Apple iPadOS 18: iPad Security Target

Version 1.1
September 16, 2025

Apple Inc.
One Apple Park Way
Cupertino, CA 95014

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision.

Trademarks

Apple's trademarks applicable to this document are listed in https://www.apple.com/legal/intellectual-property/trademark/appletmlist.html.

Other company, product, and service names may be trademarks or service marks of others.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Table of Contents

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

List of Figures

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

List of Tables

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

1. Introduction

1.1. Security Target (ST) Identification

Security Target identifiers	
ST Title	Apple iPadOS 18: iPad Security Target
ST Version	1.1
ST Date	September 16, 2025
Sponsor	Apple Inc.
Developer	Apple Inc.
Validation Body	US NIAP
Keywords	Apple, iPad, iPadOS, mobile device

1.2. Target of Evaluation (TOE) Identification

The TOE is Apple iPadOS 18: iPad with iPadOS version 18.3.1.

1.3. TOE Type

The TOE type is mobile device.

1.4. TOE Overview

This document is the Common Criteria (CC) Security Target for the following:

- Apple iPadOS 18: iPad

The devices were tested using the following operating system release:

- iPadOS 18.3.1

The iPad hardware platforms covered by this evaluation are provided in Appendix A.1 "Devices Covered by this Evaluation". The listed hardware platforms, with Apple iPadOS 18.3.1 installed, were evaluated as mobile devices in exact conformance with the protection profiles listed in Section 2 "CC Conformance Claim".

The TOE is a series of Apple iPad mobile devices running the iPadOS 18 operating system and includes components such as a Mobile Device Management (MDM) Agent, VPN client, WLAN client, Bluetooth, and biometric authentication factors (BAFs).

For simplicity, the term "TOE OS" is used throughout this document and refers to the OS release listed above.
The TOE OS manages the device hardware, provides MDM Agent functionality, and provides the technologies required to implement native applications (a.k.a. apps). (A native app is an app compiled to run on a specific mobile platform.) It provides a built-in MDM framework application programming interface (API), giving management features that may be utilized by external MDM solutions, allowing enterprises to use profiles to control some of the device settings.

The TOE OS provides a consistent set of capabilities allowing the supervision of enrolled devices. This includes the preparation of devices for deployment, the subsequent management of the devices, and the termination of management.

The TOE provides cryptographic services for the encryption of data at rest (DAR) within the TOE, for secure communication channels, for protection of configuration profiles, and for use by apps. User data protection is provided by encrypting user data, restricting access by apps, and restricting access until the user has been successfully authenticated.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

User identification and authentication is provided by a user-defined passcode (and supplemented by biometric technologies) where the minimum length of the passcode, passcode rules, and the maximum number of consecutive failed authentication attempts can be configured by an administrator.

Security management capabilities are provided to users via the user interface of the device and to administrators through the installation of configuration profiles on the device. This installation can be done using the Apple Configurator 2 tool or by using an MDM system.

The TOE protects itself by having its own code and data protected from unauthorized access (using hardware- provided memory protection features), by encrypting internal user and TOE Security Functionality (TSF) data using TSF protected keys and encryption/decryption functions, by conducting self-tests, by ensuring the integrity and authenticity of TSF updates and downloaded apps, and by locking the TOE upon user request or after a defined time of user inactivity.

In addition, the TOE implements multiple cryptographic protocols that can be used to establish a trusted channel to other IT entities.

The MDM Agent provides secure alerts to the MDM Server indicating status events.

From a hardware perspective, the TOE devices use Apple System on Chip (SoC) technology. Each SoC contains multiple components including the following:

- Application processor
- Secure Enclave
    - Secure Enclave Processor (SEP)
    - True Random Number Generator (TRNG)
    - Secure Enclave AES Engine

The Secure Enclave is a dedicated secure subsystem integrated into the SoC. It is isolated from the application processor to provide an extra layer of security and is designed to keep sensitive user data secure even when the application processor's kernel becomes compromised.

The Secure Enclave includes the SEP, which runs Secure Enclave Processor OS (sepOS). The sepOS is bundled with the TOE OS. The Secure Enclave also includes a hardware TRNG and hardware AES engine. The TRNG and AES engine are directly connected to the SEP and are only accessible through the SEP.

1.5. TOE Description

1.5.1. Introduction

1.5.1.1. General Information

The TOE is intended to be used as a communication solution providing mobile staff connectivity to enterprise data.

The TOE hardware is uniquely identified by the model number (see Appendix A.1 "Devices Covered by this Evaluation") and the TOE software is identified by its version number. The TOE includes documentation that is listed in Section 1.5.3 "TOE Documentation".

