subjectAltName field is the same as the Card UUID in the CHUID in the PIV card.

AS07.04.16: If the public key algorithm is RSA, the exponent shall be equal to 65,537.

VE07.04.16.01: The vendor shall specify in its documentation the size of the exponent permitted while generating an RSA key pair for card authentication.

TE07.04.16.01: The tester shall validate that the RSA public key exponent size is equal to 65,537.

7.5 Secure Messaging Card Verifiable Certificate

7.5.1 Certificate Profile Conformance

AS07.05.01: The signature field in the certificate shall be in accordance with Table 15 of Part 2 in SP80073 and shall contain either an ECDSA signature, using P-256 if the CardHolderPublicKey is P-256 or P-384 if the CardHolderPublicKey is P-384.

VE07.05.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.05.01.01: The tester shall validate that signature field in the certificate is in accordance with Table 15 of Part 2 in SP80073 and contains either an ECDSA signature using P-256 if the CardHolderPublicKey is P-256 or P-384 if the CardHolderPublicKey is P-384.

AS07.05.02: The CardHolderPublicKey data object shall assert an algorithm in the Algorithm OID in accordance with Table 15 of SP80073 Part 2.

VE07.05.02.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate card authentication keys.

TE07.05.02.01: The tester shall validate that the algorithm used to generate card authentication keys are in accordance with Table 15 of SP80073 Part 2.

AS07.05.03: The Credential Profile Identifier of the secure messaging CVC shall be 0x80.

No requirements for vendor.

TE07.05.03.01: The tester shall verify that the Credential Profile Identifier of the secure messaging CVC is 0x80.

AS07.05.04: The Issuer Identification Number of the secure messaging CVC shall be the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature. If the public key needed to verify the signature on the secure messaging CVC appears in an Intermediate CVC, then value shall be the value of the Subject Identifier in the Intermediate CVC.

No requirements for the vendor.

TE07.05.04.01: The tester shall verify that the Issuer Identification Number of the secure messaging CVC is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature.

TE07.05.04.02: The tester shall verify that if the public key needed to verify the signature on the secure messaging CVC appears in an Intermediate CVC, then the Issuer Identification Number shall be the value of the Subject Identifier in the Intermediate CVC.

AS07.05.05: The Role Identifier of the secure messaging CVC shall be 0x00 for card-application key CVC.

No requirements for vendor.

TE07.05.05.01: The tester shall verify that the Role Identifier of the secure messaging CVC is 0x00 for card-application key CVC.

AS07.05.06: The Subject Identifier of the secure messaging CVC shall be the same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID.

No requirements for vendor.

TE07.05.06.01: The tester shall validate that the Subject Identifier of the secure messaging CVC is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID.

7.5.2 Key Pair and Certificate Conformance

AS07.05.07: If the PIV Card Application supports the virtual contact interface [SP80073] and the digital signature key, the key management key, or any of the retired key management keys are elliptic curve keys corresponding to Curve P-384, then the PIV Secure Messaging key shall use P-384, otherwise it may use P-256 or P-384.

No requirements for vendor.

TE07.05.07.01: The tester shall validate that the Secure Messaging key size is in accordance with the interfaces supported by the card and the keys that are present on the card.

AS07.05.08: The PIV card shall store a corresponding secure messaging CVC to support validation of the public key by the relying party. The format for the secure messaging CVC shall be as specified in SP80073 Part 2, Section 4.1.5.

No requirements for vendor.

TE07.05.08.01: This test is not tested separately and tested throughout Section 11.5: Secure Messaging Card Verifiable Certificate.

AS07.05.09: The Public Key present in the secure messaging CVC corresponds to the Secure Messaging private key.

No requirements for vendor.

TE07.05.09.01: The tester shall validate that the public key present in the CVC is part of the key pair corresponding to the secure messaging private key on the PIV card.

AS07.05.10: The public key required to verify the digital signature of the secure messaging CVC is an ECC key. It shall be provided in either an X.509 Certificate for Content Signing or an Intermediate CVC. If the public key required to verify the digital signature of the secure messaging CVC is provided in an Intermediate CVC, then the format of the Intermediate CVC shall be as specified in Table 16 of SP80073 Part 2, Section 4.1.5, and the public key required to verify the digital signature of the Intermediate CVC shall be provided in an X.509 Certificate for Content Signing.

No requirements for vendor

TE07.05.10.01: The tester shall validate that the public key is either an X.509 Certificate for Content Signing or an Intermediate CVC. If it is provided in an Intermediate CVC, then the format of the Intermediate CVC shall be as specified in Table 16 of SP80073 Part 2, Section 4.1.5, and the public key required to verify the digital signature of the Intermediate CVC shall be provided in an X.509 Certificate for Content Signing.

AS07.05.11: The X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card shall not be expired.

No requirements for vendor.

TE07.05.11.01: The tester shall validate that the X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card does not be expired.

AS07.05.12: The Public Key object of the secure messaging CVC shall be encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve.

No requirements for vendor.

TE07.05.12.01: The tester shall verify that the Public Key object of the secure messaging CVC is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve**.**

7.6 Intermediate Card Verifiable Certificate (CVC)

7.6.1 Certificate Profile Conformance

AS07.06.01: The signature field in the certificate shall be in accordance with Table 16 of Part 2 in SP80073 and shall contain an algorithm for RSA with SHA-256 and PKCS #1 v1.5 padding.

VE07.06.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.06.01.01: The tester shall validate that the signature field in the certificate is in accordance with Table 16 of Part 2 in SP80073 and contains an algorithm for RSA with SHA-256 and PKCS #1 v1.5 padding.

AS07.06.02: The CardHolderPublicKey data object shall assert an algorithm in the AlgorithmOID in accordance with Table 16 of SP80073 Part 2.

VE07.06.02.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate card authentication keys.

TE07.06.02.01: The tester shall validate that the algorithm used to generate card authentication keys are in accordance with Table 16 of SP80073 Part 2.

AS07.06.03: The Intermediate Card Verifiable Certificate shall have a tag value of 0x7F21.

VE07.06.03.01: The vendor shall specify in its document the tag value for the Card Verifiable Certificate.

TE07.06.03.01: The tester shall verify that the tag value of the Intermediate CVC is 0x7F21.

AS07.06.04: The Credential Profile Identifier of the Intermediate CVC shall be 0x80.

No requirements for vendor.

TE07.06.04.01: The tester shall verify that the Credential Profile Identifier of the Intermediate CVC is 0x80.

AS07.06.05: The Issuer Identification Number of the Intermediate CVC shall be the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature on the Intermediate CVC.

No requirements for the vendor.

TE07.06.05.01: The tester shall verify that the Issuer Identification Number of the Intermediate CVC is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature.

AS07.06.06: The Subject Identifier of the Intermediate CVC shall be the leftmost 8 bytes of the SHA-1 hash of the Public Key object.

No requirements for the vendor.

TE07.06.06.01: The tester shall verify that the Subject Identifier of the Intermediate CVC is the leftmost 8 bytes of the SHA-1 hash of the Public Key object.

AS07.06.07: The Algorithm OID of the Intermediate CVC shall be either 0x2A8648CE3D030107 for ECDH (Curve P-256) or 0x2B81040022 for ECDH (Curve P-384).

VE07.06.07.01: The vendor shall specify in its documentation the Algorithm OID of the Intermediate CVC.

TE07.06.07.01: The tester shall verify that the Algorithm OID of the Intermediate CVC is either 0x2A8648CE3D030107 for ECDH (Curve P-256) or 0x2B81040022 for ECDH (Curve P-384).

AS07.06.08: The Role Identifier of the Intermediate CVC shall be 0x12 for card-application root CVC.

No requirements for vendor.

TE07.06.08.01: The tester shall verify that the Role Identifier of the Intermediate CVC is 0x12 for card-application root CVC.

7.6.2 Key Pair and Certificate Conformance

AS07.06.09: The X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card shall not be expired.

No requirements for vendor.

TE07.06.09.01: The tester shall validate that the X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card is not expire.

AS07.06.10: The Public Key object of the Intermediate CVC shall be encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve.

No requirements for vendor.

TE07.06.10.01: The tester shall verify that the Public Key object of the Intermediate CVC is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve**.**

AS07.06.11: The public key in the Intermediate CVC used to verify the signature on the secure messaging CVC shall conform to Table 16 SP80073 Part 2, Section 4.1.5.

No requirements for vendor.

TE07.06.11.01: The tester shall verify that the public key in the Intermediate CVC used to verify the signature on the secure messaging CVC shall conform to Table 16 SP80073 Part 2, Section 4.1.5.

7.7 X.509 Certificate for Content Signing

7.7.1 Certificate Profile Conformance

AS07.07.01: The signature field in the certificate shall specify an algorithm from Table 3-3 of SP80078 in the AlgorithmIdentifier (other than sha1WithRSAEncryption).

VE07.07.01.01: The vendor shall specify in its documentation the algorithms used to sign certificates issued.

TE07.07.01.01: The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.

AS07.07.02: If RSA with PSS padding is used, the parameters field of the AlgorithmIdentifier type shall assert SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

VE07.07.02.01: The vendor shall specify in its documentation the permitted values of the AlgorithmIdentifier field based on the signature algorithm used to sign certificates issued.

TE07.07.02.01: The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

AS07.07.03: The subjectPublicKeyInfo field shall assert an algorithm in the AlgorithmIdentifier in accordance with Table 3-4 of SP80078.

VE07.07.03.01: The vendor shall specify in its documentation the applicable algorithms that can be used to generate the X.509 Certificate for Content Signing.

TE07.07.03.01: The tester shall validate that the algorithm used to generate the X.509 Certificate for Content Signing are in accordance with Table 3-4 of SP80078.

AS07.07.04: If the public key algorithm is Elliptic Curve, then the parameters field contains the namedCurve choice populated with the appropriate OID from Table 3-5 of SP80078.

VE07.07.04.01: The vendor shall specify in its documentation the allowed values of the parameters field of the algorithm of the subjectPublicKeyInfo field as part of the X.509 Certificate for Content Signing profile. These values shall be based on the algorithm used to generate the key pair.

TE07.07.04.01: The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the X.509 Certificate for Content Signing issued by the vendor.

AS07.07.05: The keyUsage extension shall assert both the digitalSignature and nonRepudiation bits. No other bits shall be asserted.

VE07.07.05.01: The vendor shall specify in its documentation the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension as part of the X.509 Certificate for Content Signing profile.

TE07.07.05.01: The tester shall validate the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension in the X.509 Certificate for Content Signing issued by the vendor.

AS07.07.06: For signatures created before October 15, 2015, the public key required to verify the digital signature shall be provided in the certificates field of the CMS external digital signature in a content signing certificate, which shall be an X.509 digital signature certificate issued under the id-fpki-common-hardware, the public key required to verify the digital signature shall be provided in the certificates field of the CMS external digital signature in a content signing certificate, which shall be an X.509 digital signature certificate issued under the id-fpki-common-piv-contentSigning policy of [COMMON]. The content signing certificate shall also include an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing.

No requirements for vendor.

TE07.07.06.01: The tester shall validate that the signatures created before October 15, 2015, the public key required to verify the digital signature is provided in the certificates field of the CMS external digital signature in a content signing certificate, which is an X.509 digital signature certificate issued under the id-fpki-common-hardware, the public key required to verify the digital signature shall be provided in the certificates field of the CMS external digital signature in a content signing certificate, which is an X.509 digital signature certificate issued under the id-fpki-common-piv-contentSigning policy of [COMMON]. The content signing certificate also includes an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing.

AS07.07.07: The cRLDistributionPoints field shall contain a URI that uses HTTP and must point to a file that has an extension of ".crl" that contains the DER encoded CRL (see RFC 2585) for status information about the certificate.

VE07.07.07.01: The vendor shall specify in its documentation the inclusion of a URI in the cRLDistributionPoints field that uses HTTP.

TE07.07.07.01: The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the X.509 Certificate for Content Signing.

AS07.07.08: The authorityInfoAccess field shall contain an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location shall use a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).No requirements for vendor.

TE07.07.08.01: The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

7.7.2 Key Pair and Certificate Conformance

AS07.07.09: The size of the public key for X.509 Certificate for Content Signing shall be in accordance with Table 3-2 of SP80078.

VE07.07.09.01: The vendor shall specify in its documentation the allowable public key size to be used for signing PIV data objects.

TE07.07.09.01: The tester shall validate that the public key size is in accordance with Table 3-2 of SP80078.

AS07.07.10: If the public key algorithm is RSA, the exponent shall be equal to 65,537.

No requirements for vendor.

TE07.07.10.01: The tester shall validate that the RSA public key exponent size is equal to 65,537.

AS07.07.11: The X.509 Certificate for Content Signing shall not be expired.

No requirement for vendor.

TE07.07.11.01: The tester shall verify that the X.509 Certificate for Content Signing is not expired.

8. BER-TLV Test Assertions

Assumptions:

1.0	When the length of the value field is between 0 and 127 bytes, the length field consists of a single byte where bit 8 is set to 0 and bits 7 to 1 encode the number of bytes in the value field. When the length of the value field is greater than 127 bytes, the length field consists of two or more bytes. The first byte is '81', '82', '83' or '84' where the low order nibble of each of these possible first-byte values (1, 2, 3, or 4 respectively) encodes the number of subsequent and remaining bytes in the length field. These subsequent and remaining bytes are taken together in order to be a big-endian integer encoding the number of bytes in the value field. Table 8-1 shows the encoding of the length field.
1.1	Except for the Discovery Object tag and the BIT Group Template, each BER-TLV tag is encoded as three bytes.
1.2	Each data object returned is appended with a 2 byte status word.
1.3	All variable length value fields can have zero lengths, which will result in a tag length field being immediately followed by the next tag, if applicable.
1.4	The final byte of the command string can be set to 0x00 to retrieve an entire data object regardless of the size of that object.

Number of Bytes in the Length Field	First Byte	Subsequent Bytes	Length of the Value Field
1 byte	'00' to '7F'	None	0 to 127
2 byte	'81'	'00' to 'FF'	0 to 255
3 byte	'82'	'0000' to 'FFFF'	0 to 65,535
4 byte	'83'	'000000' to 'FFFFFF'	0 to 16,777,215
5 byte	'84'	'00000000' to 'FFFFFFFF'	0 to 4,294,967,295

Table 8-1. Encoding of Length Field

8.1 "Card Capabilities Container" Data Object

Purpose	Confirms that the CCC of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.02.01

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CCC is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>>. 2. Set OID := <<CCC (2.16.840.1.101.3.7.1.219.0)>>. 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. The registered data model element is present and has a value of 0x10. 2. All mandatory tags in CCC table are present and may not have a value field. 3. The values of the available tags conform with the vendor provided data.

1891

1892 **8.2 CHUID Data Object**

Purpose	Confirms that the CHUID of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.03.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. All mandatory tags in CHUID table are present. 2. The Agency Code, System Code, and Credential Number of the FASC-N are present. The credential series, individual credential issue, person identifier, organizational category, organizational identifier, and person/organization association category of the FASC-N are populated

	or contain all zeros. 3. Expiration date is encoded as YYYYMMDD and its date-value is within the next six years. 4. The Global Unique Identification number (GUID) includes a Card Universally Unique Identifier (UUID) as described in 800-73-4 Part 1 Section 3.4.1. 5. If the CHUID contains the optional Cardholder UUID, then the data element shall be in accordance with 800 73-4 Part 1 Section 3.4.2. 6. The retired key map field is absent.

1893

1894

1895 **8.3 "X.509 Certificate for PIV Authentication" Data Object**

Purpose	Confirms that the X.509 Certificate for PIV Authentication of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073. Part 1
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid X.509 certificate for PIV authentication object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "X.509 Certificate for PIV Authentication" table are present.

1896

1897 **8.4 Off-Card Comparison "Card Holder Fingerprints" Data Object**

Purpose	Confirms that the "Card Holder Fingerprints" data object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073, Part 1
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.04.01

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. All mandatory tags in "Card Holder Fingerprints" table are present. 2. The fingerprint data is nested in tag value 0xBC.

8.5 "Printed Information" Data Object

Purpose	Confirms that the "Printed Information" Data Object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid printed information is stored on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Printed Information (2.16.840.1.101.3.7.2.48.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "Printed Information" table are present.

8.6 "Card Holder Facial Image" Data Object

Purpose	Confirms that the "Card Holder Facial Image" data object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.05.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. All mandatory tags in "Card Holder Facial Image" table are present. 2. The facial image data is nested in tag value 0xBC

8.7 "X.509 Certificate for Digital Signature" Data Object

Purpose	Confirms that the X.509 Certificate for Digital Signature of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1 Appendix A 2. AS04.01.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid X.509 certificate for digital signature object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "X.509 Certificate for Digital Signature" table are present.

1903

1904 **8.8 "X.509 Certificate for Key Management" Data Object**

Purpose	Confirms that the X.509 Certificate for Key Management of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid X.509 certificate for key management object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "X.509 Certificate for Key Management" table are present.

1905

1906 **8.9 "X.509 Certificate for Card Authentication" Data Object**

Purpose	Confirms that the X.509 Certificate for Card Authentication of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid X.509 certificate for card authentication object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "X.509 Certificate for Card Authentication" table are present.

1907

1908 **8.10 "Security Object" Data Object**

Purpose	Confirms that the "Security Object" data object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1 2. AS04.01.01 3. AS04.06.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format. 5. Parse the tag 0xBA to extract the Data Groups to Container ID mapping instances. 6. Verify that all unsigned data objects, such as the Printed Information data object, are included in the Security Object if present and that the message digests for the various data objects present in the security object are identical to the message digest of the data object itself 7. Verify that the PIV data containers exist on the card by selecting each container.
Expected Result(s)	1. From Step 4: All mandatory tags in "Security Object" table are present. 2. From Step 6: All unsigned data objects (if present) are included in the Security Object and the message digests are identical to the message digest of the actual data object. 3. From Step 7: Verify that all data containers found in the mapping are actually present in the card by performing a select on each container and expecting '90 00' in all cases.

1909

1910 **8.11 "Discovery Object" Data Object**

Purpose	Confirms that the Discovery Object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073, Part 1.
Reference(s)	1. SP80073, Part 1, Appendix A 2. AS04.01.01 3. AS04.09.01 4. AS04.10.02
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Discovery Object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Discover Object (2.16.840.1.101.3.7.2.96.80)>> 3. Call pivGetData w/ • (IN) cardHandle

	• (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. All mandatory tags in the Discovery Object are present, specifically both tag 0x4F (PIV Card Application AID) and tag 0x5F2F (PIN Usage Policy). 2. The values of the tags conform with the vendor provided data. 3. The PIN usage policy matches the card capabilities provided by the vendor documentation. Associated optional data objects are present when the PIN usage policy asserts an optional capability (i.e., OCC, global PIN and pairing code) 4. From step 4, '90 00' is returned.

1911

1912

1913 **8.12 "Card Holder Iris Images" Data Object**

Purpose	Confirms that the "Card Holder Iris Images" data object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073, Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.07.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	1. All mandatory tags in "Card Holder Iris Images" table are present. 2. The iris data is nested in tag value 0xBC

1914

1915

8.13 "Retired X.509 Certificate for Key Management" Data Object(s)

Purpose	Confirms that the Retire X.509 Certificate(s) for Key Management of the PIV Card Application conform to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Retired X.509 certificate for key management object is present on the PIV card.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Retired Key Management Certificate (2.16.840.1.101.3.7.2.16.1–2.16.840.1.101.3.7.2.16.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format 5. Using the public key in the retired certificate key management certificate, issue a challenge to the corresponding private key on the card.
Expected Result(s)	1. From Step 4: All mandatory tags in "Retired X.509 Certificate for Key Management" table are present. 2. From Step 5: The responses to all challenges to the associated private (retired) key matches.

8.14 "Key History" Data Object

Purpose	Confirms that the Key History of the PIV Card Application conforms to the logical requirements in Section 3.3.7 of SP80073 Part 1 and the PIV data model requirements as per Appendix A of 800-73-4 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.08.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A Key History object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	Step 1: Set cardHandle := <<valid card handle>>

	Step 2: Set OID := <<Key History Object (2.16.840.1.101.3.7.2.96.96)>> Step 3: Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data Step 4: Read and parse the byte array in accordance with BER-TLV format. If keysWithOnCardCerts = 0 and keysWithOffCardCerts > 0 Read the certificate(s) and key references (pairs) from the vendor provided URL file. For each key reference value in the range (0x95 – keysWithOffCardCerts + 1) through 0x95, verify that the provided URL file includes that key reference, issue a challenge for that key reference, and verify the response using the public key from the corresponding certificate from the provided URL file. If keysWithOnCardCerts > 0 and keyWithOffCardCerts = 0 For each key reference value in the range 0x82 through (0x82 + keysWithOnCardCerts – 1), read the certificates from the card. Issue a challenge and verify response for each retired private key. If keysWithOnCardCerts > 0 and keyWithOffCardCerts > 0 For each key reference value in the range 0x82 through (0x82 + keysWithOnCardCerts – 1) and in the range (0x95 – keysWithOffCardCerts + 1) through 0x95, verify that the provided URL file includes that key reference, issue a challenge for that key reference, and verify the response using the public key from the corresponding certificate from the provided URL file.
Expected Result(s)	1. Step 4: All mandatory tags in "Key History Object" table are present. And the values associated with keysWithOnCardCerts > 0 and keyWithOffCardCerts are consistent with the data on the card based on the details in step 4.

8.15 "Biometric Information Templates Group Template" Data Object

Purpose	Confirms that the Biometric Information Templates Group Template of the PIV Card Application conforms to the PIV data model requirements

	as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.10.01
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid BIT Group Template data object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Biometric Information Templates Group Template (2.16.840.1.101.3.7.2.16.22)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in “BIT Group Template” table are present.

8.16 “Secure Messaging Certificate Signer” Data Object

Purpose	Confirms that the Secure Messaging Certificate Signer data object of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1, Appendix A 2. AS04.01.01 3. AS04.11.01
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Secure Messaging Certificate Signer data object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := << Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected	All mandatory tags in “Secure Messaging Certificate Signer” table are

Result(s)	present.

1923

1924 ### 8.17 "Pairing Code Reference Data Container" Data Object

Purpose	Confirms that the Pairing Code Reference Data Container of the PIV Card Application conforms to the PIV data model requirements as per Appendix A of SP80073 Part 1.
Reference(s)	1. SP80073 Part 1 2. AS04.01.01 3. AS04.12.01
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Pairing Code Reference Data Container data object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := << Pairing Code Reference Data Container (2.16.840.1.101.3.7.2.16.24)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Read and parse the byte array in accordance with BER-TLV format.
Expected Result(s)	All mandatory tags in "Pairing Code Reference Data Container" table are present.

1925

1926

1927 ### 8.18 Unused Data Objects on the PIV Card

Purpose	Confirms that data objects that are created but not used shall be set to zero-length value.
Reference(s)	4. SP80073 Part 1 5. AS04.01.01 6. AS04.13.01
Precondition	5. A valid PIV card is inserted into the contact reader. 6. A valid PC/SC connection exists between the test application and the contact reader. 7. The test application is currently connected to the card application which is accessible through card handle.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := << Vendor Supply Unused Data Objects on the PIV Card (OID Variable)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 8. Read and parse the byte array in accordance with BER-TLV format. 9. Repeat steps 1 - 4 for all unused data objects
Expected Result(s)	For step 9: Each data containers that is created but not personalized returns the container's tag and length (L = 0x00). The value field is absent.

9. Biometric Data Object Test Assertions

The test assertions documented in this section relate to test specification of the Off-Card Comparison Fingerprint and Facial Image. It specifies the test cases on conformance to the common CBEFF Patron Format (Table 15 of SP80076) as well the corresponding INCITS 378 minutiae template profile (for On-Card fingerprint Minutia, Table 6 of SP80076) and INCITS 385 (for a Facial Image profile).

9.1 CBEFF Patron Format for Off-Card Comparison Fingerprint Template

9.1.1 CBEFF Structure for Fingerprint Template

Purpose	Validates that the CBEFF structure generated complies with SP80076 Table 13, "CBEFF concatenation structure".
References(s)	1. SP80076, Table 13. 2. AS05.01.01 3. AS05.01.03
Precondition(s)	1. All templates have been generated and stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. A valid Card Holder Fingerprints object is present on the PIV card. 6. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the BDB Length field of CBEFF Header. 5. Extract the SB Length field of CBEFF Header.
Expected Result(s)	1. From step 4: BDB Length field is non-zero and the recorded length matches the actual length. 2. From step 5: SB Length field is non-zero and the recorded length matches the actual length. 3. The Card Holder Fingerprint object length is equal to CBEFF Header length + BDB Length + SB Length.

9.1.2 CBEFF Header for Off-Card Comparison Fingerprint Template

9.1.2.1 Patron Header Version

Purpose	Validates that the CBEFF field "Patron Header Version" complies with SP80076 Table 14, "Patron Format PIV Specification"
References(s)	1. SP80076, Table 14.

	2. AS05.01.02 3. AS05.01.04
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Patron Header Version field (based on its position) in CBEFF Header.
Expected Result(s)	From Step 4: The Patron Header Version field has a value of 0x03.

1938 #### 9.1.2.2 SBH Security Option

Purpose	Validates that the biometric data block on the PIV card is digitally signed but not encrypted.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.05
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the SBH Security Option field (based on its position) in CBEFF Header.
Expected Result(s)	From Step 4: The SBH Security Options field has a value of b00001101.

9.1.2.3 BDB Format Owner Values

Purpose	Validates that BDB Format Owner is set to a value of 0x001B denoting M1, the INCITS Technical Committee on Biometrics.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.06
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Owner field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Owner field has a value of 0x001B.

9.1.2.4 BDB Format Type

Purpose	Validates that for mandatory fingerprint minutiae template data stored on a PIV card, the BDB Format Type is set to a value of 0x0201.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.07
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Type field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Type field has a value of 0x0201.

1941 #### 9.1.2.5 Biometric Creation Date

Purpose	Validates that the creation date in the PIV Patron Format is encoded in 8 bytes using a binary representation of "YYYYMMDDhhmmssZ".
References(s)	1. SP80076, Table 14. 2. AS05.01.02 3. AS05.01.03 4. AS05.03.08
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Creation Date field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Creation Date is in the YYYYMMDDhhmmssZ format.

1942 #### 9.1.2.6 Validity Period Dates

Purpose	Validates that the Validity Period in the PIV Patron Format contains two dates encoded in the same format as expected in 9.1.2.5 above.
References(s)	1. SP80076, Table 14 2. AS05.01.02 3. AS05.01.03 4. AS05.01.09

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Validity Period field (based on its position) in CBEFF Header.
Expected Result(s)	The Validity Period contains two dates and each is encoded in the YYYYMMDDhhmmssZ format.

1943 #### 9.1.2.7 Biometric Type Values

Purpose	Validates that Biometric Type has the value 0x000008
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.10
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Type field (based on its position) in CBEFF Header.
Expected Result(s)	The value of the Biometric Type field for the fingerprint template is 0x000008

1944 **9.1.2.8 Biometric Data Type**

Purpose	Validates that for the mandatory minutia PIV card templates, the CBEFF biometric data type encoding value shall be `b100xxxxx`, which corresponds to biometric data that has been processed.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.11
Precondition(s)	1. All required sample finger images have been recorded. 2. All templates have been generated and stored on the PIV card. 3. A valid PIV card is inserted into the contact reader. 4. A valid PC/SC connection exists between the test application and the contact reader. 5. The test application is currently connected to the card application which is accessible through card handle. 6. A valid Card Holder Fingerprints object is present on the PIV card. 7. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Type field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Type field has a value of b100xxxxx.

1945 **9.1.2.9 Biometric Data Quality**

Purpose	Validates that the biometric data quality field carries valid values
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.12
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Quality field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Quality field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Quality field has a value between -2 and 100.

1946 **9.1.2.10 Creator Field Value**

Purpose	Validates that the Creator field in the PIV Patron Format contains 18 bytes of which the first K <= 17 bytes shall be ASCII characters, and the first of the remaining 18-K shall be a null terminator (zero).
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.13
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Creator field (based on its position) in CBEFF Header. Start the following procedure at the first byte of the Creator field: 5. For bytes 0 to 16, check if byte represents an ASCII character If yes, continue. Else, check if zero. If no, fail. Else iterate through remaining bytes in field and fail if non-zero. 6. For bytes 17 through the end of the field, iterate through all bytes and fail if non-zero

Expected Result(s)	Procedure completes without failing.

1947 **9.1.2.11 FASC-N Value**

Purpose	The FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.14
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the FASC-N field of CBEFF Header. 5. Set cardHandle := <<valid card handle>> 6. Set OID := <<Card Holder Unique Identifier (2.16.840.1.101.3.7.2.48.0)>> 7. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 8. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the FASC-N field (based on its position) in CBEFF Header.
Expected Result(s)	The FASC-N field in CBEFF header is the same as the one extracted from the CHUID.

1948 **9.1.2.12 Reserved Field Value**

Purpose	Validates that the “Reserved for Future Use” field is equal to 0x00000000.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.15

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the “Reserved for Future Use” field (based on its position) in CBEFF Header.
Expected Result(s)	The field has a value of 0x00000000.

1949

1950

9.2 CBEFF Patron Format for Facial Image

9.2.1 CBEFF Structure for Facial Image

Purpose	Validates that the CBEFF structure generated complies with SP80076 Table 13, "CBEFF concatenation structure".
References(s)	1. SP80076, Table 13. 2. AS05.01.01 3. AS05.01.03
Precondition(s)	1. All templates have been generated and stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. A valid Card Holder Facial Image object is present on the PIV card. 6. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the BDB Length field of CBEFF Header. 5. Extract the SB Length field of CBEFF Header.
Expected Result(s)	1. From step 4: BDB Length field is non-zero and the recorded length matches the actual length. 2. From step 5: SB Length field is non-zero and the recorded length matches the actual length. 3. The Card Holder Facial Image object length is equal to CBEFF Header length + BDB Length + SB Length.

9.2.2 CBEFF Header for Facial Image

9.2.2.1 Patron Header Version

Purpose	Validates that the CBEFF field "Patron Header Version" complies with SP80076 Table 14, "Patron Format PIV Specification"
References(s)	1. SP80076, Table 14. 2. AS05.01.02 3. AS05.01.04

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Patron Header Version field (based on its position) in CBEFF Header.
Expected Result(s)	The Patron Header Version field has a value of 0x03.

1955 #### 9.2.2.2 SBH Security Option

Purpose	Validates that the biometric data block on the PIV card is digitally signed but not encrypted.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.05
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the SBH Security Option field (based on its position) in CBEFF Header.
Expected Result(s)	The SBH Security Options field has a value of b00001101.

1956

1957

1958 #### 9.2.2.3 BDB Format Owner Values

Purpose	Validates that BDB Format Owner is set to a value of 0x001B denoting M1, the INCITS Technical Committee on Biometrics.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.06
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Owner field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Owner field has a value of 0x001B.

1959 #### 9.2.2.4 BDB Format Type

Purpose	Validates that for mandatory facial image data stored on a PIV card, the BDB Format Type is set to a value of 0x0501.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.07
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Type field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Type field has a value of 0x0501.

1960 #### 9.2.2.5 Biometric Creation Date

Purpose	Validates that the creation date in the PIV Patron Format is encoded in 8 bytes using a binary representation of "YYYYMMDDhhmmssZ".
References(s)	1. SP80076, Table 14. 2. AS05.01.02 3. AS05.01.03 4. AS05.01.08
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Creation Date field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Creation Date is in the YYYYMMDDhhmmssZ format.

1961 #### 9.2.2.6 Validity Period Dates

Purpose	Validates that the Validity Period in the PIV Patron Format contains two dates encoded in the same format as expected in 9.2.2.5 above.
References(s)	1. SP80076, Table 14 2. AS05.01.02 3. AS05.01.03 4. AS05.01.09

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Validity Period field (based on its position) in CBEFF Header.
Expected Result(s)	The Validity Period contains two dates and each is encoded in the YYYYMMDDhhmmssZ format.

1962 #### 9.2.2.7 Biometric Type Values

Purpose	Validates that Biometric Type has the value 0x000002
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.10
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Type field (based on its position) in CBEFF Header.
Expected Result(s)	The value of the Biometric Type field for the facial image is 0x000002

9.2.2.8 Biometric Data Type

Purpose	Validates that the CBEFF biometric data type encoding value shall be `b001xxxxx`, which corresponds to the raw biometric data.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.11
Precondition(s)	1. All required sample finger images have been recorded. 2. All templates have been generated and stored on the PIV card. 3. A valid PIV card is inserted into the contact reader. 4. A valid PC/SC connection exists between the test application and the contact reader. 5. The test application is currently connected to the card application which is accessible through card handle. 6. A valid Card Holder Facial Image object is present on the PIV card. 7. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Type field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Type field has a value of b001xxxxx.

9.2.2.9 Biometric Data Quality

Purpose	Validates that the biometric data quality field carries valid values
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.12
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Quality field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Quality field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Quality field is -2 or 0 - 100.

1965 **9.2.2.10 Creator Field Value**

Purpose	Validates that the Creator field in the PIV Patron Format contains 18 bytes of which the first K <= 17 bytes shall be ASCII characters, and the first of the remaining 18-K shall be a null terminator (zero).
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.13
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Creator field (based on its position) in CBEFF Header. Start the following procedure at the first byte of the Creator field: 5. For bytes 0 to 16, check if byte represents an ASCII character If yes, continue. Else, check if zero. If no, fail. Else iterate through remaining bytes in field and fail if non-zero. 6. For bytes 17 through the end of the field, iterate through all bytes and fail if non-zero

Expected Result(s)	Procedure completes without failing.

1966

1967

1968

1969 #### 9.2.2.11 FASC-N Value

Purpose	The FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.14
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the FASC-N field of CBEFF Header. 5. Set cardHandle := <<valid card handle>> 6. Set OID := <<Card Holder Unique Identifier (2.16.840.1.101.3.7.2.48.0)>> 7. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 8. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the FASC-N field (based on its position) in CBEFF Header.
Expected Result(s)	The FASC-N field in CBEFF header is the same as the one extracted from the CHUID.

1970 #### 9.2.2.12 Reserved Field Value

Purpose	Validates that the “Reserved for Future Use” field is equal to 0x00000000.

References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.15
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Facial Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the "Reserved for Future Use" field (based on its position) in CBEFF Header.
Expected Result(s)	The field has a value of 0x00000000.

9.3 CBEFF Patron Format for Iris Image

9.3.1 CBEFF Structure for Iris Image

Purpose	Validates that the CBEFF structure generated complies with SP80076 Table 13, "CBEFF concatenation structure".
References(s)	1. SP80076, Table 13. 2. AS05.01.01 3. AS05.01.03
Precondition(s)	1. All templates have been generated and stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. A valid Card Holder Iris Image object is present on the PIV card. 6. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data

	4. Extract the BDB Length field of CBEFF Header. 5. Extract the SB Length field of CBEFF Header.
Expected Result(s)	1. From step 4: BDB Length field is non-zero and the recorded length matches the actual length. 2. From step 5: SB Length field is non-zero and the recorded length matches the actual length. 3. The Card Holder Facial Image object length is equal to CBEFF Header length + BDB Length + SB Length.

1974

1975 ### 9.3.2 CBEFF Header for Iris Image

1976 #### 9.3.2.1 Patron Header Version

Purpose	Validates that the CBEFF field "Patron Header Version" complies with SP80076 Table 14, "Patron Format PIV Specification"
References(s)	1. SP80076, Table 14. 2. AS05.01.02 3. AS05.01.04
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Patron Header Version field (based on its position) in CBEFF Header.
Expected Result(s)	The Patron Header Version field has a value of 0x03.

1977 #### 9.3.2.2 SBH Security Option

Purpose	Validates that the biometric data block on the PIV card is digitally signed but not encrypted.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.05

Precondition(s)	6. A valid PIV card is inserted into the contact reader. 7. A valid PC/SC connection exists between the test application and the contact reader. 8. The test application is currently connected to the card application which is accessible through card handle. 9. A valid Card Holder Iris Image object is present on the PIV card. 10. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the SBH Security Option field (based on its position) in CBEFF Header.
Expected Result(s)	The SBH Security Options field has a value of b00001101.

1978 #### 9.3.2.3 BDB Format Owner Values

Purpose	Validates that BDB Format Owner is set to a value of 0x0101 denoting ISO/IEC JTC 1/SC37 Biometrics.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.06
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Owner field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Owner field has a value of 0x0101.

1979 #### 9.3.2.4 BDB Format Type

Purpose	Validates that for optional iris image data stored on a PIV card, the BDB Format Type is set to a value of 0x0009.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.07
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the BDB Format Type field (based on its position) in CBEFF Header.
Expected Result(s)	The BDB Format Type field has a value of 0x0009.

1980 #### 9.3.2.5 Biometric Creation Date

Purpose	Validates that the creation date in the PIV Patron Format is encoded in 8 bytes using a binary representation of “YYYYMMDDhhmmssZ”.
References(s)	1. SP80076, Table 14. 2. AS05.01.02 3. AS05.01.03 4. AS05.01.08
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Creation Date field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Creation Date is in the YYYYMMDDhhmmssZ format.

1981 **9.3.2.6 Validity Period Dates**

Purpose	Validates that the Validity Period in the PIV Patron Format contains two dates encoded in the same format as expected in 9.3.2.5 above.
References(s)	1. SP80076, Table 14 2. AS05.01.02 3. AS05.01.03 4. AS05.01.09
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Validity Period field (based on its position) in CBEFF Header.
Expected Result(s)	The Validity Period contains two dates and each is encoded in the YYYYMMDDhhmmssZ format.

1982 **9.3.2.7 Biometric Type Values**

Purpose	Validates that Biometric Type has the value 0x000010.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.10
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Type field (based on its position) in CBEFF Header.
Expected Result(s)	The value of the Biometric Type field for the iris image is 0x000010.

1983 **9.3.2.8 Biometric Data Type**

Purpose	Validates that the CBEFF biometric data type encoding value shall be b010xxxxx, which corresponds to the raw biometric data.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.11
Precondition(s)	1. All required sample finger images have been recorded. 2. All templates have been generated and stored on the PIV card. 3. A valid PIV card is inserted into the contact reader. 4. A valid PC/SC connection exists between the test application and the contact reader. 5. The test application is currently connected to the card application which is accessible through card handle. 6. A valid Card Holder Iris Image object is present on the PIV card. 7. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Type field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Type field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Type field has a value of b010xxxxx.

1984 **9.3.2.9 Biometric Data Quality**

Purpose	Validates that the biometric data quality field carries valid values
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.12

Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the Biometric Data Quality field of CBEFF Header. 5. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Biometric Data Quality field (based on its position) in CBEFF Header.
Expected Result(s)	The Biometric Data Quality field is -2 and 100.

1985 **9.3.2.10 Creator Field Value**

Purpose	Validates that the Creator field in the PIV Patron Format contains 18 bytes of which the first K <= 17 bytes shall be ASCII characters, and the first of the remaining 18-K shall be a null terminator (zero).
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.13
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the Creator field (based on its position) in CBEFF Header. Start the following procedure at the first byte of the Creator field: 5. For bytes 0 to 16, check if byte represents an ASCII character

	If yes, continue. Else, check if zero. If no, fail. Else iterate through remaining bytes in field and fail if non-zero. 6. For bytes 17 through the end of the field, iterate through all bytes and fail if non-zero
Expected Result(s)	Procedure completes without failing.

1986 **9.3.2.11 FASC-N Value**

Purpose	The FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier.
References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.14
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the FASC-N field of CBEFF Header. 5. Set cardHandle := <<valid card handle>> 6. Set OID := <<Card Holder Unique Identifier (2.16.840.1.101.3.7.2.48.0)>> 7. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 8. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the FASC-N field (based on its position) in CBEFF Header.
Expected Result(s)	The FASC-N field in CBEFF header is the same as the one extracted from the CHUID.

1987 **9.3.2.12 Reserved Field Value**

Purpose	Validates that the “Reserved for Future Use” field is equal to 0x00000000.

References(s)	1. SP80076 2. AS05.01.02 3. AS05.01.03 4. AS05.01.15
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Iris Image object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, extract the first 88 bytes (CBEFF Header) and read the “Reserved for Future Use” field (based on its position) in CBEFF Header.
Expected Result(s)	The field has a value of 0x00000000.

9.4 Off-Card Comparison Fingerprint Template

9.4.1 General Record Header Conformance

Purpose	Verify that the General Record Header of the fingerprint template conforms to specifications in Table 6 of SP80076.
Reference(s)	1. SP80076, Table 6 2. AS05.01.03 3. AS05.02.01 4. AS05.02.02 5. AS05.02.03 6. AS05.02.04 7. AS05.02.05 8. AS05.02.06 9. AS05.02.07 10. AS05.02.08 11. AS05.02.09 12. AS05.02.10 13. AS05.02.11 14. AS05.02.12
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, parse the first 88 bytes (CBEFF Header) to obtain the BDB and SB lengths and then continue on to parse the General Record Header of the BDB a. Extract contents of Format Identifier. b. Extract contents of Version Number. c. Extract contents of Record Length field. d. Extract contents of CBEFF Product Identifier Owner e. Extract contents of CBEFF Product Identifier Type f. Extract contents of Capture Equipment Compliance field. g. Extract contents of Capture Equipment ID h. Extract contents of “Scanned Image in X Direction” field i. Extract contents of “Scanned Image in Y Direction” field j. Extract contents of X (horizontal) resolution. k. Extract contents of Y (vertical) resolution. l. Extract contents of “Number of Finger Views” field. m. Extract contents of Reserved Byte.

Expected Result(s)	The expected values for each of the fields parsed in Step 4 above are given below: a. Format Identifier has a value 0x4646D5200 b. Version Number has a value of 0x20323030 c. Record Length has value of $26 \leq L \leq 1574$ d. & e. Both these fields shall be > 0 (non-zero). The two most significant bytes shall identify the vendor while the least two significant bytes identifier the version number of the minutiae detection algorithm. f. Capture Equipment Compliance has a value of 1000b g. Capture Equipment ID has a value of > 0. h. Width of the Size of Scanned Image in x direction is the larger of the widths of the two input i. Height of the Size of Scanned Image in y direction is the larger of the heights of the two input images. j & k. X and Y resolution has a value of 197 l. Number of Finger Views is 2 m. Reserved Byte value is zero

9.4.2 View Header Conformance

Purpose	Verify that the View Header of the BDB conforms to specifications in Table 6 of SP80076.
Reference(s)	1. SP80076, Table 6 2. AS05.01.03 3. AS05.02.13 4. AS05.02.14 5. AS05.02.15 6. AS05.02.16
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.

Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, parse the first 88 bytes (CBEFF Header) to obtain the BDB length and then continue on to parse the View Header of the BDB a. Extract contents of View Number. b. Extract contents of Impression Type. c. Extract contents of Finger Quality. d. Extract contents of Number of Minutiae. 5. Repeat steps 4a through 4f for the second view header.
Expected Result(s)	For Step 4: a. View Number shall be 0 if there is only one minutiae record for a finger. b. Impression Type is 0 or 2. c. Finger Quality value shall be between 20, 40, 60, 80, 100, 254, or 255. d. Number of Minutiae value shall be $0 \leq M \leq 128$. For Step 5: The expected values for each of the fields are the same as in step 4.