The TOE provides wireless connectivity and includes support for virtual private network (VPN) connections; for access to the protected enterprise network, enterprise data and apps; and for communicating with other mobile devices.

The TOE does not include the user apps that run on top of the operating system but does include controls that limit the behavior of the apps and enforce data segregation and impermeability across apps by establishing containerization

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

principles. The TOE may be used as a mobile device within an enterprise environment where the configuration of the device is managed through an MDM solution.

1.5.1.2. Obtaining the Mobile Devices

Customers can acquire the TOE devices through various distribution channels.

Normal distribution channel

Individual customers can obtain the TOE devices from the following sources:

- The Apple Store (physical store or online at https://www.apple.com/store)
- Apple retailers
- Service carriers (e.g., AT&T, Verizon, T-Mobile)
- Resellers

Business-specific distribution channel

Business customers can use the link https://www.apple.com/retail/business/

Government-specific distribution channel

Government customers can use the link https://www.apple.com/r/store/government/

Apple Store Catalog for large customers

Large customers can have their own Apple Store Catalog for their employees to purchase devices directly from Apple under their corporate employee purchase program.

1.5.1.3. Obtaining Software Updates

The TOE supports software updates through both wireless and wired connections. Software update availability can be prompted on the device with a message pushed in the Notification Center or through the Settings » General » Software Update interface. Installation of the latest version of the operating system can be performed automatically (if Automatic Updates is enabled), manually from the device, manually using Finder on macOS 10.15 or later, or manually using iTunes on macOS 10.14 or earlier and on a PC.

1.5.1.4. Supervising and Configuring the Mobile Devices

In the evaluated configuration, the TOE is managed by an MDM solution that enables an enterprise to control and administer the TOE instances that are enrolled in the MDM solution. The TOE provides an interface allowing the enterprise to supervise devices under the enterprise's control.

Supervision gives enterprises greater control over the TOE devices for which they are responsible. With supervision, the administrator can apply extra restrictions like turning off AirDrop or preventing access to the App Store. It also provides additional device configurations and features, like silently updating apps or filtering web usage.

The TOE needs to be configured by an administrator to operate in compliance with the requirements defined in this Security Target. The evaluated configuration for this includes the following:

- The requirement to define a passcode for user authentication
- The specification of a passcode policy defining criteria on the minimum length and complexity of a passcode
- The specification of the maximum number of consecutive failed attempts to enter the passcode
- The specification of the session locking policy
- The specification of the audio and video collection devices allowed

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- The specification of the VPN connection
- The specification of the external storage via device connector policy
- The specification of the wireless networks allowed
- The requirement of the certificates in the trust anchor database

1.5.2. TOE Architecture

The operating system part of the TOE acts as an intermediary between the underlying hardware and the apps running on the TOE. Apps do not talk to the underlying hardware directly. Instead, they communicate with the hardware through a set of well-defined system interfaces. These interfaces make it easy to write apps that work consistently on devices having different hardware capabilities.

The implementation of the TOE OS can be viewed as a set of layers, which are shown in Figure 1 below. Lower layers contain fundamental services and technologies. Higher-level layers build upon the lower layers and provide more sophisticated services and technologies.

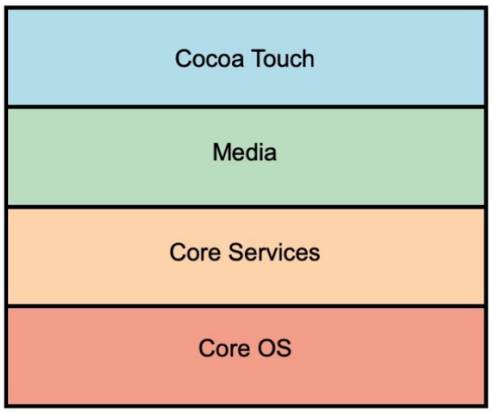

Figure 1: TOE OS Layers

The **Cocoa Touch layer** contains key frameworks for building apps. These frameworks define the appearance of apps. They also provide the basic app infrastructure and support for key technologies such as multitasking, touch-based input, push notifications, and many high-level system services. When designing apps, one should investigate the technologies in this layer first to see if they meet the needs of the developer.

The **Media layer** contains the graphics, audio, and video technologies you use to implement multimedia experiences in apps.

The **Core Services layer** contains fundamental system services for apps. Key among these services are the Core Foundation and Foundation frameworks, which define the basic types that all apps use. This layer also contains individual technologies to support features such as location, iCloud, social media, and networking.