9.4.3 Fingerprint Minutiae Data Records

Purpose	Verify that each instance of Fingerprint Minutiae data records conforms to specifications in Table 6 of SP80076.
Reference(s)	1. SP80076, Table 6 2. AS05.02.17 3. AS05.02.19
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid Card Holder Fingerprints object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0xBC, parse the first 88 bytes (CBEFF Header) to obtain the BDB length and then continue on to parse the Minutiae data instances following

	the View Header of the BDB. a. Extract contents of Minutiae Type. b. Extract contents of Extended Block Length. 5. Repeat steps 4a to 4b for each Minutiae Data instance.
Expected Result(s)	The expected values for each of the fields parsed in Step 4 and 5 above are given below: a. Minutiae Type value shall be01b, 10b, or 00b. b. Extended Data Block Length shall be 0

9.5 On-Card Comparison

9.5.1 BIT Group Template data conformance for on-card comparison

Purpose	Verify that the BIT group template follows the specifications as defined in section 5.3 and in Table 7 of SP80076.
Reference(s)	1. SP80076, Table 7 2. AS05.04.01 3. AS05.04.02 4. AS05.04.03 5. AS05.04.04 6. AS05.04.05 7. AS05.04.06 8. AS05.04.07 9. AS05.04.08 10. AS05.04.09 11. AS04.10.01
Precondition(s)	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid BIT Group Template data object is present on the PIV card. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Biometric Information Templates Group Template (2.16.840.1.101.3.7.2.16.22)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value field for tag 0x7F61 a. Extract contents of Biometric Header Template b. Extract contents of Number of BIT's in group c. Extract contents of Reference data qualifier used by VERIFY d. Extract contents of BHT biometric type e. Extract contents of Biometric subtype f. Extract contents of BHT CBEFF BDB format owner

	g. Extract contents of BHT CBEFF format type h. Extract contents of Biometric matching algorithm Tag 83
Expected Result(s)	For Step 4: a. BHT is in accordance with Table 7 in SP80076 b. number of BIT's in group (fingers) has the value of '2', one for each finger c. value of Reference data qualifier is '96' for first finger and '97' for the second finger d. value for BHT Biometric type for first and second finger has the value of '08' e. value of Biometric subtype correctly matches SP800-76 Table 8 (Column B) f. value for the BHT CBEFF BDB format owner for the first and second finger is '0101' g. value for BHT CBEFF BDB format type for the first and second finger is '00 05' h. value for Biometric matching algorithm subfield Tag 83 for the first and second finger, should not be present

1998

1999

9.6 Facial Image

9.6.1 Facial Image Header Conformance

Purpose	Verify that the Record Header of the facial image is conformant to the PIV profile presented in Table 12 of SP80076.
Reference(s)	1. SP80076, Table 12 2. AS05.03.01 3. AS05.03.02 4. AS05.03.03 5. AS05.03.04 6. AS05.03.05 7. AS05.03.06
Precondition(s)	1. All required sample facial image is stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the contents of Facial Image Header Record. 5. Extract contents of Format Identifier. 6. Extract contents of Version Number. 7. Extract contents of Number of Facial Images. 8. Extract contents of Number of Feature Points. 9. Extract contents of Record Length.
Expected Result(s)	1. From Step 5: Format Identifier has a value 0x46414300. 2. From Step 6: Version Number has a value of 0x30313000. 4. From Step 7: Number of Facial Images value is ≥1. 5. From Step 8: Number of Feature Points is ≥0. 6. From Step 9: Record Length fits within container size limits specified in 800-73.

9.6.2 Facial Image Data Conformance

Purpose	Verify that the Facial Image Instance is conformant to the PIV profile presented in Table 12 of SP80076.

Reference(s)	1. SP80076, Table 12 2. AS05.03.01 3. AS05.03.07 4. AS05.03.08 5. AS05.03.09 6. AS05.03.10
Precondition(s)	1. All required sample facial image is stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the contents of Facial Image Instance Record 5. Extract contents of Facial Image Type. 6. Extract contents of Image Data Type. 7. Extract contents of Image Color Space. 8. Extract contents of Source Type.
Expected Result(s)	1. Step 5: Facial Image Type is 1. 2. Step 6: Image Data Type is 0 or 1. 3. Step 7: Image Color Space is 1. 4. Step 8: Source Type is 2 or 6.

9.7 Iris Image

9.7.1 Iris Image Profile

Purpose	Verify that the data recorded of the iris image is conformant to the PIV profile presented in Table 9 of SP80076-2.
Reference(s)	1. SP80076-2, Table 9 2. AS05.05.01 3. AS05.05.02 4. AS05.05.03 5. AS05.05.04 6. AS05.05.05 7. AS05.05.06 8. AS05.05.07

Precondition(s)	1. All required sample iris images are stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the contents of Iris General Header. 5. Extract contents of Format identifier. 6. Extract contents of Version number. 7. Extract contents of Length of record 8. Extract contents of Number of iris representations 9. Extract contents of Certification flag 10. Extract contents of Number of eyes represented.
Expected Result(s)	Compare profile with SP800-76 Table 9 to validate appropriate information is present to include: 1. Step 5: Format identifier is 0x49495200. 2. Step 6: Version number is 0x30323000. 3. Step 7: Length of record is 3K for each iris. 4. Step 8: Number of iris representation is 1 or 2. 5. Step 9: Certification flag is 0x00. 6. Step 10: Number of eyes represented is 1 or 2

2006

2007 ### 9.7.2 Iris Image Data Conformance

Purpose	Verify that the iris instance is conformant to the PIV profile presented in Table 12 of SP80076.
Reference(s)	1. SP80076, Table 12 2. AS05.05.08 3. AS05.05.09 4. AS05.05.10 5. AS05.05.11 6. AS05.05.12 7. AS05.05.13 8. AS05.05.14 9. AS05.05.15 10. AS05.05.16 11. AS05.05.17 12. AS05.05.18

Precondition(s)	1. All required sample iris images are stored on the PIV card. 2. A valid PIV card is inserted into the contact reader. 3. A valid PC/SC connection exists between the test application and the contact reader. 4. The test application is currently connected to the card application which is accessible through card handle. 5. Security conditions to read the object are met.
Test Scenario	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the contents of Iris representation header and image data 5. Extract contents of Capture date and time. 6. Extract contents of Representation number. 7. Extract contents of Eye label. 8. Extract contents of Image type. 9. Extract contents of Image format. 10. Extract contents of Iris image properties bit field. 11. Extract contents of Image width, W. 12. Extract contents of Image height, H. 13. Extract contents of Bit depth. 14. Extract contents of Iris centre, lowest X. 15. Extract contents of Iris centre, highest X. 16. Extract contents of Iris centre, lowest Y. 17. Extract contents of Iris centre, highest Y. 18. Extract contents of Iris diameter, lowest. 19. Extract contents of Iris diameter, highest.

Expected Result(s)	Compare profile with SP800-76 Table 9 to validate appropriate information is present to include: 1. Step 5: Capture date and time is 2011 onwards. 2. Step 6: Representation number is 1 and then, optionally, 2. 3. Step 7: Eye label is 1 or 2. 4. Step 8: Image type is 7. 5. Step 9: Image format is 10 = 0x0A 6. Step 10: Iris image properties bit field is (Bits 1-2: 01 or 10), (Bits 3-4: 01 or 10), (Bits 5-6: 01 and scan type shall be progressive), and (Bits 7-8: 01 and the compression history shall be "none"). 7. Step 11: Image width, W is $288 \leq W \leq 448$. 8. Step 12: Image height, H is $216 \leq H \leq 336$. 9. Step 13: Bit depth is 8. 10. Step 14: Iris centre, lowest X is (W/2 for W odd, else). 11. Step 15: Iris centre, highest X is (W/2+1 for W even). 12. Step 16: Iris centre, lowest Y is (H/2 for H odd, else). 13. Step 17: Iris centre, highest Y is (W/2+1 for H even). 14. Step 18: Iris diameter, lowest is $D \geq 160$. 15. Step 19: Iris diameter, highest is $D \leq 280$.

2008

2009

10. Signed Data Elements Test Assertions

10.1 Card Holder Unique Identifier (CHUID)

10.1.1 Signature Block Contents

10.1.1.1 Verify presence of CMS SignedData asymmetric digital signature

Purpose	Confirms that the CHUID buffer contains an asymmetric digital signature implemented as a SignedData type in accordance with the Cryptographic Message Syntax as defined in RFC 5652.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.01 3. AS06.01.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the obtained CHUID and extract the contents from the asymmetric digital signature field (i.e. Tag 0x3E) 5. Process the contents of the digital signature
Expected Result(s)	The CHUID buffer contains an asymmetric digital signature that is implemented as a SignedData type and is encoded as a CMS external signature according to RFC 5652.

10.1.1.2 Verify version in SignedData

Purpose	Confirms that the version of the SignedData content type is 3.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.03
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the version field contents from the asymmetric signature of the CHUID
Expected Result(s)	The value of the version field of the SignedData is 3.

2016

2017 **10.1.1.3 Verify digest Algorithm in SignedData**

Purpose	Confirm that the digestAlgorithms field of the SignedData content type is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.01.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the digestAlgorithms field contents from the asymmetric digital signature of the CHUID 5. Extract the certificates field contents from the asymmetric signature of the CHUID 6. From the certificate obtained, extract the subjectPublicKeyInfo->subjectPublicKey 7. Compute the size of the signer’s public key 8. Match the digest algorithm with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the CHUID. 9.
Expected Result(s)	The digestAlgorithms field value of the SignedData is in accordance with Table 3-2 of SP80078.

2018

2019

10.1.1.4 Verify contents of encapContentInfo

Purpose	Confirms that the eContentType of the encapContentInfo is id-PIV-CHUIDSecurityObject and the eContent field of the encapContentInfo is omitted.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.05 3. AS06.01.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract and parse the encapContentInfo field contents from the asymmetric digital signature of the CHUID
Expected Result(s)	The eContent field has been omitted and the eContentType asserts id-piv-CHUIDSecurityObject in encapContentInfo.

10.1.1.5 Verify crls field omission

Purpose	Confirm that the crls field from the SignedData content type is omitted.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the crls field contents from the asymmetric digital signature of the CHUID

Expected Result(s)	The crls field is omitted from the SignedData.

10.1.1.6 Verify contents of signerInfos

Purpose	Confirms that the signerInfos in the SignedData is populated with a single SignerInfo.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfos field contents from the asymmetric digital signature of the CHUID
Expected Result(s)	The signerInfos field in the SignedData contains a single SignerInfo.

10.1.1.7 Verify Signer Identifier in SignerInfo

Purpose	Confirms that the SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice.
Reference(s)	1. SP80073 Part 1, Section 3.1.2.1 2. AS06.01.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the signerInfo->signerIdentifier field contents from the asymmetric signature of the CHUID 5. Extract the certificates field contents from the asymmetric signature of the CHUID 6. Extract the issuer and serialNumber fields from the certificate obtained in the previous step
Expected Result(s)	The SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice and it corresponds to the issuer and serialNumber fields found in the X.509 certificate of the signer.

10.1.1.8 Verify Digest Algorithm in SignerInfo

Purpose	Confirm that the digestAlgorithm field of the SignerInfo is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.01.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->digestAlgorithm field contents from the asymmetric digital signature of the CHUID 5. Extract the certificate field contents from the asymmetric signature of the CHUID 6. From the certificate obtained, extract the subjectPublicKeyInfo->subjectPublicKey 7. Compute the size of the signer’s public key 8. Match the digest algorithm with the Table 3-2 of SP80078 based on the public key algorithm and size used to sign the CHUID 9. Extract the digestAlgorithms field contents from the asymmetric digital signature of the CHUID i.e. SignedData

Expected Result(s)	The digestAlgorithm field value of the SignerInfo is in accordance with Table 3-2 of SP80078 and it matches the value present in the digestAlgorithms field of the SignedData.

2029

2030 **10.1.1.9 Verify message digest signed attribute in SignerInfo**

Purpose	Confirms that the signedAttrs of the SignerInfo includes a message digest attribute containing the hash computed over the concatenated contents of the CHUID, excluding the asymmetric signature field and the Buffer Length field (if present).
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the asymmetric signature field of the CHUID to locate the message digest attribute (OID=1.2.840.113549.1.9.4) and its corresponding attribute value 5. Extract the SignerInfo->digestAlgorithm field contents from the asymmetric digital signature of the CHUID 6. Using the digest Algorithm obtained in the previous step, calculate the hash of the concatenated contents of the CHUID, excluding the asymmetric digital signature field and the Buffer Length field, if present.
Expected Result(s)	The value of the hash obtained from the message digest attribute of the signedAttrs of the SignerInfo is identical to that obtained after hashing the concatenated contents of the CHUID, excluding the asymmetric digital signature field and the Buffer Length field, if present.

2031

2032 **10.1.1.10 Verify PIV signer distinguished name**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivSigner-DN attribute containing the subject name that appears in the X.509 certificate for the entity that signed the CHUID.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.13

Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the asymmetric signature field of the CHUID to locate the pivSigner-DN attribute (OID=2.16.840.1.101.3.6.5) 5. Extract the certificates field contents from the asymmetric signature of the CHUID. 6. Extract the subject DN from the certificate obtained in the previous step
Expected Result(s)	The value of the subject DN obtained from the certificate in the certificates field in the SignedData is identical to that obtained from the pivSigner-DN attribute of the signedAttrs of the SignerInfo.

2033

2034 **10.1.1.11 Verify signature algorithm in SignerInfo**

Purpose	Confirms that the signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.
Reference(s)	1. SP80078 , Section 3.2.1 2. AS06.01.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the CHUID's SignerInfo->signatureAlgorithm field contents.

Expected Result(s)	The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

10.1.1.12 Verify digital signature

Purpose	Confirms that the signature in the SignerInfo corresponds to the signed CHUID
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.01.07 3. AS06.01.15
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the certificates field contents from the asymmetric signature of the CHUID. 5. Extract the asymmetric signature contents from the CHUID 6. Using the certificate extracted, verify the SignedData located in the asymmetric signature field of the CHUID.
Expected Result(s)	The certificates field in the SignedData contains a single certificate that can be used to verify the digital signature in the SignerInfo.

10.2 Off-Card Comparison Biometric Fingerprint

10.2.1 Signature Block Contents

10.2.1.1 Verify presence of CMS SignedData asymmetric digital signature

Purpose	Confirms that the CBEFF_SIGNATURE_BLOCK is implemented as a SignedData type and is encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652
Reference(s)	1. SP80076 2. AS06.02.01 3. AS06.02.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. Security conditions to read the object are met.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the obtained biometric and extract the contents from the asymmetric digital signature field (i.e. from the CBEFF_SIGNATURE_BLOCK) 5. Process the contents of the digital signature
Expected Result(s)	The CBEFF_SIGNATURE_BLOCK is present in the biometric CBEFF structure containing an asymmetric digital signature that is implemented as a SignedData type according to RFC 5652.

10.2.1.2 Verify version in SignedData

Purpose	Confirms that the version of the SignedData content type is v3.
Reference(s)	1. SP80076 2. AS06.02.03
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the version field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present)
Expected Result(s)	The value of the version field of the SignedData is v3.

2043

2044 **10.2.1.3 Verify digest Algorithm in SignedData**

Purpose	Confirm that the digestAlgorithms field of the SignedData content type is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078 , Section 3.2.1 2. AS06.02.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist (step 5), then call pivGetData with the OID for the CHUID and extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer’s public key 9. Match the digest algorithm obtained from step 4 with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the fingerprint biometric

Expected Result(s)	The digestAlgorithms field value of the SignedData is in accordance with Table 3-2 of SP80078.

2045

2046 #### 10.2.1.4 Verify contents of encapContentInfo

Purpose	Confirms that the eContentType of the encapContentInfo is id-PIV-biometricObject and the eContent field of the encapContentInfo is omitted.
Reference(s)	1. SP80076 2. AS06.02.05 3. AS06.02.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract and parse the encapContentInfo field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The eContent field has been omitted and the eContentType asserts id-piv-biometricObject in encapContentInfo.

2047

2048 #### 10.2.1.5 Verify crls field omission

Purpose	Confirm that the crls field from the SignedData content type is omitted.
Reference(s)	1. SP80076 2. AS06.02.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle

	• (IN) OID • (OUT) data 4. Extract the crls field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The crls field is omitted from the SignedData.

2049

2050 **10.2.1.6 Verify contents of signerInfos**

Purpose	Confirms that the signerInfos in the SignedData is populated with a single SignerInfo.
Reference(s)	1. SP80076 2. AS06.02.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfos field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The signerInfos field in the SignedData contains a single SignerInfo.

2051

2052 **10.2.1.7 Verify Signer Identifier in SignerInfo**

Purpose	Confirms that the SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice.
Reference(s)	1. SP80076 2. AS06.02.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>>

	2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the signerInfo->SignerIdentifier field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. Extract the issuer and serialNumber fields from the certificate obtained in the previous step
Expected Result(s)	The SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice which corresponds to the issuer and serialNumber fields found in the X.509 certificate of the signer.

2053

2054 #### 10.2.1.8 Verify Digest Algorithm in SignerInfo

Purpose	Confirm that the digestAlgorithm field of the SignerInfo is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.02.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->digestAlgorithm field from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist (step 5), then extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field),

	extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer's public key 9. Match the digest algorithm obtained from step 4 with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the biometric fingerprint 10. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The digestAlgorithm field value of the SignerInfo is in accordance with Table 3-2 of SP80078 and it matches the value present in the digestAlgorithms field of the SignedData.

2055

2056 **10.2.1.9 Verify message digest signed attribute in SignerInfo**

Purpose	Confirms that the signedAttrs of the SignerInfo includes a message digest attribute containing the hash computed over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.02.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the message digest attribute (OID=1.2.840.113549.1.9.4) and its corresponding attribute value 5. Extract the SignerInfo->digestAlgorithm field contents from the CBEFF_SIGNATURE_BLOCK 6. Using the digest Algorithm obtained in the previous step, compute the hash over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Expected Result(s)	The value of the hash obtained from the message digest attribute of the signedAttrs of the SignerInfo is identical to that obtained after hashing the concatenated contents of the Fingerprint Object buffer, excluding the CBEFF_SIGNATURE_BLOCK.

2057

2058 **10.2.1.10 Verify PIV signer distinguished name**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivSigner-DN attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometrics.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.02.13
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivSigner-DN attribute (OID=2.16.840.1.101.3.6.5) 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. Extract the subject DN from the certificate obtained in the previous step
Expected Result(s)	The value of the subject DN obtained from the certificate in the certificates field in the SignedData is identical to that obtained from the pivSigner-DN attribute of the signedAttrs of the SignerInfo.

2059

2060 **10.2.1.11 Verify FASC-N**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivFASC-N attribute whose value matches the value of the FASC-N in the CHUID of the PIV card.
Reference(s)	1. SP 80076 2. AS06.02.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application

	which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivFASC-N attribute (OID=2.16.840.1.101.3.6.6) and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the FASC-N
Expected Result(s)	A pivFASC-N attribute exists in the signedAttrs of the SignerInfo and its value matches the FASC-N present in the CHUID of the PIV card.

10.2.1.12 Verify signature algorithm in SignerInfo

Purpose	Confirms that the signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP 80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.02.15
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->signatureAlgorithm field
Expected Result(s)	The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP 80078.

2063

2064 #### 10.2.1.13 Verify digital signature

Purpose	Confirms that the signature in the SignerInfo corresponds to the signed biometric
Reference(s)	1. SP 80073, Section 3.1.2.1 2. AS06.02.07 3. AS06.02.16
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK. If this field is omitted then extract the certificate from the CHUID signature 4. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 5. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 6. Extract the digital signature string from CBEFF_SIGNATURE_BLOCK. 7. Using the certificate extracted either from the CBEFF_SIGNATURE_BLOCK or the CHUID asymmetric signature, verify the signature on the biometric
Expected Result(s)	The certificates field in the SignedData contains a single certificate that can be used to verify the digital signature in the SignerInfo. If the certificates field is omitted, then the certificates field of the SignedData

	for the CHUID contains the certificate that can be used to verify the digital signature.

2065

2066 #### 10.2.1.14 Verify entryUUID

Purpose	Confirms that the signedAttrs of the SignerInfo includes the entryUUID (OID = 1.3.6.1.1.16.4) attribute and that it is the same value as the Card UUID in the GUID data element of the PIV card's CHUID data object.
Reference(s)	1. SP80076 2. AS06.02.17
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder fingerprint object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Fingerprints (2.16.840.1.101.3.7.2.96.16)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the entryUUID attribute and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the GUID
Expected Result(s)	An entryUUID (OID = 1.3.6.1.1.16.4) attribute exists in the signedAttrs of the SignerInfo and its value matches the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

2067

2068

10.3 Biometric Facial Image

10.3.1 Signature Block Contents

10.3.1.1 Verify presence of CMS SignedData asymmetric digital signature

Purpose	Confirms that the CBEFF_SIGNATURE_BLOCK is implemented as a SignedData type and is encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652
Reference(s)	1. SP80076 2. AS06.03.01 3. AS06.03.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. Security conditions to read the object are met.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the obtained biometric and extract the contents from the asymmetric digital signature field (i.e. from the CBEFF_SIGNATURE_BLOCK) 5. Process the contents of the digital signature
Expected Result(s)	The CBEFF_SIGNATURE_BLOCK is present in the biometric CBEFF structure containing an asymmetric digital signature that is implemented as a SignedData type according to RFC 5652.

10.3.1.2 Verify version in SignedData

Purpose	Confirms that the version of the SignedData content type is v3.
Reference(s)	1. SP 80076 2. AS06.03.03
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the version field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present)
Expected Result(s)	The value of the version field of the SignedData is v3.

10.3.1.3 Verify digest Algorithm in SignedData

Purpose	Confirm that the digestAlgorithms field of the SignedData content type is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.03.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK (step 5)does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer’s public key 9. Match the digest algorithm obtained from step 4 with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the biometric facial image (from step 8)

Expected Result(s)	The digestAlgorithms field value of the SignedData is in accordance with Table 3-2 of SP80078.

2076

2077 #### 10.3.1.4 Verify contents of encapContentInfo

Purpose	Confirms that the eContentType of the encapContentInfo is id-PIV-biometricObject and the eContent field of the encapContentInfo is omitted.
Reference(s)	1. SP80076 2. AS06.03.05 3. AS06.03.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract and parse the encapContentInfo field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The eContent field has been omitted and the eContentType asserts id-piv-biometricObject in encapContentInfo.

2078

2079 #### 10.3.1.5 Verify crls field omission

Purpose	Confirm that the crls field from the SignedData content type is omitted.
Reference(s)	1. SP80076 2. AS06.03.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle

	• (IN) OID • (OUT) data 4. Extract the crls field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The crls field is omitted from the SignedData.

2080

2081 #### 10.3.1.6 Verify contents of signerInfos

Purpose	Confirms that the signerInfos in the SignedData is populated with a single SignerInfo.
Reference(s)	1. SP80076 2. AS06.03.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfos field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The signerInfos field in the SignedData contains a single SignerInfo.

2082

2083 #### 10.3.1.7 Verify Signer Identifier in SignerInfo

Purpose	Confirms that the SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice.
Reference(s)	1. SP80076 2. AS06.03.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>>

	2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the signerInfo->SignerIdentifier field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. Extract the issuer and serialNumber fields from the certificate obtained in the previous step
Expected Result(s)	The SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice which corresponds to the issuer and serialNumber fields found in the X.509 certificate of the signer.

2084

2085 #### 10.3.1.8 Verify Digest Algorithm in SignerInfo

Purpose	Confirm that the digestAlgorithm field of the SignerInfo is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.03.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->digestAlgorithm field from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist(step 5), then extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer's public key 9. Match the digest algorithm obtained from step 4 to an entry in Table 3-2 of SP80078 based on the public key algorithm and size used to sign the biometric facial image 10. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The digestAlgorithm field value of the SignerInfo is in accordance with Table 3-2 of SP80078 and it matches the value present in the digestAlgorithms field of the SignedData.

10.3.1.9 Verify message digest signed attribute in SignerInfo

Purpose	Confirms that the signedAttrs of the SignerInfo includes a message digest attribute containing the hash computed over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.03.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the message digest attribute (OID=1.2.840.113549.1.9.4) and its corresponding attribute value 5. Extract the SignerInfo->digestAlgorithm field contents from the CBEFF_SIGNATURE_BLOCK 6. Using the digest Algorithm obtained in the previous step, compute the hash over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Expected Result(s)	The value of the hash obtained from the message digest attribute of the signedAttrs of the SignerInfo is identical to that obtained after hashing the concatenated contents of the CBEFF_HEADER and the STD_BIOMETRIC_RECORD

2088

2089 **10.3.1.10 Verify PIV signer distinguished name**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivSigner-DN attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometrics.
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.03.13
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivSigner-DN attribute (OID=2.16.840.1.101.3.6.5) 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. Extract the subject DN from the certificate obtained in the previous step
Expected	The value of the subject DN obtained from the certificate in the certificates field in the SignedData is identical to that obtained from the

Result(s)	pivSigner-DN attribute of the signedAttrs of the SignerInfo.

2090

2091 #### 10.3.1.11 Verify FASC-N

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivFASC-N attribute whose value matches the value of the FASC-N in the CHUID of the PIV card.
Reference(s)	1. SP80076 2. AS06.03.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivFASC-N attribute (OID=2.16.840.1.101.3.6.6) and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the FASC-N
Expected Result(s)	A pivFASC-N attribute exists in the signedAttrs of the SignerInfo and its value matches the FASC-N present in the CHUID of the PIV card.

2092

2093 #### 10.3.1.12 Verify signature algorithm in SignerInfo

Purpose	Confirms that the signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP 80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.03.15

Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data **4.** Extract the SignerInfo->signatureAlgorithm field contents.
Expected Result(s)	The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP 80078.

10.3.1.13 Verify digital signature

Purpose	Confirms that the signature in the SignerInfo corresponds to the signed biometric
Reference(s)	1. SP80073, Section 3.1.2.1 2. AS06.03.07 3. AS06.03.16
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image is present on the PIV card. 5. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK. If this field is omitted then extract the certificate from the CHUID signature 4. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 5. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data a. Extract the certificates field contents from the asymmetric signature of the CHUID. 6. Using the certificate extracted either from the CBEFF_SIGNATURE_BLOCK or the CHUID asymmetric signature, verify the signature on the biometric
Expected Result(s)	The certificates field in the SignedData contains a single certificate that can be used to verify the digital signature in the SignerInfo. If the certificates field is omitted, then the certificates field of the SignedData for the CHUID contains the certificate that can be used to verify the digital signature.

2096

2097 #### 10.3.1.14 Verify entryUUID

Purpose	Confirms that the signedAttrs of the SignerInfo includes the entryUUID (OID = 1.3.6.1.1.16.4) attribute and that it is the same value as the Card UUID in the GUID data element of the PIV card's CHUID data object.
Reference(s)	1. SP80076 2. AS06.03.17
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder facial image object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Facial Image (2.16.840.1.101.3.7.2.96.48)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the entryUUID attribute and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the GUID
Expected Result(s)	An entryUUID (OID = 1.3.6.1.1.16.4) attribute exists in the signedAttrs of the SignerInfo and its value matches the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

10.4 Security Object

10.4.1 Data Integrity

10.4.1.1 Verify integrity of data element hashes

Purpose	Confirms the integrity of the hashes of the data elements of the PIV card present in the security object.
Reference(s)	1. SP80073, Section 1.8.5 2. AS06.04.01
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the card. 5. Valid objects whose hashes are referenced in the security object are present on the card. 6. Security conditions to read the object are met.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Identify the various data elements that are part of the security object by parsing the Mapping of Data Group (DG) to ContainerID (i.e. TAG 0xBA) 5. Extract the ldsSecurityObject from the eContent field of the Security Object Asymmetric Signature (i.e. TAG 0xBB) 6. Call pivGetData for all those data elements that are present in the mapping obtained from step 4 7. Compute the hash for each data element and verify that it matches the hash value present in the ldsSecurityObject
Expected Result(s)	The actual hash of the data elements on the PIV card are identical to their corresponding hash values present in the security object.

10.4.2 Signature Block Contents

10.4.2.1 Verify presence of CMS SignedData asymmetric digital signature

Purpose	Confirms that the security object data object contains an asymmetric digital signature, implemented as a SignedData type in accordance with RFC 5652.

Reference(s)	1. AS06.04.02 2. AS06.04.03
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the obtained security object and extract the contents from the asymmetric digital signature field (i.e. TAG 0xBB) 5. Process the contents of the digital signature
Expected Result(s)	The security object is present in the security object data object and contains an asymmetric digital signature that is implemented as a SignedData type according to RFC 5652.

2106

2107 #### 10.4.2.2 Verify version in SignedData

Purpose	Confirms that the version of the SignedData content type is v3.
Reference(s)	1. AS06.04.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the version field contents from the asymmetric signature of the Security Object (i.e. TAG 0xBB)
Expected Result(s)	The value of the version field of the SignedData is v3.

2108

2109 **10.4.2.3 Verify digest Algorithm in SignedData**

Purpose	Confirm that the digestAlgorithms field of the SignedData content type is in accordance with Tables 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.04.05
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the digestAlgorithms field contents from the Security Object 5. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 6. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 7. Extract the certificates field contents from the asymmetric signature of the CHUID 8. From the certificate obtained (from the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 9. Compute the size of the signer's public key 10. Match the digest algorithm obtained from step 4 with Table 3-6 of SP80078 10. Match the digest algorithm obtained from step 4 to an entry of Table 3-2 of SP80078 based on the public key algorithm (step 8) and size (step 9)
Expected Result(s)	The digestAlgorithms field value of the SignedData is in accordance with Tables 3-2 of SP80078.

2110

2111 **10.4.2.4 Verify contents of encapContentInfo**

Purpose	Confirms that the eContentType of the encapContentInfo is id-icao-ldsSecurityObject and the eContent field of the encapContentInfo

	contains the contents of the ldsSecurity object.
Reference(s)	1. AS06.04.06 2. AS06.04.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract and parse the encapContentInfo field contents from the security object
Expected Result(s)	The eContent field contains a correctly formatted ldsSecurityobject and the eContentType asserts id-icao-ldsSecurityObject in encapContentInfo.

2112

2113 **10.4.2.5 Verify certificates field omission**

Purpose	Confirm that the certificates field from the SignedData content type is omitted.
Reference(s)	1. AS06.04.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card..
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the certificates field contents from the security object
Expected Result(s)	The certificates field is omitted from the SignedData.

2114

2115 #### 10.4.2.6 Verify Digest Algorithm in SignerInfo

Purpose	Confirm that the digestAlgorithm field of the SignerInfo is in accordance with Tables 3-6 and 3-2 of SP80078.
Reference(s)	1. AS06.04.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->digestAlgorithm field from the security object 5. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 6. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 7. Extract the certificates field contents from the asymmetric signature of the CHUID 8. From the certificate obtained (from the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey **9.** Compute the size of the signer's public key 10. Match the digest algorithm obtained from step 4 with Table 3-6 of SP80078 11. Match the digest algorithm obtained from step 4 to an entry in Table 3-2 based on the public key algorithm (step 8) and size (step 9) used to sign the security object 11. Extract the digestAlgorithms field contents from the security object's SignedData
Expected Result(s)	The digestAlgorithm field value of the SignerInfo is in accordance with Tables 3-6 and 3-2 of SP80078 and it matches the value present in the digestAlgorithms field of the SignedData.

2116

2117

2118 #### 10.4.2.7 Verify signature algorithm in SignerInfo

Purpose	Confirms that the signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.04.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>> 2. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 3. From the signature block (TAG 0xBB), match the SignerInfo->signatureAlgorithm field contents to an entry in Table 3-3 of SP80078
Expected Result(s)	The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

2119

2120 #### 10.4.2.8 Verify digital signature

Purpose	Confirms that the signature in the SignerInfo corresponds to the signed security object and that it is it signed with the certificate that is used to sign the CHUID.
Reference(s)	1. SP80073 2. AS06.04.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid security object is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Security Object (2.16.840.1.101.3.7.2.144.0)>>

	3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the contents of the Security Object asymmetric signature (TAG 0xBB) 5. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 6. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 7. Extract the certificates field contents from the asymmetric signature of the CHUID. 8. Using the certificate extracted from the CHUID asymmetric signature block, verify the signature of the security object
Expected Result(s)	The certificates field of the SignedData for the CHUID contains the certificate that can be used to verify the digital signature on the security object.

10.5 Biometric Iris

10.5.1 Signature Block Contents

10.5.1.1 Verify presence of CMS SignedData asymmetric digital signature

Purpose	Confirms that the CBEFF_SIGNATURE_BLOCK is implemented as a SignedData type and is encoded as a Cryptographic Message Syntax external digital signature as defined in RFC 5652
Reference(s)	1. SP80076 2. AS06.05.01 3. AS06.05.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. Security conditions to read the object are met.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the obtained biometric and extract the contents from the asymmetric digital signature field (i.e. from the CBEFF_SIGNATURE_BLOCK) 5. Process the contents of the digital signature
Expected Result(s)	The CBEFF_SIGNATURE_BLOCK is present in the biometric CBEFF structure containing an asymmetric digital signature that is implemented as a SignedData type according to RFC 5652.

10.5.1.2 Verify version in SignedData

Purpose	Confirms that the version of the SignedData content type is v3.
Reference(s)	1. SP80076 2. AS06.05.03
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the version field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present)
Expected Result(s)	The value of the version field of the SignedData is v3.

2128

2129 #### 10.5.1.3 Verify digest Algorithm in SignedData

Purpose	Confirm that the digestAlgorithms field of the SignedData content type is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078 , Section 3.2.1 2. AS06.05.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. **4.** A valid card holder iris image is present on the PIV card. 5. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist (step 5), then call pivGetData with the OID for the CHUID and extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer's public key 9. Match the digest algorithm obtained from step 4 with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the iris biometric
Expected Result(s)	The digestAlgorithms field value of the SignedData is in accordance with Table 3-2 of SP80078.

10.5.1.4 Verify contents of encapContentInfo

Purpose	Confirms that the eContentType of the encapContentInfo is id-PIV-biometricObject and the eContent field of the encapContentInfo is omitted.
Reference(s)	1. SP80076 2. AS06.05.05 3. AS06.05.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract and parse the encapContentInfo field contents

	from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The eContent field has been omitted and the eContentType asserts id-piv-biometricObject in encapContentInfo.

2132

2133 #### 10.5.1.5 Verify crls field omission

Purpose	Confirm that the crls field from the SignedData content type is omitted.
Reference(s)	1. SP80076 2. AS06.05.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the crls field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The crls field is omitted from the SignedData.

2134

2135 #### 10.5.1.6 Verify contents of signerInfos

Purpose	Confirms that the signerInfos in the SignedData is populated with a single SignerInfo.
Reference(s)	1. SP80076 2. AS06.05.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract the SignerInfos field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The signerInfos field in the SignedData contains a single SignerInfo.

2136

2137 #### 10.5.1.7 Verify Signer Identifier in SignerInfo

Purpose	Confirms that the SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice.
Reference(s)	1. SP80076 2. AS06.05.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the signerInfo->SignerIdentifier field contents from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the asymmetric signature of the CHUID 7. Extract the issuer and serialNumber fields from the certificate obtained in the previous step
Expected Result(s)	The SignerIdentifier in the SignerInfo uses the issuerAndSerialNumber choice which corresponds to the issuer and serialNumber fields found in the X.509 certificate of the signer.

2138

2139 #### 10.5.1.8 Verify Digest Algorithm in SignerInfo

Purpose	Confirm that the digestAlgorithm field of the SignerInfo is in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.05.11

Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the SignerInfo->digestAlgorithm field from the CBEFF_SIGNATURE_BLOCK 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK (if present) 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist (step 5), then extract the certificates field contents from the asymmetric signature of the CHUID 7. From the certificate obtained (either from the CBEFF_SIGNATURE_BLOCK or the CHUID signature field), extract the subjectPublicKeyInfo->subjectPublicKey 8. Compute the size of the signer's public key 9. Match the digest algorithm obtained from step 4 with Table 3-2 of SP80078 based on the public key algorithm and size used to sign the biometric iris 12. Extract the digestAlgorithms field contents from the CBEFF_SIGNATURE_BLOCK
Expected Result(s)	The digestAlgorithm field value of the SignerInfo is in accordance with Table 3-2 of SP80078 and it matches the value present in the digestAlgorithms field of the SignedData.

10.5.1.9 Verify message digest signed attribute in SignerInfo

Purpose	Confirms that the signedAttrs of the SignerInfo includes a message digest attribute containing the hash computed over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Reference(s)	**1.** SP80073, Section 3.1.2.1 2. AS06.05.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris

	(2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the message digest attribute (OID=1.2.840.113549.1.9.4) and its corresponding attribute value 5. Extract the SignerInfo->digestAlgorithm field contents from the CBEFF_SIGNATURE_BLOCK 6. Using the digest Algorithm obtained in the previous step, compute the hash over the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD
Expected Result(s)	The value of the hash obtained from the message digest attribute of the signedAttrs of the SignerInfo is identical to that obtained after hashing the concatenated contents of the Iris Object buffer, excluding the CBEFF_SIGNATURE_BLOCK.

2142

2143 **10.5.1.10 Verify PIV signer distinguished name**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivSigner-DN attribute containing the subject name that appears in the X.509 certificate for the entity that signed the biometrics.
Reference(s)	**1.** SP80073, Section 3.1.2.1 2. AS06.05.13
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivSigner-DN attribute (OID=2.16.840.1.101.3.6.5) 5. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK 6. If a certificate extracted from the certificates field of the CBEFF_SIGNATURE_BLOCK does not exist, then extract the certificates field contents from the

	asymmetric signature of the CHUID 7. Extract the subject DN from the certificate obtained in the previous step
Expected Result(s)	The value of the subject DN obtained from the certificate in the certificates field in the SignedData is identical to that obtained from the pivSigner-DN attribute of the signedAttrs of the SignerInfo.

2144

2145 **10.5.1.11 Verify FASC-N**

Purpose	Confirms that the signedAttrs of the SignerInfo includes the pivFASC-N attribute whose value matches the value of the FASC-N in the CHUID of the PIV card.
Reference(s)	1. SP80076 2. AS06.05.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Fingerprints (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the pivFASC-N attribute (OID=2.16.840.1.101.3.6.6) and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the FASC-N
Expected Result(s)	A pivFASC-N attribute exists in the signedAttrs of the SignerInfo and its value matches the FASC-N present in the CHUID of the PIV card.

2146

2147 **10.5.1.12 Verify signature algorithm in SignerInfo**

Purpose	Confirms that the signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption

	OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS06.05.15
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. A valid CHUID object is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data **4.** Extract the SignerInfo->signatureAlgorithm field
Expected Result(s)	The signatureAlgorithm field specified in the SignerInfo field for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm shall be in accordance with Table 3-3 of SP80078.

10.5.1.13 Verify digital signature

Purpose	Confirms that the signature in the SignerInfo corresponds to the signed biometric
Reference(s)	**1.** SP80073, Section 3.1.2.1 2. AS06.05.07 3. AS06.05.16
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image is present on the PIV card. 5. A valid CHUID object is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Cardholder Iris Images (2.16.840.1.101.3.7.2.16.21)>> 3. Extract the certificates field contents from the CBEFF_SIGNATURE_BLOCK. If this field is omitted then extract the certificate from the CHUID signature 4. Set OID := <<CHUID (2.16.840.1.101.3.7.2.48.0)>> 5. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 6. Extract the digital signature string from CBEFF_SIGNATURE_BLOCK. 7. Using the certificate extracted either from the CBEFF_SIGNATURE_BLOCK or the CHUID asymmetric signature, verify the signature on the biometric
Expected Result(s)	The certificates field in the SignedData contains a single certificate that can be used to verify the digital signature in the SignerInfo. If the certificates field is omitted, then the certificates field of the SignedData for the CHUID contains the certificate that can be used to verify the digital signature.

10.5.1.14 Verify entryUUID

Purpose	Confirms that the signedAttrs of the SignerInfo includes the entryUUID (OID = 1.3.6.1.1.16.4) attribute and that it is the same value as the Card UUID in the GUID data element of the PIV card's CHUID data object.
Reference(s)	1. SP80076 2. AS06.05.17
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A valid card holder iris image object is present on the PIV card. 5. A valid CHUID object is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Holder Iris Image (2.16.840.1.101.3.7.2.16.21)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the SignerInfo-> signedAttrs field contents from the CBEFF_SIGNATURE_BLOCK to locate the entryUUID attribute and its corresponding attribute value 5. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the GUID
Expected Result(s)	An entryUUID (OID = 1.3.6.1.1.16.4) attribute exists in the signedAttrs of the SignerInfo and its value matches the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

11. PKI Certificate Profile Test Assertions

11.1 PIV Authentication Certificate

11.1.1 SP 800-78 Algorithms Conformance

11.1.1.1 Verify signature algorithm

Purpose	Confirms that the proper signature algorithm has been used to sign the certificate as specified in Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.01.01 3. AS07.01.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract signature->algorithm field value from the certificate
Expected Result(s)	1. From Step 4: The signatureAlgorithm value is in accordance with Table 3-3 of SP80078 2. From Step 4: If the algorithm value is id-RSASSA-PSS, verify that the signature->parameters field is populated with SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

11.1.1.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 3-4 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.01.03

	3. AS07.01.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Match the algorithm value to the Table 3-4 of SP80078 6. If the algorithm is Elliptic Curve, ensure that one of the approved curves is used and the OID is populated in the subjectPublicKeyInfo->algorithm->parameters->namedCurve field from Table 3-5 of SP80078. The parameters field may contain NULL to indicate that parameters are inherited. Note: - If the RSA algorithm is used, the subjectPublicKeyInfo->algorithm->parameters field shall be NULL
Expected Result(s)	The PIV authentication key is generated using an allowed asymmetric key algorithm.

11.1.1.3 Verify public key size

Purpose	Verifies that the key size requirements are in accordance with Table 3-1 of SP80078.
Reference(s)	1. SP80078, Section 3.1 2. AS07.01.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>>

	3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate 6. Match the key size to Table 3-1 of SP80078
Expected Result(s)	The key sizes used are in accordance with Table 3-1 of SP80078.

11.1.2 Data Integrity Checks

11.1.2.1 Verify key usage extension

Purpose	Confirms that the PIV authentication certificate asserts the appropriate purpose of the key.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.01.05
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value of the keyUsage extension field from the certificate
Expected Result(s)	The digitalSignature bit has been set. No other bits have been set.

11.1.2.2 Verify id-fpki-common-authentication OID

Purpose	Confirms that the PIV Authentication certificate asserts the id-fpki-common-authentication OID.
Reference(s)	1. AS07.01.06
Precondition	1. A valid PIV card is inserted into the contact reader.

	2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract certificatePolicies->policyIdentifier extension field values from the certificate
Expected Result(s)	A policyIdentifier field in the certificatePolicies extension asserts id-fpki-common-authentication.

11.1.2.3 Verify authority information access extension

Purpose	Confirms the authority information access extension is populated with the location to the OCSP Server that provides status information for this certificate.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.01.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	An accessMethod containing id-ad-ocsp is present. The accessLocation for this AccessMethod is of type uniformResourceIdentifier and that the

	scheme is “http” (not “https”).

2171

2172 #### 11.1.2.4 Verify interim status extension

Purpose	Confirms that the piv-interim extension is present in the certificate.
Reference(s)	1. FIPS201, Appendix B 2. AS07.01.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.

Test Steps	• `Set cardHandle := <<valid card handle>>`
	• `Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>>`
	• `Call pivGetData w/`
	  • `(IN) cardHandle`
	  • `(IN) OID`
	  • `(OUT) data`
	• `Extract the piv-interim extension in the certificate`

Expected Result(s)	The piv-interim extension is present and contains the interim_indicator field which is of type BOOLEAN.

2173

2174 #### 11.1.2.5 Verify asymmetric key pair

Purpose	Confirms that the public key that exists in the certificate corresponds to the private key on the card.
Reference(s)	1. FIPS201, Section 5.2.1 2. AS07.01.13
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. `Take an arbitrary stream of data` 2. `Hash the data using a hash algorithm` 3. `Set cardHandle := <<valid card handle>>` 4. `Set authenticators := <<valid authenticator>>` 5. `Call pivLogIntoCardApplication` • `(IN) cardHandle`

	• (IN) authenticators 6. Set keyReference := <<key reference for PIV Authentication Key i.e. 9A>> 7. Set algorithmIdentifier := <<identifier of the algorithm to be used for the cryptographic operation>> 8. Set algorithmInput := <<hashed data from Step 2>> 9. Call pivCrypt with the following parameters • (IN) keyReference • (IN) algorithmIdentifier • (IN) algorithmInput • (OUT) algorithmOutput 10. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 11. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 12. Verify the signature in algorithmOutput (step 9) with subjectPublicKeyInfo->subjectPublicKey from the certificate
Expected Result(s)	The private key corresponds to the public key contained in the certificate as the signature verification succeeds.

11.1.2.6 Verify FASC-N and Card UUID

Purpose	Confirms that the FASC-N and Card UUID in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. This test shall also validate that no other name forms appear in the subjectAltName extension.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.01.08 3. AS07.01.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData with the following parameters

	• (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the GeneralNames field from the subjectAltName extension in the certificate 5. Parse the different GeneralName fields 6. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 7. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 8. Parse the CHUID and extract the FASC-N 9. Parse the CHUID and extract the GUID
Expected Result(s)	Step 8: A GeneralName field exists that contains an otherName with a type-id asserting the pivFASC-N OID. The value field of this otherName contains the FASC-N for the cardholder which matches the FASC-N obtained from parsing the CHUID. Step 9: A GeneralName field exists that contain a URI asserting a Card UUID as specified by [RFC4122, Section 3] that matches the GUID value in the CHUID. Note: No other name forms appear in the subjectAltName extension

11.1.2.7 Verify expiration dates consistency

Purpose	Confirms that the expiration date of the PIV authentication certificate is not past the expiration date of the PIV card.
Reference(s)	1. FIPS201, Section 4.2.2 2. AS07.01.15
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract validity->notAfter->utcTime field value from

	the certificate 5. Set OID := << CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the expiration date
Expected Result(s)	The expiration date of the PIV authentication certificate is not beyond the expiration date of the CHUID i.e. the PIV card.

11.1.2.8 Verify RSA exponent

Purpose	For RSA keys, confirms that the exponent of the RSA asymmetric key for PIV authentication is equal to 65,537.
Reference(s)	1. SP80078, Section 3.1 2. AS07.01.16
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate. 5. Parse the exponent from the extracted public key
Expected Result(s)	The exponent of the RSA asymmetric key for PIV authentication is equal to 65,537.

11.1.2.9 Verify HTTP URI in cRLDistributionPoints extension field

Purpose	Confirms that the URI in the cRLDistributionPoints extension field uses HTTP.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.01.10
Precondition	1. A valid PIV card is inserted into the contact reader.

	2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the cRLDistributionPoints extension fields from the certificate
Expected Result(s)	The URI is present with the "HTTP" scheme and points only to files with ".crl" extensions.

2183

2184 **11.1.2.10 Verify HTTP URI in authorityInfoAccess extension field**

Purpose	Confirms that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.01.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Authentication Certificate (2.16.840.1.101.3.7.2.1.1)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	The authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using

	HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

2185

2186

2187

11.2 Digital Signature Certificate

11.2.1 SP 800-78 Algorithm Conformance

11.2.1.1 Verify signature algorithm

Purpose	Confirms that the proper signature algorithm has been used to sign the certificate as specified in Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.02.01 3. AS07.02.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract signature->algorithm field value from the certificate
Expected Result(s)	From Step 4: The signatureAlgorithm value is in accordance with Table 3-3 of SP80078. If the algorithm value is id-RSASSA-PSS, verify that the signature->parameters field is populated with SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

11.2.1.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 3-4 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.02.03 3. AS07.02.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader.

	3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Match the algorithm value to the Table 3-4 of SP80078 6. If the algorithm is Elliptic Curve, ensure that one of the approved curves is used and the OID is populated in the subjectPublicKeyInfo->algorithm->parameters->namedCurve field from Table 3-5 of SP80078. The parameters field may contain NULL to indicate that parameters are inherited. Note: - If the RSA algorithm is used, the subjectPublicKeyInfo->algorithm->parameters field shall be NULL
Expected Result(s)	The digital signature key is generated using an allowed asymmetric key algorithm.

11.2.1.3 Verify public key size

Purpose	Verifies that the key size requirements are in accordance with Table 3-1 of SP80078.
Reference(s)	1. SP80078, Section 3.1 2. AS07.02.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate 6. Match the key size to Table 3-1 of SP80078
Expected Result(s)	The key sizes requirements are in accordance with Table 3-1 of SP80078.

2196

2197 ### 11.2.2 Data Integrity Checks

2198 #### 11.2.2.1 Verify key usage extension

Purpose	Confirms that the digital signature certificate asserts the appropriate purpose of the key contained in the certificate.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.02.05
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value of the keyUsage extension field from the certificate
Expected Result(s)	The digitalSignature and nonRepudiation bits have been set.

2199

2200 #### 11.2.2.2 Verify asymmetric key pair

Purpose	Confirms that the public key that exists in the certificate corresponds to the private key on the card.
Reference(s)	1. FIPS201, Section 5.2.1 2. AS07.02.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application

	which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Take an arbitrary stream of data 2. Hash the data using a hash algorithm 3. Set cardHandle := <<valid card handle>> 4. Set authenticators := <<valid authenticator>> 5. Call pivLogIntoCardApplication • (IN) cardHandle • (IN) authenticators 6. Set keyReference := <<key reference for Digital Signature Key i.e. 9C>> 7. Set algorithmIdentifier := <<identifier of the algorithm to be used for the cryptographic operation>> 8. Set algorithmInput := <<hashed data from Step 2>> 9. Call pivCrypt with the following parameters • (IN) keyReference • (IN) algorithmIdentifier • (IN) algorithmInput • (OUT) algorithmOutput 10. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 11. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 12. Verify the signature from step 9 algorithmOutput with subjectPublicKeyInfo->subjectPublicKey from the certificate
Expected Result(s)	The private key corresponds to the public key contained in the certificate as the signature verification succeeds.

11.2.2.3 Verify expiration dates consistency

Purpose	Confirms that the expiration date of the digital signature certificate is not past the expiration date of the PIV card.
Reference(s)	1. FIPS201, Section 4.2.2 2. AS07.02.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card. 5. A valid CHUID is present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<PIV Digital Signature certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract validity->notAfter->utcTime field value from the certificate 5. Set OID := << CHUID (2.16.840.1.103.3.7.2.48.0)>> 6. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 7. Parse the CHUID and extract the expiration date
Expected Result(s)	The expiration date of the digital signature certificate is not beyond the expiration date of the CHUID i.e. the PIV card.

11.2.2.4 Verify RSA exponent

Purpose	For RSA keys, confirm that the exponent of the RSA asymmetric key for digital signature is equal to 65,537.
Reference(s)	1. SP80078, Section 3.1 2. AS07.02.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate. 5. Parse the exponent from the extracted public key
Expected Result(s)	The exponent of the RSA asymmetric key for digital signature is equal to 65,537.

2206 **11.2.2.5 Verify the policyIdentifier field in the certificatePolicies**

Purpose	Confirms that the Digital Signature certificate asserts one of the following OIDs as part of the Digital Signature certificate profile: the id-fpki-common-hardware or id-fpki-common-High.
Reference(s)	1. AS07.02.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A Digital Signature key and corresponding certificate is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract certificatePolicies->policyIdentifier extension field values from the certificate
Expected Result(s)	A policyIdentifier field in the certificatePolicies extension asserts one of the following: id-fpki-common-hardware or id-fpki-common-High.

2207

2208 **11.2.2.6 Verify HTTP URI in cRLDistributionPoints extension field**

Purpose	Confirms that the URI in the cRLDistributionPoints extension field uses HTTP.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.02.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract the cRLDistributionPoints extension fields from the certificate
Expected Result(s)	The URI is present with the "HTTP" scheme and points only to files with ".crl" extensions.

2209

2210 **11.2.2.7 Verify HTTP URI in authorityInfoAccess extension field**

Purpose	Confirms that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.02.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Digital Signature Certificate (2.16.840.1.101.3.7.2.1.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	The authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

11.3 Key Management Certificate

11.3.1 SP 800-78 Algorithm Conformance

11.3.1.1 Verify signature algorithm

Purpose	Confirms that the proper signature algorithm has been used to sign the certificate as specified in Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.03.01 3. AS07.03.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data **4.** Extract signature->algorithm field value from the certificate
Expected Result(s)	1. From Step 4: The signatureAlgorithm value is in accordance with Table 3-3 of SP80078. 2. From Step 4: If the algorithm value is id-RSASSA-PSS, verify that the signature->parameters field is populated with SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

11.3.1.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 3-4 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.03.03 3. AS07.03.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and

	the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Match the algorithm value to the Table 3-4 of SP80078 6. If the algorithm is Elliptic Curve, ensure that one of the approved curves is used and the OID is populated in the subjectPublicKeyInfo->algorithm->parameters->namedCurve field from Table 3-5 of SP80078. The parameters field may contain NULL to indicate that parameters are inherited. Note: - If the RSA public key algorithm is used, the subjectPublicKeyInfo->algorithm->parameters field shall be NULL
Expected Result(s)	The key management key is generated using an allowed asymmetric key algorithm.

11.3.1.3 Verify public key size

Purpose	Verifies that the key size requirements are in accordance with Table 3-1 of SP80078.
Reference(s)	1. SP80078, Section 3.1 2. AS07.03.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate 6. Match the key size to Table 3-1 of SP80078 Note: - Since ECDH does not have any size restrictions based on dates, this test case does not apply to keys generated using these algorithms.
Expected Result(s)	The key sizes used are in accordance with Table 3-1 of SP80078.

2219

2220 ### 11.3.2 Data Integrity Checks

2221 #### 11.3.2.1 Verify key usage extension

Purpose	Confirms that the digital signature certificate asserts the appropriate purpose of the key contained in the certificate.
Reference(s)	1. AS07.03.05 2. AS07.03.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value of the keyUsage extension field from the certificate
Expected Result(s)	If the public key algorithm is RSA, then the keyUsage extension shall only assert the keyEncipherment bit. If the algorithm is Elliptic Curve key, then the keyUsage extension shall only assert the keyAgreement bit.

2222

2223 #### 11.3.2.2 Verify asymmetric key pair

Purpose	Confirms that the public key that exists in the certificate corresponds to

	the private key on the card.
Reference(s)	1. FIPS201, Section 5.2.1 2. AS07.03.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Take an arbitrary stream of data 2. Hash the data using a hash algorithm 3. Set cardHandle := <<valid card handle>> 4. Set authenticators := <<valid authenticator>> 5. Call pivLogIntoCardApplication • (IN) cardHandle • (IN) authenticators 6. Set keyReference := <<key reference for Key Management Key i.e. 9D>> 7. Set algorithmIdentifier := <<identifier of the algorithm to be used for the cryptographic operation>> 8. Set algorithmInput := <<hashed data from Step 2>> 9. Call pivCrypt with the following parameters • (IN) keyReference • (IN) algorithmIdentifier • (IN) algorithmInput • (OUT) algorithmOutput 10. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 11. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 12. Verify the signature from step 9 (algorithmOutput) with subjectPublicKeyInfo->subjectPublicKey from the certificate
Expected Result(s)	The private key corresponds to the public key contained in the certificate as the signature verification succeeds.

11.3.2.3 Verify RSA exponent

Purpose	For RSA keys, confirms that the exponent of the RSA asymmetric key for digital signature is equal to 65,537.
Reference(s)	1. SP80078, Section 3.1 2. AS07.03.12

Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A digital signature key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate. 5. Parse the exponent from the extracted public key
Expected Result(s)	The exponent of the RSA asymmetric key for key management is equal to 65,537.

11.3.2.4 Verify the policyIdentifier field in the certificatePolicies

Purpose	Confirms that the Key Management certificate asserts one of the following OIDs as part of the Key Management certificate profile: the id-fpki-common-policy, id-fpki-common-hardware, or id-fpki-common-High.
Reference(s)	1. AS07.03.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A Key Management key and corresponding certificate is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract certificatePolicies->policyIdentifier extension field values from the certificate
Expected Result(s)	A policyIdentifier field in the certificatePolicies extension asserts one of the following: id-fpki-common-policy, id-fpki-common-hardware, or id-

	fpki-common-High.

2228

2229 #### 11.3.2.5 Verify HTTP URI in cRLDistributionPoints extension field

Purpose	Confirms that the URI in the cRLDistributionPoints extension field uses HTTP.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.03.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the cRLDistributionPoints extension fields from the certificate
Expected Result(s)	The URI is present with the “HTTP” scheme and points only to files with “.crl” extensions.

2230

2231 #### 11.3.2.6 Verify HTTP URI in authorityInfoAccess extension field

Purpose	Confirms that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of “.p7c” containing a certs-only CMS message (see RFC 3851).
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.03.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A key management key and corresponding certificate are present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Key Management Certificate (2.16.840.1.101.3.7.2.1.2)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	The authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

11.4 Card Authentication Certificate

11.4.1 SP 800-78 Algorithm Conformance

11.4.1.1 Verify signature algorithm

Purpose	Confirms that the proper signature algorithm has been used to sign the certificate as specified in Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.04.01 3. AS07.04.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` **4.** `Extract signature->algorithm field value from the certificate`
Expected Result(s)	1. From Step 4: The signatureAlgorithm value is in accordance with Table 3-3 of SP80078 2. From Step 4: If the algorithm value is id-RSASSA-PSS, verify that the signature->parameters field is populated with SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

11.4.1.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 3-4 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.04.03 3. AS07.04.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle.

	4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Match the algorithm value to the Table 3-4 of SP80078 6. If the algorithm is Elliptic Curve, ensure that one of the approved curves is used and the OID is populated in the subjectPublicKeyInfo->algorithm->parameters->namedCurve field from Table 3-5 of SP80078. The parameters field may contain NULL to indicate that parameters are inherited. Note: - If the RSA algorithm is used, the subjectPublicKeyInfo->algorithm->parameters field shall be NULL
Expected Result(s)	The card authentication key is generated using the allowed asymmetric key algorithm.

11.4.1.3 Verify public key size

Purpose	Verifies that the key size requirements are in accordance with Table 3-1 of SP80078
Reference(s)	1. SP80078, Section 3.1 2. AS07.04.13
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract subjectPublicKeyInfo->algorithm->algorithm field value 5. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate 6. Match the key size to Table 3-1 of SP80078
Expected Result(s)	The key sizes used are in accordance with Table 3-1 of SP80078.

11.4.2 Data Integrity Checks

11.4.2.1 Verify key usage extension

Purpose	Confirms that the card authentication certificate asserts the appropriate purpose of the key.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.05
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the value of the keyUsage extension field from the certificate
Expected Result(s)	The digitalSignature bit has been set. No other bits have been set.

11.4.2.2 Verify id-fpki-common-cardAuth OID

Purpose	Confirms that the card authentication certificate asserts the id-fpki-

	common-cardAuth OID.
Reference(s)	1. AS07.04.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A PIV authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Extract certificatePolicies->policyIdentifier extension field values from the certificate`
Expected Result(s)	A policyIdentifier field in the certificatePolicies extension asserts id-fpki-common-cardAuth.

2244

2245 **11.4.2.3 Verify extended key usage extension**

Purpose	Confirms that the extKeyUsage in the card authentication certificate is present, is marked as critical, asserts the id-PIV-cardAuth OID, and does not assert any other OID.s
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Extract all KeyPurposeId fields from the extended key usage extension from the certificate`
Expected Result(s)	The extKeyUsage is present, is marked as critical, asserts the id-PIV-

	cardAuth OID, and does not assert any other OIDs.

2246

2247 #### 11.4.2.4 Verify authority information access extension

Purpose	Confirms the authority information access extension is populated with the location to the OCSP Server that provides status information for this certificate.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	An accessMethod containing id-ad-ocsp is present. The accessLocation for this AccessMethod is of type uniformResourceIdentifier and that the scheme is “http” (not “https”).

2248

2249 #### 11.4.2.5 Verify interim status extension

Purpose	Confirms that the piv-interim extension is present in the certificate.
Reference(s)	1. FIPS201, Appendix B 2. AS07.04.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.

Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the piv-interim extension in the certificate
Expected Result(s)	The piv-interim extension is present and contains the interim_indicator field which is of type BOOLEAN.

2250

2251 **11.4.2.6 Verify asymmetric key pair**

Purpose	Confirms that the public key that exists in the certificate corresponds to the private key on the card.
Reference(s)	1. FIPS201, Section 5.2.1 2. AS07.04.14
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Take an arbitrary stream of data 2. Hash the data using a hash algorithm 3. Set cardHandle := <<valid card handle>> 4. Set authenticators := <<valid authenticator>> 5. Call pivLogIntoCardApplication • (IN) cardHandle • (IN) authenticators 6. Set keyReference := <<key reference for card Authentication Key i.e. 9E>> 7. Set algorithmIdentifier := <<identifier of the algorithm to be used for the cryptographic operation>> 8. Set algorithmInput := <<hashed data from Step 2>> 9. Call pivCrypt with the following parameters • (IN) keyReference • (IN) algorithmIdentifier • (IN) algorithmInput • (OUT) algorithmOutput 10. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 11. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 12. Verify the signature from step 9 (algorithmOutput)with subjectPublicKeyInfo->subjectPublicKey from the certificate
Expected Result(s)	The private key corresponds to the public key contained in the

	certificate as the signature verification succeeds.

2252

2253 **11.4.2.7 Verify FASC-N and Card UUID**

Purpose	Confirms that the FASC-N and Card UUID in the subjectAltName field in the Card authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. This test shall also validate that no other name forms appear in the subjectAltName extension
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.09 3. AS07.04.15
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the GeneralNames field from the subjectAltName extension in the certificate 5. Parse the different GeneralName fields 6. Set OID := <<CHUID (2.16.840.1.103.3.7.2.48.0)>> 7. Call pivGetData with the following parameters • (IN) cardHandle • (IN) OID • (OUT) data 8. Parse the CHUID and extract the FASC-N 9. Parse the CHUID and extract the GUID
Expected Result(s)	Step 8: A GeneralName field exists that contains an otherName with a type-id asserting the pivFASC-N OID. The value field of this otherName contains the FASC-N for the cardholder which matches the FASC-N obtained from parsing the CHUID. Step 9: A GeneralName field exists that contain a URI asserting a UUID as specified by [RFC4122, Section 3] that matches the GUID value in the CHUID. Note: No other name forms appear in the subjectAltName extension.

2254

2255 #### 11.4.2.8 Verify RSA exponent

Purpose	For RSA keys, confirms that the exponent of the RSA asymmetric key for card authentication is equal to 65,537.
Reference(s)	1. SP80078, Section 3.1 2. AS07.04.16
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate. 5. Parse the exponent from the extracted public key.
Expected Result(s)	The exponent of the RSA asymmetric key for card authentication is equal to 65,537.

2256

2257 #### 11.4.2.9 Verify HTTP URI in cRLDistributionPoints extension field

Purpose	Confirms that the URI in the cRLDistributionPoints extension field uses HTTP.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID

	• (OUT) data 4. Extract the cRLDistributionPoints extension fields from the certificate
Expected Result(s)	The URI is present with the "HTTP" scheme and points only to files with ".crl" extensions.

2258

2259 #### 11.4.2.10 Verify HTTP URI in authorityInfoAccess extension field

Purpose	Confirms that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.04.12
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A card authentication key and corresponding certificate are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Card Authentication Certificate (2.16.840.1.101.3.7.2.5.0)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate
Expected Result(s)	The authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

2260

2261

2262 ## 11.5 Secure Messaging Card Verifiable Certificate

2263 ### 11.5.1 Secure Messaging CVC Profile Conformance

2264 #### 11.5.1.1 Validate General CVC Format

Purpose	Ensure expected behavior of initial secure messaging protocol establishment.
Reference(s)	1. SP80078 2. SP80073, part 2, Section 4

	3. AS07.05.06 4. AS07.05.10 5. AS07.05.11
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A secure messaging key and corresponding card verifiable certificate are present on the PIV card.
Test Steps	1. (Create Secure Messaging Channel) Send the GENERAL AUTHENTICATE command • CLA is set to: - '00' if command chaining is not needed or - '10' if command chaining is used. (The last chain of the command sets CLA to '00') • P1, algorithm reference, is set to '27', or '2E' • P2, key reference, is set to '03' indicating the PIV Secure Messaging Key 2. Verify that the control byte returned by the PIV card is 0x10 to ensure protocol was executed successfully according to input parameters 3. Perform public key validation of CVC CardHolderPublicKeyDataObject, where the domain parameters are those of Curve P-256 if P1 is '27' and those of Curve P-384 if P1 is '2E'. 4. Create CVC and ensure the Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 5. If the CVC is not signed with an Intermediate CVC skip this step and moved to Step 6. If the CVC is signed with an Intermediate CVC Verify the Signature on Intermediate CVC signer's signature in CVC. The Intermediate CVC is available within the Secure Messaging Certificate Signer data object using GETDATA <OID=2.16.840.1.101.3.7.2.16.23> 6. Verify Signature on content signer's signature in CVC. The X.509 Certificate for Content Signing shall also include an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing. The X.509 Certificate for Content Signing is available within the Secure Messaging Certificate Signer data object using GETDATA <OID=2.16.840.1.101.3.7.2.16.23>. 7. Parse the notBefore and notAfter values of the validity field and verify that the X.509 Certificate for Content Signing does not expire
Expected Result(s)	1. PIV card responds with control byte, nonce, authentication cryptogram, encrypted GUID, and confidential CVC 2. The control byte returned by the PIV card is 0x10

	3. Value matches expected result detailed in the "Value" column of table 13 in SP80073 Part 2 4. The Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 5. The Intermediate CVC signing certificate is available within the Secure Messaging Certificate Signer data object. 6. The X.509 Certificate for Content Signing includes an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing. 7. The expiration date for the X.509 Certificate for Content Signing is set to never expire

2265

2266 **11.5.1.2 Verify Secure Messaging CVC Profile**

Purpose	Ensure expected behavior of the defined elements within the secure messaging CVC certificate.
Reference(s)	1. SP80078 2. SP80073, part 2, Section 4 3. AS07.05.03 4. AS07.05.04 5. AS07.05.05
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A secure messaging key and corresponding card verifiable certificate are present on the PIV card.
Test Steps	1. (Create Secure Messaging Channel) Send the GENERAL AUTHENTICATE command • CLA is set to: - '00' if command chaining is not needed or - '10' if command chaining is used. (The last chain of the command sets CLA to '00') • P1, algorithm reference, is set to '27', or '2E' • P2, key reference, is set to '03' indicating the PIV Secure Messaging Key 2. Verify that the control byte returned by the PIV card is 0x10 to ensure protocol was executed successfully according to input parameters 3. Perform public key validation of CVC CardHolderPublicKeyDataObject, where the domain parameters are those of Curve P-256 if P1 is '27' and those of Curve P-384 if P1 is '2E'. 4. Create CVC and ensure the Subject Identifier is same 16-byte binary representation of the Card UUID value

	in the GUID field of the CHUID 5. Extract the tag value of the secure messaging CVC 6. Extract the Credential Profile Identifier 7. Parse the subjectKeyIdentifier and extract the Issuer Identification Number 8. Extract the Role Identifier
Expected Result(s)	1. PIV card responds with control byte, nonce, authentication cryptogram, encrypted GUID, and confidential CVC 2. The control byte returned by the PIV card is 0x10 3. Value matches expected result detailed in the “Value” column of table 13 in SP80073 Part 2 4. The Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 5. The Secure Messaging CVC will have a tag value of 0x7F21 when Subject Identifier contains a 16-byte GUID and shall be 0x7F22 when the length of Subject Identifier is 0 6. The Credential Profile Identifier is 0x80 7a. The Issuer Identification Number is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature 7b. If the public key needed to verify the signature on the secure messaging CVC appears in an Intermediate CVC then the Issuer Identification Number shall be the value of the Subject Identifier in the Intermediate CVC 8. The Role Identifier is 0x00 for card-application key CVC

11.5.2 Algorithm Conformance

11.5.2.1 Verify signature algorithm

Purpose	Confirms that the signature field in the certificate is in accordance with Table 15 of Part 2 in SP80073 and contains either an ECDSA signature using P-256 if the CardHolderPublicKey is P-256 or P-384 if the CardHolderPublicKey is P-384.
Reference(s)	1. SP80073, Part 2 Table 15 2. AS07.05.01 3. AS07.05.10
Precondition	1. A valid piv card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application

	which is accessible through card handle. 4. A secure messaging key and corresponding card verifiable certificate are present on the PIV card.
Test Steps	1. (Create Secure Messaging Channel) Send the GENERAL AUTHENTICATE command • CLA is set to: - '00' if command chaining is not needed or - '10' if command chaining is used. (The last chain of the command sets CLA to '00') • P1, algorithm reference, is set to '27', or '2E' • P2, key reference, is set to '03' indicating the PIV Authentication Key 2. Verify that the control byte returned by the PIV card is 0x10 to ensure protocol was executed successfully according to input parameters 3. Create CVC and ensure the Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 4. Extract Digital Signature->algorithm field value from the CVC
Expected Result(s)	1. PIV card responds with control byte, nonce, authentication cryptogram, encrypted GUID, and confidential CVC 2. The control byte returned by the PIV card is 0x10 3. The Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 4. The signatureAlgorithm value is in accordance with Table 15 of Part 2 in SP80073. The AlgorithmIdentifer OID is 1.2.840.10045.4.3.2 for ECDSA with SHA-256 (Cipher Suite 2) or 1.2.840.10045.4.3.3 for ECDSA with SHA-384 (Cipher Suite 7). For both algorithms, the parameters field is absent.

11.5.2.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 15 of SP80073 Part 2.
Reference(s)	1. SP80073 Part 2 Section 4.1.5 2. AS07.05.02 3. AS07.05.12
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A secure messaging key and corresponding card verifiable

	certificate are present on the PIV card.
Test Steps	1. (Create Secure Messaging Channel) Send the GENERAL AUTHENTICATE command • CLA is set to: - '00' if command chaining is not needed or - '10' if command chaining is used. (The last chain of the command sets CLA to '00') • P1, algorithm reference, is set to '27', or '2E' • P2, key reference, is set to '03' indicating the Secure Messaging Key 2. Verify that the control byte returned by the PIV card is 0x10 to ensure protocol was executed successfully according to input parameters 3. Create CVC and ensure the Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 4. Extract CVC CardHolderPublicKeyDataObject->algorithm OID->algorithm OID value **5.** Extract the Public Key object
Expected Result(s)	1. PIV card responds with control byte, nonce, authentication cryptogram, encrypted GUID, and confidential CVC 2. The control byte returned by the PIV card is 0x10 3. The Subject Identifier is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID 4. Verify appropriate value for the implementation: • 0x2A8648CE3D030107 for ECDH (Curve P-256) for CS2 • 0x2B81040022 for ECDH (Curve P-384) for CS7 5. The Public Key object is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve.

11.5.2.3 Secure Messaging Cipher Suite Implementation

Purpose	Ensure that secure messaging algorithms are supported and that the proper cipher suite is in use according to key strength.
Reference(s)	1. SP80078 Part 1, Section C.3 2. AS07.05.07 3. AS07.05.10
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through 4. card handle.

	5. A secure messaging key and corresponding card verifiable certificate are present on the PIV card.
Test Steps	1. Send SELECT card command with, 1. AID == A0 00 00 03 08 00 00 10 00 01 00 2. Parse the returned APT 3. Identify the cryptographic algorithms supported within the AC tag. If '27' is present proceed to step 4. If '27' is not present in the AC tag, the test is complemented 4. Set OID := <<valid OID>> (Repeat this for all implemented objects in the following set - X.509 Certificate for Digital Signature, X.509 Certificate for Key Management, 20 retired X.509 Certificates for Key Management) 5. Create data reference 6. Call pivGetData w/ (each data object identified in step 4) (IN) cardHandle (IN) OID (IN) oidLength (OUT) data (INOUT) DataLength 7. Examine SubjectPublicKeyInfo of each returned certificate. Ensure that the AlgorithmIdentifier is not an elliptic Curve P-384 algorithm identifier. For the digital signature certificate, the key management certificate and retired key management certificate, the OID shall not be 1.2.840.10045.4.2.1.
Expected Result(s)	1. Returns with status_word of PIV_OK and initialized applicationProperties reference 2. If the PIV card supports secure messaging the 27 and/or 2E algorithms are returned within the AC tag of the APT 3. None of the certificates shall contain ECC keys with a P-384 curve.

11.5.3 Data Integrity Checks

11.5.3.1 Verify asymmetric key pair

Purpose	Confirms that the public key that exists in the certificate corresponds to the private key on the card.
Reference(s)	1. SP80073 Part 2 Section 4.1.4 2. AS07.05.09
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader.

	2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. A secure messaging key and corresponding card verifiable certificate are present on the PIV card.
Test Steps	1. (Create Secure Messaging Channel) Send the GENERAL AUTHENTICATE command • CLA is set to: • '00' if command chaining is not needed or • '10' if command chaining is used. (The last chain of the command sets CLA to '00') • P1, algorithm reference, is set to '27', or '2E' • P2, key reference, is set to '03' indicating the PIV Authentication Key 2. Verify that the control byte returned by the PIV card is 0x10 to ensure protocol was executed successfully according to input parameters 3. Perform public key validation of CVC CardHolderPublicKeyDataObject, where the domain parameters are those of Curve P-256 if P1 is '27' and those of Curve P-384 if P1 is '2E'. Compute key confirmation key per SP80073 part 2, section 4.1.6.
Expected Result(s)	1. PIV card responds with control byte, nonce, authentication cryptogram, encrypted GUID, and confidential CVC 2. The control byte returned by the PIV card is 0x10 3. Value matches expected result detailed in the "Value" column of table 13 in SP80073 Part 2. Executed successful key confirmation using key confirmation key and authentication cryptogram returned in step 1.

11.6 Intermediate Card Verifiable Certificate

11.6.1 Intermediate CVC Profile Conformance

11.6.1.1 Validate General CVC Format

Purpose	Ensure that the general Intermediate CVC format is in accordance with SP80073 Part 2, Section 4
Reference(s)	1. SP80073, part 2, Section 4 2. AS07.06.09
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader.

	3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding Intermediate CVC are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the Intermediate CVC 5. Extract signature->algorithm field value from the certificate 6. Perform public key validation of the Intermediate CVC CardHolderPublicKeyDataObject, where the domain parameters are those of Curve P-256 if P1 is '27' and those of Curve P-384 if P1 is '2E. 7. Ensure the Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object 8. Verify signature on content signer's signature in the Intermediate CVC. The X.509 Certificate for Content Signing shall also include an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing. The X.509 Certificate for Content Signing is available within the Secure Messaging Certificate Signer data object using GETDATA <OID=2.16.840.1.101.3.7.2.16.23>. 9. Parse the notBefore and notAfter values of the validity field and verify that the X.509 Certificate for Content Signing does not expire
Expected Result(s)	1. From Step 6: Value matches expected result detailed in the "Value" column of table 13 in SP80073 Part 2 2. From Step 7: The Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object 3. From Step 8: The X.509 Certificate for Content Signing includes an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing. 4. From Step 9: The expiration date for the X.509 Certificate for Content Signing is set to never expire

2283

2284 **11.6.1.2 Verify Intermediate CVC Profile**

Purpose	Ensure expected behavior of the defined elements within the Intermediate CVC profile.

Reference(s)	1. SP80073, part 2, Section 4 2. AS07.06.03 3. AS07.06.04 4. AS07.06.05 5. AS07.06.06 6. AS07.06.07 7. AS07.06.08
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding Intermediate CVC are present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the Intermediate CVC` 5. `Perform public key validation of CVC CardHolderPublicKeyDataObject, where the domain parameters are those of Curve P-256 if P1 is '27' and those of Curve P-384 if P1 is '2E'.` 6. `Ensure the Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object` 7. `Extract the tag value of the Intermediate CVC` 8. `Extract the Credential Profile Identifier` 9. `Parse the subjectKeyIdentifier and extract the Issuer Identification Number` 10. `Extract the Algorithm OID of the Intermediate CVC` 11. `Extract the Role Identifier`
Expected Result(s)	1. `From Step 5: Value matches expected result detailed in the “Value” column of table 13 in SP80073 Part 2` 2. `From Step 6: The Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object` 3. `From Step 7: The Intermediate CVC will have a tag value of 0x7F21` 4. `From Step 8: The Credential Profile Identifier is 0x80` 5. `From Step 9: The Issuer Identification Number is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the`

	signature 6. From Step 10: The Algorithm OID of the Intermediate CVC is 0x2A8648CE3D030107 for ECDH (Curve P-256) or 0x2B81040022 for ECDH (Curve P-384) 7. From Step 11: The Role Identifier is 0x12 for card-application root CVC.

2285

2286 ### 11.6.2 Algorithm Conformance

2287 #### 11.6.2.1 Verify signature algorithm

Purpose	Confirms that the signature field in the certificate is in accordance with Table 16 of Part 2 in SP80073 and contains an algorithm for RSA with SHA-256 and PKCS #1 v1.5 padding.
Reference(s)	1. SP80073, Part 2 Table 16 2. AS07.06.01 3. AS07.06.11
Precondition	1. A valid piv card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding Intermediate CVC are present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the Intermediate CVC 5. Ensure the Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object 6. Extract Digital Signature->algorithm field value from the Intermediate CVC
Expected Result(s)	1. From Step 5: The Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object 2. From Step 6: The signatureAlgorithm value is in accordance with Table 16 of Part 2 in SP80073. The AlgorithmIdentifer OID is 1.2.840.113549.1.1.11 for RSA with SHA-256 and PKCS #1 V1.5 padding. The parameters field shall be NULL.

2288

2289 **11.6.2.2 Verify subject public key algorithm**

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 16 of SP80073 Part 2.
Reference(s)	1. SP80073, Part 2 Table 16 2. AS07.06.02 3. AS07.06.10
Precondition	1. A valid PIV card that supports secure messaging is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding Intermediate CVC are present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the Intermediate CVC` 5. `Ensure the Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object` 6. `Extract CVC CardHolderPublicKeyDataObject->algorithm OID->algorithm OID value` 7. `Extract the Public Key object`
Expected Result(s)	1. `From Step 5: The Subject Identifier is the leftmost 8 bytes of the SHA-1 hash of the Public Key object` 2. `From Step 6: Verify appropriate value for the implementation:` • `0x2A8648CE3D030107 for ECDH (Curve P-256) for CS2` • `0x2B81040022 for ECDH (Curve P-384) for CS7` 3. `From Step 7: The Public Key object is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve`

2290

2291

11.7 X.509 Certificate for Content Signing

11.7.1 Algorithm Conformance

11.7.1.1 Verify signature algorithm

Purpose	Confirms that the proper signature algorithm has been used to sign the certificate as specified in Table 3-3 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.07.01 3. AS07.07.02
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the X.509 Certificate for Content Signing` 5. `Extract signature->algorithm field value from the certificate`
Expected Result(s)	The signatureAlgorithm value is in accordance with Table 3-3 of SP80078. If the algorithm value is id-RSASSA-PSS, verify that the signature->parameters field is populated with SHA-256 (OID = 2.16.840.1.101.3.4.2.1). For the other RSA algorithms, the parameters field is populated with NULL. For ECDSA, the parameters field is absent.

11.7.1.2 Verify subject public key algorithm

Purpose	Confirms that the public key algorithm used for generating the keys is as specified in Table 3-4 of SP80078.
Reference(s)	1. SP80078, Section 3.2.1 2. AS07.07.03 3. AS07.07.04
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and

	the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the X.509 Certificate for Content Signing` 5. `Extract subjectPublicKeyInfo->algorithm->algorithm field value` 6. `Match the algorithm value to the Table 3-4 of SP80078` 7. `If the algorithm is Elliptic Curve, ensure that one of the approved curves is used and the OID is populated in the subjectPublicKeyInfo->algorithm->parameters->namedCurve field from Table 3-5 of SP80078. The parameters field may contain NULL to indicate that parameters are inherited.` `Note: - If the RSA algorithm is used, the subjectPublicKeyInfo->algorithm->parameters field shall be NULL`
Expected Result(s)	The X.509 Certificate for Content Signing is generated using an allowed asymmetric key algorithm.

11.7.1.3 Verify public key size

Purpose	Verifies that the key size requirements are in accordance with Table 3-2 of SP80078.
Reference(s)	1. SP80078, Table 3-2 2. AS07.07.09
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer`

	(2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the X.509 Certificate for Content Signing 5. Extract subjectPublicKeyInfo->algorithm->algorithm field value 6. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate 7. Match the key size to Table 3-1 of SP80078
Expected Result(s)	The key sizes requirements are in accordance with Table 3-2 of SP80078.

2299

2300 ### 11.7.2 Data Integrity Checks

2301 #### 11.7.2.1 Verify key usage extension

Purpose	Confirms that the X.509 Certificate for Content Signing asserts the appropriate purpose of the key contained in the certificate.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.07.05
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the X.509 Certificate for Content Signing 5. Extract the value of the keyUsage extension field from the certificate
Expected Result(s)	The digitalSignature and nonRepudiation bits have been set and no other keyUsage bits are set.

2302

2303 #### 11.7.2.2 Verify expiration dates consistency

Purpose	Confirms that the expiration date of the X.509 Certificate for Content Signing is not expired.
Reference(s)	1. FIPS201, Section 4.2.2 2. AS07.07.11
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card. 5. A valid CHUID is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the X.509 Certificate for Content Signing` 5. `Extract validity->notAfter->utcTime field value from the certificate`
Expected Result(s)	The X.509 Certificate for Content Signing is not expired.

2304

2305 #### 11.7.2.3 Verify RSA exponent

Purpose	For RSA keys, confirm that the exponent of the RSA asymmetric key for the X.509 Certificate for Content Signing is equal to 65,537.
Reference(s)	1. SP80078, Section 3.1 2. AS07.07.10
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer`

	(2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the X.509 Certificate for Content Signing 5. Extract the subjectPublicKeyInfo->subjectPublicKey from the certificate. 6. Parse the exponent from the extracted public key
Expected Result(s)	The exponent of the RSA asymmetric key for the X.509 Certificate for Content Signing is equal to 65,537.

11.7.2.4 Verify the policyIdentifier field in the certificatePolicies

Purpose	Confirms that the policyIdentifier field in the certificatePolicies extension of the X.509 Certificate for Content Signing asserts the id-fpki-common-piv-contentSigning policy of [COMMON] (OID = 2.16.840.1.101.3.2.1.3.39) and it include an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing.
Reference(s)	1. FIPS201, Appendix B 2. AS07.07.06
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. Set cardHandle := <<valid card handle>> 2. Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>> 3. Call pivGetData w/ • (IN) cardHandle • (IN) OID • (OUT) data 4. Parse the X.509 Certificate for Content Signing 5. Extract certificatePolicies->policyIdentifier extension field values from the certificate 6. Extract all KeyPurposeId fields from the extended keyUsage extension
Expected Result(s)	A policyIdentifier field in the certificatePolicies extension asserts the id-fpki-common-piv-contentSigning policy of [COMMON] (OID = 2.16.840.1.101.3.2.1.3.39) and the extended key usage (extKeyUsage)

	extension will assert the id-PIV-content-signing.

2308

2309 **11.7.2.5 Verify HTTP URI in cRLDistributionPoints extension field**

Purpose	Confirms that the URI in the cRLDistributionPoints extension field uses HTTP.
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.07.07
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle. 4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the X.509 Certificate for Content Signing` 5. `Extract the cRLDistributionPoints extension fields from the certificate`
Expected Result(s)	The URI is present with the “HTTP” scheme and points only to files with “.crl” extensions.

2310

2311 **11.7.2.6 Verify HTTP URI in authorityInfoAccess extension field**

Purpose	Confirms that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of “.p7c” containing a certs-only CMS message (see RFC 3851).
Reference(s)	1. X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework [COMMON] 2. AS07.07.08
Precondition	1. A valid PIV card is inserted into the contact reader. 2. A valid PC/SC connection exists between the test application and the contact reader. 3. The test application is currently connected to the card application which is accessible through card handle.

	4. The Secure Messaging Certificate Signer data object and corresponding X.509 Certificate for Content Signing is present on the PIV card.
Test Steps	1. `Set cardHandle := <<valid card handle>>` 2. `Set OID := <<Secure Messaging Certificate Signer (2.16.840.1.101.3.7.2.16.23)>>` 3. `Call pivGetData w/` • `(IN) cardHandle` • `(IN) OID` • `(OUT) data` 4. `Parse the X.509 Certificate for Content Signing` 5. `Extract the AuthorityInfoAccess->accessMethod and AuthorityInfoAccess->accessLocation extension fields from the certificate`
Expected Result(s)	The authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod and the access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).

Appendix A—DTRs to Test Assertion Mapping

The following table provides an association between the Required Test Procedures in DTRs in Sections 4, 5, 6, and 7 (those that can be electronically tested) and the test assertions in Sections 8, 9, 10, and 11.

A.1 BER-TLV Mapping

DTR from Section 4	Test Assertion from Section 8	DTR Description
TE04.01.01.01	8.1 “Card Capabilities Container “Data Object 8.2 “Card Holder Unique Identifier” Data Object 8.3 “X.509 Certificate for PIV Authentication” Data Object 8.4 “Off-Card Comparison Card Holder Fingerprints” Data Object 8.5 “Printed Information” Data Object 8.6 “Card Holder Facial Image” Data Object 8.7 “X.509 Certificate for Digital Signature” Data Object 8.8 “X.509 Certificate for Key Management” Data Object 8.9 “X.509 Certificate for Card Authentication” Data Object 8.10 “Security Object Data Object 8.11 “Discovery” Data Object 8.12 “Card Holder Iris Images” Data Object 8.13 Retired X.509 Certificate for Key Management” Data Object 8.15 “Biotmetric Information Templates Group Template” Data Object 8.16 “Secure Messaging Certificate Signer” Data Object 8.17 “Pairing Code Reference Data Container” Data Object 8.18 Unused Data Objects on the PIV Card	The tester shall validate that the formatting, encoding and the content of all the elements in each data container conforms to SP80073 Part 1.

DTR from Section 4	Test Assertion from Section 8	DTR Description
TE04.02.01.01	8.1 “Card Capabilities Container” Data Object	The tester shall validate the format and the content of all the elements in CCC data container on the card.
TE04.02.01.02	8.1 “Card Capabilities Container” Data Object	The tester shall validate that the registered data model value is 0x10.
TE04.03.01.01	8.2 “Cardholder Unique Identifier” Data Object	The tester shall validate the format and the content of all the elements in CHUID data container on the card.
TE04.04.01.01	8.4 “Cardholder Fingerprint” Data Object	The tester shall validate that the fingerprint data follows the tag value 0xBC within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.
TE04.05.01.01	8.6 “Cardholder Facial Image” Data Object	The tester shall validate that the facial image follows the tag value 0xBC within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.
TE04.06.01.01	8.10 “Security Object” Data Object	The tester shall validate that all unsigned data objects, such as the Printed Information data object, are included in the Security Object if present and that the message digests for the various data objects present in the security object are identical to the message digest of the data object itself.
TE04.07.01.01	8.12 “Card Holder Iris Images” Data Object	The tester shall validate that the iris image follows the tag value 0xBC within the container and the FASC-N is present in the CBEFF header as well as in the CBEFF signature block. The Card UUID is present in the CBEFF signature block.
TE04.08.01.01	8.14 “Key History” Data Object	The tester shall validate the format and the contents of all the elements in Key History data container on the card.
TE04.09.01.01	8.11 “Discovery” Data Object	The tester shall validate that both tag 0x4F (PIV Card Application AID) and tag 0x5F2F (PIN Usage Policy) data elements are present in the Discovery Object. The tester shall validate the format and the contents of these data elements in the Discovery object data container on the card.
TE04.10.01.01	8.15 “Biometric Information Templates Group Template” Data Object 9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall ensure that encoding of the BIT group template is in accordance with Table 7 of SP80076.
TE04.10.01.02	8.11 “Discovery Object” Data Object	When the BIT Group Template is present, the tester shall ensure that bit 4 of the first byte of the PIN Usage Policy is set.

DTR from Section 4	Test Assertion from Section 8	DTR Description
TE04.11.01.01	8.16 “Secure Messaging Certificate Signer” Data Object	If PIV card secure messaging for non-card-management operations is enabled, the tester shall ensure that the Secure Messaging Certificate Signer data object is present and contains the certificate (and Intermediate CVC, if applicable) needed to verify the signature on secure messaging CVC.
TE04.12.01.01	8.17 “Pairing Code Reference Data Container” Data Object	If the PIV card supports VCI with pairing, then the tester shall ensure that the Pairing Code Reference Data Container is present and includes a copy of the PIV card’s pairing code.
TE04.13.01.01	8.18 Unused Data Objects on the PIV Card	The tester shall confirm that all unused data containers on the card are set to a zero length by performing GET DATA on the unused data objects and ensure that the length of the returned objects is equal to zero (i.e., the value field is absent).

A.2 Biometric Data Mapping

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.01.01.01	9.1.1 CBEFF Structure for Fingerprint Template 9.2.1 CBEFF Structure for Facial Image 9.3.1 CBEFF Structure for Iris Image	The tester shall verify that the CBEFF structure is implemented in accordance with Table 13 of SP80076.
TE05.01.02.01	9.1.2 CBEFF Header for Fingerprint Template 9.2.2 CBEFF Header for Facial Image 9.3.2 CBEFF Header for Iris Image	The tester shall verify the length of the Patron Format header.
TE05.01.02.02	9.1.2 CBEFF Header for Fingerprint Template 9.2.2 CBEFF Header for Facial Image 9.3.2 CBEFF Header for Iris Image	The tester shall verify the values are consistent with Table 14 “Patron format PIV specification” requirements of SP80076.

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.01.03.01	9.1.1 CBEFF Structure for Off-Card Comparison Fingerprint Template 9.2.1 CBEFF Structure for Facial Image 9.3.1 CBEFF Structure for Iris Image 9.4.1 General Record Header Conformance 9.4.2 View Header Conformance	The tester shall compare value provided against the stored data.
TE05.01.04.01	9.1.2.1 Patron Header Version 9.2.2.1 Patron Header Version 9.3.2.1 Patron Header Version	The tester shall verify that the Patron Header Version value is 0x03.
TE05.01.05.01	9.1.2.2 SBH Security Option 9.2.2.2 SBH Security Option 9.3.2.2 SBH Security Option	The tester shall verify that the SBH security option value is b00001101.
TE05.01.06.01	9.1.2.3 BDB Format Owner Values 9.2.2.3 BDB Format Owner Values 9.3.2.3 BDB Format Owner	The tester shall verify that the BDB Format Owner field contains 0x001B for fingerprint and facials records and contains 0x0101 for iris images.
TE05.01.07.01	9.1.2.4 BDB Format Type 9.2.2.4 BDB Format Type 9.3.2.4 BDB Format Type	The tester shall verify that the BDB Format Type field is 0x0201 for fingerprint template, 0x0501 for facial image, and 0x0009 for the optional iris image if present.
TE05.01.08.01	9.1.2.5 Biometric Creation Date 9.2.2.5 Biometric Creation Date 9.3.2.5 Biometric Creation Date	The tester shall verify the date field is in compliance with the assertion.
TE05.01.09.01	9.1.2.6 Validity Period Dates 9.2.2.6 Validity Period Dates 9.3.2.6 Validity Period Dates	The tester shall verify that the headers contain two dates in compliance with the assertion.

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.01.10.01	9.1.2.7 Biometric Type Values 9.2.2.7 Biometric Type Values 9.3.2.7 Biometric Type Values	The tester shall verify that the Biometric Type field contains 0x000008 for fingerprint templates, 0x000002 for facial images, and 0x000010 for the iris images.
TE05.01.11.01	9.1.2.8 Biometric Data Type 9.2.2.8 Biometric Data Type 9.3.2.8 Biometric Data Type	The tester shall verify that the Biometric Data Type field within the PIV Patron Format is b100xxxxx for the fingerprint templates, b001xxxxx for the facial images, and b010xxxxx for the iris images.
TE05.01.12.01	9.1.2.9 Biometric Data Quality 9.2.2.9 Biometric Data Quality 9.3.2.9 Biometric Data Quality	The tester shall verify that the value of Biometric Data Quality is between -2 and 100 for the fingerprint templates, facial images, and iris images.
TE05.01.13.01	9.1.2.10 Creator Field Value 9.2.2.10 Creator Field Value 9.3.2.10 Creator Field Value	The tester shall verify that the Creator field in the PIV Patron Format contains 18 bytes of which the first $K \leq 17$ bytes is printable ASCII characters, and the first of the remaining 18-K is a null terminator (zero).
TE05.01.14.01	9.1.2.11 FASC-N Value 9.2.2.11 FASC-N Value 9.3.2.11 FASC-N Value	The tester shall verify that the FASC-N field in the PIV Patron Format shall contain the 25 bytes of the FASC-N component of the CHUID identifier. Note: This field may be filled with zeroes in the one exceptional case where PIV registration images are being stored before a FASC-N has been assigned. In such instances, the digital signature shall be regenerated once the FASC-N is known.
TE05.01.15.01	9.1.2.12 Reserved Field Value 9.2.2.12 Reserved Field Value 9.3.2.12 Reserved Field Value	The tester shall verify the “Reserved for future use” field is 0x00000000.
TE05.02.01.01	9.4.1 General Record Header Conformance	The tester shall parse the biometric data container to verify this assertion. Note: The CBEFF structure itself is tested in later assertions.
TE05.02.02.01	9.4.1 General Record Header Conformance	The tester shall verify that the resultant template is in compliance with the assertion.
TE05.02.03.01	9.4.1 General Record Header Conformance	The tester shall verify that the Format Identifier value is 0x464D5200.
TE05.02.04.01	9.4.1 General Record Header Conformance	The tester shall verify that the Version Number is 0x20323000.
TE05.02.05.01	9.4.1 General Record Header Conformance	The tester shall verify that the Record Length of the General Record Header is $26 \leq L \leq 1574$.

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.02.06.01	9.4.1 General Record Header Conformance	The tester shall verify that the both of the two fields ("Owner" and "Type") of the CBEFF Product Identifier are > 0 (non-zero).
TE05.02.07.01	9.4.1 General Record Header Conformance	The tester shall verify the Capture Equipment Compliance value is 1000b.
TE05.02.08.01	9.4.1 General Record Header Conformance	The tester shall verify the Capture Equipment ID of the General Record Header is > 0.
TE05.02.09.01	9.4.1 General Record Header Conformance	The tester shall verify that the width on Size of Scanned Image in X Direction is the larger of the widths of the two input images and the height on Size of Scanned Image in Y Direction is the larger of the heights of the two input images.
TE05.02.10.01	9.4.1 General Record Header Conformance	The tester shall verify that the X (horizontal) and Y (vertical) resolutions are 197.
TE05.02.11.01	9.4.1 General Record Header Conformance	The tester shall verify the Number of Views value is 2.
TE05.02.12.01	9.4.1 General Record Header Conformance	The tester shall verify the Reserved Byte value is 0.
TE05.02.13.01	9.4.2 View Header Conformance	The tester shall verify the View Number value of the Single Finger View Record is 0.
TE05.02.14.01	9.4.2 View Header Conformance	The tester shall verify the value is either 0 or 2 and is consistent with vendor reporting.
TE05.02.15.01	9.4.2 View Header Conformance	The tester shall verify that the quality value of captured fingerprint images shall be 20, 40, 60, 80, 100, 254, or 255. Note: A value of "255" shall be assigned when fingerprints are temporarily unusable for matching. A value of "254" shall be assigned when the fingerprints are permanently unusable.
TE05.02.16.01	9.4.2 View Header Conformance	The tester shall verify the Number of Minutiae is between 0 and 128.
TE05.02.17.01	9.4.3 Fingerprint Minutiae Data Records	The tester shall verify that the Minutiae Type is 00b, 01b, or 10b.
TE05.02.19.01	9.4.3 Fingerprint Minutiae Data Records	The tester shall verify that the value of Extended Data Block Length is zero.
TE05.03.01.01	9.6.1 Facial Image Data Conformance 9.6.2 Facial Image Header Conformance	The tester shall review the documentation to verify compliance with the assertion.
TE05.03.02.01	9.6.1 Facial Image Data Conformance	The tester shall verify that the Format Identifier value is 0x46414300.
TE05.03.03.01	9.6.1 Facial Image Data Conformance	The tester shall verify that the Version Number is 0x30313000.
TE05.03.04.01	9.6.1 Facial Image Data Conformance	The tester shall verify that the Record Length of the Facial fits within the container size limits specified in [800-73].

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.03.05.01	9.6.1 Facial Image Data Conformance	The tester shall verify that the Number of Facial Images of the Facial Header is ≥1 and that the most recent image appears first and serve as the default provided to the application.
TE05.03.06.01	9.6.1 Facial Image Data Conformance	The tester shall verify that the Number of Feature Points for the facial image is ≥0.
TE05.03.07.01	9.6.2 Facial Image Header Conformance	The tester shall verify that the Facial Image Type is 1.
TE05.03.08.01	9.6.2 Facial Image Header Conformance	The tester shall verify that the Image Data Type is 0 or 1. Note: Both whole-image and single-region-of-interest (ROI) compression are permitted. 800-76 recommends that newly collected facial image should be compressed using ISO/IEC 15444 (i.e., JPEG 2000).
TE05.03.09.01	9.6.2 Facial Image Header Conformance	The tester shall verify that the Image Color Space of the facial image is 1 and is converted to the sRGB color space.
TE05.03.10.01	9.6.2 Facial Image Header Conformance	The tester shall verify that the Source Type of the facial image is 2 or 6.
TE05.04.01.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the resultant BITs is in compliance with the assertion.
TE05.04.02.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the value for Number of Fingers is 2 in tag 0x02.
TE05.04.03.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the Reference data qualifier used by VERIFY (tag 0x83) for the first finger is ‘96’.
TE05.04.04.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the Reference data qualifier used by VERIFY (tag 0x83) for the second finger is ‘97’.
TE05.04.05.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the Biometric type (tag 0x81) for the first and second finger is 08.
TE05.04.06.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the subtype values for the first and second finger match the values identified in SP80076 Table 8.
TE05.04.07.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the CBEFF BDB format owner (tag 0x87) for the first and second finger is set to 0101.
TE05.04.08.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that the CBEFF BDB format type (tag 0x88) for the first and second finger is set to 0005.
TE05.04.09.01	9.5.1 BIT Group Template data conformance for on-card comparison	The tester shall verify that TAG 83 is not present in the Biometric Matching parameters for the first and second finger.
TE05.05.01.01	9.7.1 Iris Image Profile	The tester shall review the documentation to verify compliance with the assertion.

DTR from Section 5	Test Assertion from Section 9	DTR Description
TE05.05.02.01	9.7.1 Iris Image Profile	The tester shall verify that the Format identifier of the Iris General Header is 0x49495200.
TE05.05.03.01	9.7.1 Iris Image Profile	The tester shall verify that the Version number of the Iris General Header is 0x30323000.
TE05.05.04.01	9.7.1 Iris Image Profile	The tester shall verify that the Length of record of the Iris General Header is less than or equal to size specified in 800-73-4 and JPEG 2000 compressed iris image implementations are executed with a bit rate input value that corresponds to the 3Kilobyte target result.
TE05.05.05.01	9.7.1 Iris Image Profile	The tester shall verify that the Number of iris representations of the Iris General Header is 1 or 2.
TE05.05.06.01	9.7.1 Iris Image Profile	The tester shall verify that the Certification flag of the Iris General Header is 0x00.
TE05.05.07.01	9.7.1 Iris Image Profile	The tester shall verify that the Number of eyes represented is 1 or 2.
TE05.05.08.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Capture date and time is 2011 onwards.
TE05.05.09.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Representation number is 1 and then, optionally 2.
TE05.05.10.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Eye label is 1 for left eye and 2 for right eye. If camera does not estimate automatically, then these are manually assigned.
TE05.05.11.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Image type is type 7 with (0,6R 0,2R) margins.
TE05.05.12.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Image format is 10 = 0x0A and the compression algorithm and encoding is mono JPEG 2000.
TE05.05.13.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Iris image properties bit field is (Bits 1-2: 01 or 10), (Bits 3-4: 01 or 10), (Bits 5-6: 01 and scan type shall be progressive), and (Bits 7-8: 01 and the compression history shall be “none”). Note: Bit 1 is the least significant bit and Bit 8 is the most significant.
TE05.05.14.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Image width (image width, W) is $288 \leq W \leq 448$.
TE05.05.15.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Image height (image height, W) is $216 \leq H \leq 336$.
TE05.05.16.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Bit depth is 8. Note: Bit depth is in bits per pixel and shall not be used to indicate compression level.
TE05.05.17.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Iris centre, lowest X is (W/2 for W odd, else), highest X is (W/2+1 for W even), lowest Y is (H/2 for H odd, else), and highest Y is (W/2+1 for H even).
TE05.05.18.01	9.7.2 Iris Image Data Conformance	The tester shall verify that the Iris diameter, lowest is $D \geq 160$ and the highest is $D \leq 280$.

A.3 CHUID Mapping

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.01.01.01	10.1.1.1 Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the CHUID data object contains a digital signature and has been formatted correctly as a CMS external signature as defined in RFC 5652.
TE06.01.02.01	10.1.1.1 Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.
TE06.01.03.01	10.1.1.2 Verify version in SignedData	The tester shall validate the version of the SignedData type is version 3.
TE06.01.04.01	10.1.1.3 Verify digest Algorithm in SignedData	The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.
TE06.01.05.01	10.1.1.4 Verify contents of encapContentInfo	The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-CHUIDSecurityObject OID.
TE06.01.06.01	10.1.1.4 Verify contents of encapContentInfo	The tester shall validate that the eContent field has been omitted from the encapContentInfo.
TE06.01.07.01	10.2.1.13 Verify digital signature	The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.
TE06.01.08.01	10.1.1.5 Verify crls field omission	The tester shall validate that the crls field has been omitted from the SignedData.
TE06.01.09.01	10.1.1.6 Verify contents of signerInfos	The tester shall validate that only a single SignerInfo exists in the SignedData.
TE06.01.10.01	10.1.1.7 Verify Signer Identifier in SignerInfo	The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier.
TE06.01.11.01	10.1.1.8 Verify Digest Algorithm in SignerInfo	The tester shall validate that the digest algorithm is in accordance with Table 3-2.
TE06.01.12.01	10.1.1.9 Verify message digest signed attribute in SignerInfo	The tester shall validate the presence of a MessageDigest attribute in the signed attributes.
TE06.01.12.02	10.1.1.9 Verify message digest signed attribute in SignerInfo	The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated content of the CHUID, excluding the asymmetric signature field.
TE06.01.13.01	10.1.1.10 Verify PIV signer distinguished name	The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes
TE06.01.13.02	10.1.1.10 Verify PIV signer distinguished name	The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the CHUID

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.01.14.01	10.1.1.11 Verify signature algorithm in SignerInfo	The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.
TE06.01.15.01	10.1.1.12 Verify digital signature	The tester shall validate that the SignedData content type includes the digital signature corresponding to the CHUID

A.4 Biometric Fingerprint for Off-Card Comparison Mapping

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.02.01.01	10.2.1.1: Verify presence of CMS SignedData	The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.
TE06.02.02.01	10.2.1.1: Verify presence of CMS SignedData	The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.
TE06.02.03.01	10.2.1.2: Verify version in SignedData	The tester shall validate the version of the SignedData type is version 3.
TE06.02.04.01	10.2.1.3:Verify digest Algorithm in SignedData	The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.
TE06.02.05.01	10.2.1.4: Verify contents of encapContentInfo	The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.
TE06.02.06.01	10.2.1.4: Verify contents of encapContentInfo	The tester shall validate that the eContent field has been omitted from the encapContentInfo.
TE06.02.07.01	10.2.1.13: Verify digital signature	The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.
TE06.02.07.02	10.2.1.13: Verify digital signature	If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.
TE06.02.08.01	10.2.1.5: Verify crls field omission	The tester shall validate that the crls field has been omitted from the SignedData.
TE06.02.09.01	10.2.1.6: Verify contents of signerInfos	The tester shall validate that only a single SignerInfo exists in the SignedData.
TE06.02.10.01	10.2.1.7: Verify Signer Identifier in SignerInfo	The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier.

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.02.11.01	10.2.1.8: Verify Digest Algorithm in SignerInfo	The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.
TE06.02.12.01	10.2.1.9: Verify message digest signed attribute in SignerInfo	The tester shall validate the presence of a MessageDigest attribute in the signed attributes.
TE06.02.12.02	10.2.1.9: Verify message digest signed attribute in SignerInfo	The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.
TE06.02.13.01	10.2.1.10: Verify PIV signer distinguished name	The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.
TE06.02.13.02	10.2.1.10: Verify PIV signer distinguished name	The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.
TE06.02.14.01	10.2.1.11: Verify FASC-N	The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.
TE06.02.14.02	10.2.1.11: Verify FASC-N	The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in the CHUID.
TE06.02.15.01	10.2.1.12: Verify signature algorithm in SignerInfo	The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.
TE06.02.16.01	10.2.1.13: Verify digital signature	The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.
TE06.02.17.01	10.2.1.14 Verify entryUUID	The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.
TE06.02.17.02	10.2.1.14 Verify entryUUID	The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

A.5 Biometric Facial Image Mapping

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.03.01.01	10.3.1.1: Verify presence of CMS SignedData	The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.03.02.01	10.3.1.1: Verify presence of CMS SignedData	The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.
TE06.03.03.01	10.3.1.2: Verify version in SignedData	The tester shall validate the version of the SignedData type is version 3.
TE06.03.04.01	10.3.1.3: Verify digest Algorithm in SignedData	The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.
TE06.03.05.01	10.3.1.4: Verify contents of encapContentInfo	The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.
TE06.03.06.01	10.3.1.4: Verify contents of encapContentInfo	The tester shall validate that the eContent field has been omitted from the encapContentInfo.
TE06.03.07.01	10.3.1.13: Verify digital signature	The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.
TE06.03.07.02	10.3.1.13: Verify digital signature	If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.
TE06.03.08.01	10.3.1.5: Verify crls field omission	The tester shall validate that the crls field has been omitted from the SignedData.
TE06.03.09.01	10.3.1.6: Verify contents of signerInfos	The tester shall validate that only a single SignerInfo exists in the SignedData.
TE06.03.10.01	10.3.1.7: Verify Signer Identifier in SignerInfo	The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier.
TE06.03.11.01	10.3.1.8: Verify Digest Algorithm in SignerInfo	The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.
TE06.03.12.01	10.3.1.9: Verify message digest signed attribute in SignerInfo	The tester shall validate the presence of a MessageDigest attribute in the signed attributes.
TE06.03.12.02	10.3.1.9: Verify message digest signed attribute in SignerInfo	The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.
TE06.03.13.01	10.3.1.10: Verify PIV signer distinguished name	The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.
TE06.03.13.02	10.3.1.10: Verify PIV signer distinguished name	The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.
TE06.03.14.01	10.3.1.11: Verify FASC-N	The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.
TE06.03.14.02	10.3.1.11: Verify FASC-N	The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in

DTR from Section 6	Test Assertion from Section 10	DTR Description
		the CHUID.
TE06.03.15.01	10.3.1.12: Verify signature algorithm in SignerInfo	The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.
TE06.03.16.01	10.3.1.13: Verify digital signature	The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.
TE06.03.17.01	10.3.1.14 Verify entryUUID	The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.
TE06.03.17.02	10.3.1.14 Verify entryUUID	The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

A.6 Security Object Mapping

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.04.01.01	10.4.1.1: Verify integrity of data element hashes	The tester shall validate that the message digests for the various data objects present in the security object are identical to the message digest of the data object itself.
TE06.04.02.01	10.4.2.1: Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the digital signature has been formatted correctly as a CMS signature as defined in RFC (5652).
TE06.04.03.01	10.4.2.1: Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the CMS digital signature has been implemented as a SignedData type.
TE06.04.04.01	10.4.2.2: Verify version in SignedData	The tester shall validate the version of the SignedData type is version 3.
TE06.04.05.01	10.4.2.3: Verify digest Algorithm in SignedData	The tester shall validate that the digest algorithm value is in accordance with Table 3-2 of SP80078.
TE06.04.06.01	10.4.2.4: Verify contents of encapContentInfo	The tester shall validate that eContentType of the encapContentInfo asserts the id-icao-ldsSecurityObject OID.
TE06.04.07.01	10.4.2.4: Verify contents of encapContentInfo	The tester shall validate that eContent of the encapContentInfo contains the contents of the ldsSecurity object.
TE06.04.08.01	10.4.2.5: Verify certificates field omission	The tester shall validate that the certificates field has been omitted from the SignedData.
TE06.04.09.01	10.4.2.6: Verify Digest	The tester shall validate that the digest algorithm in

DTR from Section 6	Test Assertion from Section 10	DTR Description
	Algorithm in SignerInfo	the SignerInfo is in accordance with Table 3-2 of SP80078.
TE06.04.10.01	10.4.2.7: Verify signature algorithm in SignerInfo	The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.
TE06.04.11.01	10.4.2.8: Verify digital signature	The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed security object.

A.7 Biometric Iris Mapping

DTR from Section 6	Test Assertion from Section 10	DTR Description
TE06.05.01.01	10.5.1.1 Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the digital signature in the CBEFF_SIGNATURE_BLOCK has been formatted correctly as a CMS external signature as defined in RFC 5652.
TE06.05.02.01	10.5.1.1 Verify presence of CMS SignedData asymmetric digital signature	The tester shall validate that the CMS external digital signature has been implemented as a SignedData type.
TE06.05.03.01	10.5.1.2 Verify version in SignedData	The tester shall validate the version of the SignedData type is version 3.
TE06.05.04.01	10.5.1.3 Verify digest Algorithm in SignedData	The tester shall validate that the digest algorithm is in accordance with Table 3-2 of SP80078.
TE06.05.05.01	10.5.1.4 Verify contents of encapContentInfo	The tester shall validate that eContentType of the encapContentInfo asserts the id-PIV-biometricObject OID.
TE06.05.06.01	10.5.1.4 Verify contents of encapContentInfo	The tester shall validate that the eContent field has been omitted from the encapContentInfo.
TE06.05.07.01	10.5.1.13 Verify digital signature	The tester shall validate that there is a single X.509 certificate in the certificates field that can verify the digital signature in the SignerInfo.
TE06.05.07.02	10.5.1.13 Verify digital signature	If the certificates field is omitted, the tester shall validate that the certificate in the SignedData for the CHUID can verify the digital signature in the SignerInfo.
TE06.05.08.01	10.5.1.5 Verify crls field omission	The tester shall validate that the crls field has been omitted from the SignedData.
TE06.05.09.01	10.5.1.6 Verify contents of signerInfos	The tester shall validate that only a single SignerInfo exists in the SignedData.
TE06.05.10.01	10.5.1.7 Verify Signer Identifier in SignerInfo	The tester shall validate that the issuerAndSerialNumber choice has been used for the SignerIdentifier and it corresponds to the to the issuer

DTR from Section 6	Test Assertion from Section 10	DTR Description
		and serialNumber fields found in the X.509 certificate for the entity that signed the biometric data.
TE06.05.11.01	10.5.1.8 Verify Digest Algorithm in SignerInfo	The tester shall validate that the digest algorithm in the SignerInfo is in accordance with Table 3-2 of SP80078.
TE06.05.12.01	10.5.1.9 Verify message digest signed attribute in SignerInfo	The tester shall validate the presence of a MessageDigest attribute in the signed attributes.
TE06.05.12.02	10.5.1.9 Verify message digest signed attribute in SignerInfo	The tester shall validate the value of the MessageDigest attribute against the hash of the concatenated CBEFF_HEADER and the STD_BIOMETRIC_RECORD.
TE06.05.13.01	10.5.1.10 Verify PIV signer distinguished name	The tester shall validate the presence of a pivSigner-DN attribute in the signed attributes.
TE06.05.13.02	10.5.1.10 Verify PIV signer distinguished name	The tester shall validate the value of the pivSigner-DN attribute is the same as the subject name that appears in the certificate that signed the biometric data.
TE06.05.14.01	10.5.1.11 Verify FASC-N	The tester shall validate the presence of a pivFASC-N attribute in the signed attributes.
TE06.05.14.02	10.5.1.11 Verify FASC-N	The tester shall validate the value of the pivFASC-N attribute is the same as the FASC-N that is present in the CHUID.
TE06.05.15.01	10.5.1.12 Verify signature algorithm in SignerInfo	The tester shall validate that the signature algorithm value for RSA with PKCS #1 v1.5 padding specifies the rsaEncryption OID (as per Section 3.2 of RFC 3370) and for ECDSA and RSA with PSS padding, the signatureAlgorithm is in accordance with Table 3-3 of SP80078.
TE06.05.16.01	10.5.1.13 Verify digital signature	The tester shall validate that the SignedData content type includes the digital signature corresponding to the signed biometric data.
TE06.05.17.01	10.5.1.14 Verify entryUUID	The tester shall validate the presence of an entryUUID (OID = 1.3.6.1.1.16.4) attribute in the signed attributes.
TE06.05.17.02	10.5.1.14 Verify entryUUID	The tester shall validate the value of the entryUUID (OID = 1.3.6.1.1.16.4) attribute is the same as the 16-byte representation of the Card UUID value that appears in the GUID data element of the PIV card's CHUID data element.

A.8 PIV Authentication Certificate Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.01.01.01	11.1.1.1: Verify signature algorithm	The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.
TE07.01.02.01	11.1.1.1: Verify signature algorithm	The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.
TE07.01.03.01	11.1.1.2: Verify subject public key algorithm	The tester shall validate that the algorithm used to generate PIV authentication keys are in accordance with Table 3-4 of SP 800-78.
TE07.01.04.01	11.1.1.2: Verify subject public key algorithm	The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the PIV authentication certificate issued by the vendor.
TE07.01.05.01	11.1.2.1 Verify key usage extension	The tester shall validate the assertion of the digitalSignature bit in the keyUsage extension in the PIV authentication certificate issued by the vendor.
TE07.01.06.01	11.1.2.2 Verify id-fpki-common-authentication OID	The tester shall validate the presence of the id-fki-common-authentication OID in the certificatePolicies extension in the PIV authentication certificate issued by the vendor.
TE07.01.07.01	11.1.2.3 Verify authority information access extension	The tester shall validate the presence of an id-ad-ocsp accessMethod in the authorityInfoAccess extension in the PIV authentication certificate issued by the vendor. The tester shall also validate that the accessLocation for this accessMethod uses the URI name form and points to an HTTP accessible OCSP server.
TE07.01.08.01	11.1.2.6 Verify FASC-N and Card UUID	The tester shall validate that the FASC-N and Card UUID in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. The tester shall validate that no other name forms appear in the subjectAltName extension.
TE07.01.09.01	11.1.2.4 Verify interim status extension	The tester shall validate that the piv-interim extension is present in the PIV authentication certificate issued by the vendor.
TE07.01.10.01	11.1.2.9 Verify HTTP URI in cRLDistributionPoints extension field	The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the PIV authentication certificate.
TE07.01.11.01	11.1.2.10 Verify HTTP URI in authorityInfoAccess extension field	The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
TE07.01.12.01	11.1.1.3 Verify public key size	The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.01.13.01	11.1.2.5 Verify asymmetric key pair	The tester shall validate that the public key present in the PIV authentication certificate is part of the key pair corresponding to the private key on the PIV card.
TE07.01.14.01	11.1.2.6 Verify FASC-N and Card UUID	The tester shall validate that the FASC-N in the subjectAltName field in the PIV authentication certificate is the same as the FASC-N present in the CHUID in the PIV card.
TE07.01.14.02	11.1.2.6 Verify FASC-N and Card UUID	The tester shall validate that the Card UUID present in the subjectAltName field is the same as the Card UUID in the CHUID in the PIV card.
TE07.01.15.01	11.1.2.7 Verify expiration dates consistency	The tester shall validate that the expiration of the PIV authentication certificate is not beyond the expiration of the CHUID in the PIV card.
TE07.01.16.01	11.1.2.8 Verify RSA exponent	The tester shall validate that the RSA public key exponent size is equal to 65,537.

A.9 Digital Signature Certificate Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.02.01.01	11.2.1.1: Verify signature algorithm	The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.
TE07.02.02.01	11.2.1.1: Verify signature algorithm	The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.
TE07.02.03.01	11.2.1.2: Verify subject public key algorithm	The tester shall validate that the algorithm used to generate digital signature keys are in accordance with Table 3-4 of SP80078.
TE07.02.04.01	11.2.1.2: Verify subject public key algorithm	The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the digital signature certificate issued by the vendor.
TE07.02.05.01	11.2.2.1 Verify key usage extension	The tester shall validate the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension in the digital signature certificate issued by the vendor.
TE07.02.06.01	11.2.2.5 Verify the policyIdentifier field in the certificatePolicies	The tester shall validate the presence of one of the following OIDs in the certificatePolicies extension in the Digital Signature certificate issued by the vendor: the id-fpki-common-hardware or id-fpki-common-High.
TE07.02.07.01	11.2.2.6 Verify HTTP URI in cRLDistributionPoints extension field	The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Digital signature certificate.

TE07.02.08.01	11.2.2.7 Verify HTTP URI in authorityInfoAccess extension field	The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
TE07.02.09.01	11.2.1.3 Verify public key size	The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.
TE07.02.10.01	11.2.2.2 Verify asymmetric key pair	The tester shall validate that the public key present in the digital signature certificate is part of the key pair corresponding to the private key on the PIV card.
TE07.02.11.01	11.2.2.3 Verify expiration dates consistency	The tester shall validate that the expiration of the digital signature certificate is not beyond the expiration of the CHUID in the PIV card.
TE07.02.12.01	11.2.2.4 Verify RSA exponent	The tester shall validate that the RSA public key exponent size is equal to 65,537.

A.10 Key Management Certificate Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.03.01.01	11.3.1.1 Verify signature algorithm	The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.
TE07.03.02.01	11.3.1.1 Verify signature algorithm	The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.
TE07.03.03.01	11.3.1.2 Verify subject public key algorithm	The tester shall validate that the algorithm used to generate key management keys are in accordance with Table 3-4 of SP80078.
TE07.03.04.01	11.3.1.2 Verify subject public key algorithm	The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the key management certificate issued by the vendor.
TE07.03.05.01	11.3.2.1 Verify key usage extension	The tester shall validate that certificates corresponding to RSA keys assert only the keyEncipherment bit in the keyUsage extension.
TE07.03.06.01	11.3.2.1 Verify key usage extension	The tester shall validate that certificates corresponding to elliptic curve keys assert only the keyAgreement bit in the keyUsage extension.
TE07.03.07.01	11.3.2.4 Verify the policyIdentifier field in the certificatePolicies	The tester shall validate the presence of one of the following OIDs in the certificatePolicies extension in the Key Management certificate issued by the vendor: the id-fpki-common-policy, id-fpki-common-hardware, or id-fpki-common-High.
TE07.03.08.01	11.3.2.5 Verify HTTP URI in cRLDistributionPoints extension field	The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Key management certificate.

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.03.09.01	11.3.2.6 Verify HTTP URI in authorityInfoAccess extension field	The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
TE07.03.10.01	11.3.1.3 Verify public key size	The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.
TE07.03.11.01	11.3.2.2 Verify asymmetric key pair	The tester shall validate that the public key present in the key management certificate is part of the key pair corresponding to the private key on the PIV card.
TE07.03.12.01	11.3.2.3 Verify RSA exponent	The tester shall validate that the RSA public key exponent size is equal to 65,537.

A.11 Card Authentication Certificate Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.04.01.01	11.4.1.1 Verify signature algorithm	The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.
TE07.04.02.01	11.4.1.1 Verify signature algorithm	The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.
TE07.04.03.01	11.4.1.2 Verify subject public key algorithm	The tester shall validate that the algorithm used to generate card authentication keys are in accordance with Table 3-4 of SP80078.
TE07.04.04.01	11.4.1.2 Verify subject public key algorithm	The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the card authentication certificate issued by the vendor.
TE07.04.05.01	11.4.2.1 Verify key usage extension	The tester shall validate the assertion of the digitalSignature bit in the keyUsage extension in the card authentication certificate issued by the vendor.
TE07.04.06.01	11.4.2.2 Verify id-fpki-common-cardAuth OID	The tester shall validate the policyIdentifier field in certificatePolicies has asserted the id-fpki-common-cardAuth OID.
TE07.04.07.01	11.4.2.3 Verify extended key usage extension	The tester shall validate the extKeyUsage is present, is marked as critical, asserts the id-PIV-cardAuth OID, and does not assert any other OIDs.
TE07.04.08.01	11.4.2.4 Verify authority information access extension	The tester shall validate the presence of an id-ad-ocsp accessMethod in the authorityInfoAccess extension in the card authentication certificate issued by the vendor. The tester shall also validate that the accessLocation for this accessMethod uses the URI name form and points to an HTTP accessible OCSP server.

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.04.09.01	11.4.2.7 Verify FASC-N and Card UUID	The tester shall validate that the FASC-N and Card UUID in the subjectAltName field in the Card authentication certificate is the same as the FASC-N and Card UUID present in the CHUID in the PIV card. The tester shall validate that no other name forms appear in the subjectAltName extension.
TE07.04.10.01	11.4.2.5 Verify interim status extension	The tester shall validate that the piv-interim extension is present in the card authentication certificate issued by the vendor.
TE07.04.11.01	11.4.2.9 Verify HTTP URI in cRLDistributionPoints extension field	The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the Card authentication certificate.
TE07.04.12.01	11.4.2.10 Verify HTTP URI in authorityInfoAccess extension field	The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
TE07.04.13.01	11.4.1.3 Verify public key size	The tester shall validate that the public key size is in accordance with Table 3-1 of SP80078.
TE07.04.14.01	11.4.2.6 Verify asymmetric key pair	The tester shall validate that the public key present in the card authentication certificate is part of the key pair corresponding to the private key on the PIV card.
TE07.04.15.01	11.4.2.7 Verify FASC-N and Card UUID	The tester shall validate that the FASC-N in the subjectAltName field in the Card authentication certificate is the same as the FASC-N present in the CHUID in the PIV card.
TE07.04.15.02	11.4.2.7 Verify FASC-N and Card UUID	The tester shall validate that the Card UUID present in the subjectAltName field is the same as the Card UUID in the CHUID in the PIV card.
TE07.04.16.01	11.4.2.8 Verify RSA exponent	The tester shall validate that the RSA public key exponent size is equal to 65,537.

A.12 Secure Messaging Card Verifiable Certificate (CVC) Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.05.01.01	11.5.2.1 Verify signature algorithm	The tester shall validate that signature field in the certificate is in accordance with Table 15 of Part 2 in SP80073 and contains either an ECDSA signature using P-256 if the CardHolderPublicKey is P-256 or P-384 if the CardHolderPublicKey is P-384.
TE07.05.02.01	11.5.2.2 Verify subject public key algorithm	The tester shall validate that the algorithm used to generate card authentication keys are in accordance

DTR from Section 7	Test Assertion from Section 11	DTR Description
		with Table 15 of SP80073 Part 2.
TE07.05.03.01	11.5.1.2 Verify Secure Messaging CVC Profile	The tester shall verify that the Credential Profile Identifier of the secure messaging CVC is 0x80.
TE07.05.04.01	11.5.1.2 Verify Secure Messaging CVC Profile	The tester shall verify that the Issuer Identification Number of the secure messaging CVC is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature.
TE07.05.04.02	11.5.1.2 Verify Secure Messaging CVC Profile	The tester shall verify that if the public key needed to verify the signature on the secure messaging CVC appears in an Intermediate CVC, then the Issuer Identification Number shall be the value of the Subject Identifier in the Intermediate CVC.
TE07.05.05.01	11.5.1.2 Verify Secure Messaging CVC Profile	The tester shall verify that the Role Identifier of the secure messaging CVC is 0x00 for card-application key CVC.
TE07.05.06.01	11.5.1.1 Validate General CVC Format	The tester shall validate that the Subject Identifier of the secure messaging CVC is same 16-byte binary representation of the Card UUID value in the GUID field of the CHUID.
TE07.05.07.01	11.5.2.3 Secure Messaging Cipher Suite Implementation	The tester shall validate that the Secure Messaging key size is in accordance with the interfaces supported by the card and the keys that are present on the card.
TE07.05.09.01	11.5.3.1 Verify asymmetric key pair	The tester shall validate that the public key present in the CVC is part of the key pair corresponding to the secure messaging private key on the PIV card.
TE07.05.10.01	11.5.1.1 Validate General CVC Format 11.5.2.1 Verify signature algorithm 11.5.2.3 Secure Messaging Cipher Suite Implementation	The tester shall validate that the public key is either an X.509 Certificate for Content Signing or an Intermediate CVC. If it is provided in an Intermediate CVC, then the format of the Intermediate CVC shall be as specified in Table 16 of SP80073 Part 2, Section 4.1.5, and the public key required to verify the digital signature of the Intermediate CVC shall be provided in an X.509 Certificate for Content Signing.
TE07.05.11.01	11.5.1.1 Validate General CVC Format	The tester shall validate that the X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card does not be expired.
TE07.05.12.01	11.5.2.2 Verify subject public key algorithm	The tester shall verify that the Public Key object of the secure messaging CVC is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve.

A.13 Intermediate Card Verifiable Certificate (CVC) Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.06.01.01	11.6.2.1 Verify signature	The tester shall validate that the signature field in the

DTR from Section 7	Test Assertion from Section 11	DTR Description
	algorithm	certificate is in accordance with Table 16 of Part 2 in SP80073 and contains an algorithm for RSA with SHA-256 and PKCS #1 v1.5 padding.
TE07.06.02.01	11.6.2.2 Verify subject public key algorithm	The tester shall validate that the algorithm used to generate card authentication keys are in accordance with Table 16 of SP80073 Part 2.
TE07.06.03.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the tag value of the Intermediate CVC is 0x7F21.
TE07.06.04.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the Credential Profile Identifier of the Intermediate CVC is 0x80.
TE07.06.05.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the Issuer Identification Number of the Intermediate CVC is the leftmost 8 bytes of the subjectKeyIdentifier in the content signing certificate needed to verify the signature.
TE07.06.06.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the Subject Identifier of the Intermediate CVC is the leftmost 8 bytes of the SHA-1 hash of the Public Key object.
TE07.06.07.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the Algorithm OID of the Intermediate CVC is either 0x2A8648CE3D030107 for ECDH (Curve P-256) or 0x2B81040022 for ECDH (Curve P-384).
TE07.06.08.01	11.6.1.2 Verify Intermediate CVC Profile	The tester shall verify that the Role Identifier of the Intermediate CVC is 0x12 for card-application root CVC.
TE07.06.09.01	11.6.1.1 Validate General CVC Format	The tester shall validate that the X.509 Certificate for Content Signing needed to verify the digital signature of a secure messaging CVC or Intermediate CVC of a valid PIV card is not expire.
TE07.06.10.01	11.6.2.2 Verify subject public key algorithm	The tester shall verify that the Public Key object of the Intermediate CVC is encoded in tag 0x86 with a value of 04||X||Y, where X and Y are the coordinates of the point on the curve.
TE07.06.11.01	11.6.2.1 Verify signature algorithm	The tester shall verify that the public key in the Intermediate CVC used to verify the signature on the secure messaging CVC shall conform to Table 16 SP80073 Part 2, Section 4.1.5.

A.14 X.509 Certificate for Content Signing Mapping

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.07.01.01	11.7.1.1 Verify signature algorithm	The tester shall validate that the signature algorithm of the certificate is listed in Table 3-3 of SP80078 and is not sha1WithRSAEncryption.
TE07.07.02.01	11.7.1.1 Verify signature algorithm	The tester shall validate that the correctness of the values of the AlgorithmIdentifier fields.

DTR from Section 7	Test Assertion from Section 11	DTR Description
TE07.07.03.01	11.7.1.2 Verify subject public key algorithm	The tester shall validate that the algorithm used to generate the X.509 Certificate for Content Signing are in accordance with Table 3-4 of SP80078.
TE07.07.04.01	11.7.1.2 Verify subject public key algorithm	The tester shall validate the correctness of the values of the parameters field of the algorithm of the subjectPublicKeyInfo field in the X.509 Certificate for Content Signing issued by the vendor.
TE07.07.05.01	11.7.2.1 Verify key usage extension	The tester shall validate the assertion of the digitalSignature bit and the nonRepudiation bit in the keyUsage extension in the X.509 Certificate for Content Signing issued by the vendor.
TE07.07.06.01	11.7.2.4 Verify the policyIdentifier field in the certificatePolicies	The tester shall validate that the signatures created before October 15, 2015, the public key required to verify the digital signature is provided in the certificates field of the CMS external digital signature in a content signing certificate, which is an X.509 digital signature certificate issued under the id-fpki-common-hardware, the public key required to verify the digital signature shall be provided in the certificates field of the CMS external digital signature in a content signing certificate, which is an X.509 digital signature certificate issued under the id-fpki-common-piv-contentSigning policy of [COMMON]. The content signing certificate also includes an extended key usage (extKeyUsage) extension asserting id-PIV-content-signing.
TE07.07.07.01	11.7.2.5 Verify HTTP URI in cRLDistributionPoints extension field	The tester shall verify that the cRLDistributionPoints field shall contain a URI that uses HTTP and points to a file that has an extension of ".crl" containing the DER encoded CRL (see RFC 2585) for status information on the X.509 Certificate for Content Signing.
TE07.07.08.01	11.7.2.6 Verify HTTP URI in authorityInfoAccess extension field	The tester shall validate that the authorityInfoAccess field contains an id-ad-caIssuers (1.3.6.1.5.5.7.48.2) accessMethod. The access location is a URI using HTTP and points to a file that has an extension of ".p7c" containing a certs-only CMS message (see RFC 3851).
TE07.07.09.01	11.7.1.3 Verify public key size	The tester shall validate that the public key size is in accordance with Table 3-2 2of SP80078.
TE07.07.10.01	11.7.2.3 Verify RSA exponent	The tester shall validate that the RSA public key exponent size is equal to 65,537.
TE07.07.11.01	11.7.2.2 Verify expiration dates consistency	The tester shall verify that the X.509 Certificate for Content Signing is not expired.

Appendix B—Bibliography

Citation Code	Document
SP80073	NIST Special Publication 800-73-4, Interfaces for Personal Identity Verification, May 2013 or as amended. (See http://csrc.nist.gov)
SP80076	NIST Special Publication 800-76-2, Biometric Data Specification for Personal Identity Verification, July 2012 or as amended. (See http://csrc.nist.gov)
SP80078	Draft NIST Special Publication 800-78-4, Cryptographic Algorithms and Key Sizes for Personal Identity Verification, May 2013 or as amended. (See http://csrc.nist.gov)
SP80085A	NIST Special Publication 800-85A-2, PIV Card Application and Middleware Interface Test Guidelines, July 2010 or as amended. (See http://csrc.nist.gov)
FIPS201	FIPS 201-2, Personal Identity Verification (PIV) National Institute of Standards and Technology), September 2013. (See http://csrc.nist.gov)
FINGSTD	INCITS 381-2004, American National Standard for Information Technology - Finger Image-Based Data Interchange Format.
HSPD12	HSPD 12, Policy for a Common Identification Standard for Federal Employees and Contractors, August 27, 2004. (See http://www.dhs.gov/homeland-security-presidential-directive-12)
MINUSTD	INCITS 378-2004, American National Standard for Information Technology - Finger Minutiae Format for Data Interchange.
FACESTD	INCITS 385-2004, American National Standard for Information Technology - Face Recognition Format for Data Interchange.
CBEFF	INCITS 398-2005, American National Standard for Information Technology - Common Biometric Exchange Formats Framework (CBEFF).
EBTS	AFIS-DOC-01078-9.1 CJIS-RS-0010 (V9.4) – Electronic Biometric Transmission Specification, Criminal Justice Information Services, Federal Bureau of Investigation, Department of Justice, May 25, 2010. Implementers should consult https://www.fbibiospecs.org/ or request the full EBTS documentation from the FBI..
IRISSTD	ISO/IEC 19794-6:2011 Information technology -- Biometric data interchange formats -- Part 6: Iris image data This document revises and replaces the 2005 iris standard. Published September 29, 2011. This standard is being amended, with publication expected 2014. The amendment does two things: It establishes conformance tests for 19794-6 iris records and, in support of that, clarifies the Image Type 7 appearance requirements. In particular it emphasizes that the sclera shall be masked. The title of the amendment is: Information Technology — Biometric data interchange formats — Part 6: Iris image format - Amendment 1: Conformance testing methodologies.
ISO7816	ISO/IEC 7816 (Parts 4, 5, 6, 8, and 9), Information technology — Identification cards — Integrated circuit(s) cards with contacts.
TIG SCEPACS	*Technical Implementation Guidance: Smart Card Enabled Physical Access Control Systems*, Version 2.2, The Government Smart Card Interagency Advisory Board's Physical Security Interagency Interoperability Working Group, July 27, 2004. (see

Citation Code	Document
	https://www.idmanagement.gov/sites/default/files/documents/TIG_SCEPACS_v2.2_0.pdf)
X509	X.509 Certificate Policy.
X509 Extensions	X.509 Certificate and Certificate Revocation List (CRL) Extensions Profile for the Shared Service Providers (SSP) Program, January 7, 2008.

2333

2334

Appendix C—Glossary of Terms and Acronyms

C.1 Glossary of Terms

Term	Meaning
Offline Test	Offline tests use previously captured images as inputs to core biometric implementations. Such tests are repeatable and can readily be scaled to very large populations and large numbers of competing products. They institute a level-playing field and produce robust estimates of the core biometric power of an algorithm. This style of testing is particularly suited to interoperability testing of a fingerprint template (see [ISOSWAP]).
Scenario Test	Scenario testing is intended to mimic an operational application and simultaneously institute controls on the procedures. Scenario testing requires members of a human test population to transact with biometric sensors. Scenario tests are appropriate for capturing and assessing the effects of interactions human users have with biometric sensors and interfaces.
Operational Test	Operational tests involve a deployed system and are usually conducted to measure in-the-field performance and user-system interaction effects. Such tests require the members of a human test population to transact with biometric sensors. False acceptance rates may not be measurable, depending on the controls instituted.
Interoperability Test	Interoperability tests measure the performance associated with the use of standardized biometric data records in a multiple vendor environment. It involves the production of the templates by N enrollment products and authentication of these against images processed by M others.
Template Matcher	In the PIV context a matcher is a software library providing for the comparison of images conformant to FINGSTD and templates conformant to MINUSTD. The output of the matcher, a similarity score, will be the basis of accept or reject decision.
Template Generator	In the PIV context a template generator is a software library providing facilities for the conversion of images conformant to FINGSTD to templates conformant to MINUSTD for storage on the PIV card.

C.2 Acronyms

ANSI American National Standards Institute

BDB Biometric Data Block

BER-TLV Basic Encoding Rules Tag-Length-Value

CBEFF Common Biometric Exchange Formats Framework

CCC Card Capability Container

CHUID Cardholder Unique Identifier

CMS Cryptographic Message Syntax

CRL Certificate Revocation List

DTR Derived Test Requirement

ECDSA	Elliptic Curve Digital Signature Algorithm
FICC	Federal Identity Credentialing Committee
FIPS	Federal Information Processing Standards
FISMA	Federal Information Security Management Act
GUID	Global Unique Identification Number
HSPD	Homeland Security Presidential Directive
HTTP	Hypertext Transfer Protocol
INCITS	International Committee for Information Technology Standards
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OMB	Office of Management and Budget
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
PSS	Probabilistic Signature Scheme
RSA	Rivest Shamir Adleman
SBH	Signature Block Header
SHA	Secure Hash Algorithm
SP	Special Publication
SSP	Shared Service Providers
TIG SCEPACS	Technical Implementation Guidance Smart Card Enabled Physical Access Control System
URI	Uniform Resource Identifier