This layer also implements data protection functions that allow apps that work with sensitive user data to take advantage of the built-in encryption available on some devices. When an app designates a specific file as protected, the system stores that file in an encrypted format. While the device is locked, the contents of the file are inaccessible to both the app and to any potential intruders. However, when the device is unlocked by the user, a decryption key is created to allow the app to access the file. Other levels of data protection are also available.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The **Core OS layer** contains the low-level features that most other technologies are built upon. Even if an app does not use these technologies directly, they are most likely being used by other frameworks. And in situations where an app needs to explicitly deal with security or communicating with an external hardware accessory, it does so by using the frameworks in this layer. This layer provides the following security-related frameworks:

- The Generic Security Services Framework, providing services as specified in RFC 2743 "Generic Security Service Application Program Interface Version 2, Update 1" and RFC 4401 "A Pseudo-Random Function (PRF) API Extension for the Generic Security Service Application Program Interface (GSS-API)".
- The Local Authentication Framework.
- The Network Extension Framework, providing support for configuring and controlling VPN tunnels.
- The Security Framework, providing services to manage and store certificates, public and private keys, and trust policies (this framework also provides the Common Crypto library for symmetric encryption and hash-based message authentication codes).
- The System Framework, providing the kernel environment, drivers, and low-level UNIX interfaces (the kernel manages the virtual memory system, threads, file system, network, and inter-process communication and is therefore responsible for separating apps from each other and controlling the use of low-level resources).

1.5.2.1. Physical Boundaries

The TOE's physical boundaries are those of the mobile devices.

1.5.2.2. Security Functions provided by the TOE

The TOE provides the security functionality required by the protection profiles listed in Section 2 "CC Conformance Claim".

1.5.2.2.1. Security Audit

The TOE provides the ability for responses to be sent from the MDM Agent to the MDM Server. These responses are configurable by the organization by deploying MDM Enrollment Profiles.

1.5.2.2.2. Cryptographic Support

The TOE provides the security functionality required by the protection profiles listed in Section 2 “CC Conformance Claim”.

- Apple corecrypto Module v18 [Apple silicon, User, Software, SL1] (User Space)
- Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1] (Kernel Space)
- Apple corecrypto Module v18 [Apple silicon, Secure Key Store, Hardware, SL2] (SKS)

The Apple corecrypto Module v18 [Apple silicon, User, Software, SL1] is a dynamically loadable library that resides within the TOE OS user space. The library is loaded into an app running in user space to provide cryptographic functions.

Figure 2 below shows the logical boundary of the Apple corecrypto Module v18 [Apple silicon, User, Software, SL1] within the TOE.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

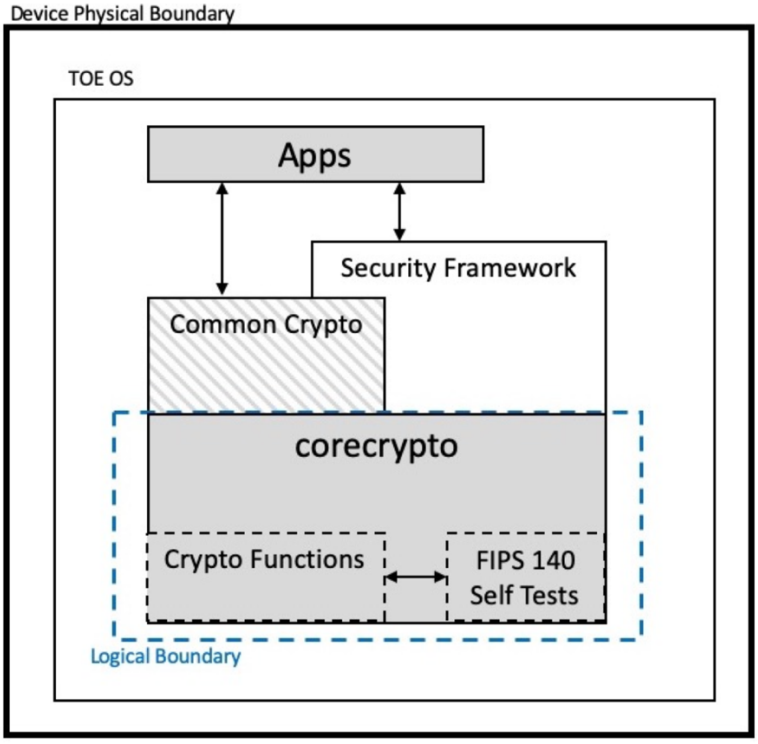

Figure 2: Block Diagram of Apple corecrypto Module v18 [Apple silicon, User, Software, SL1]

Note that the Wi-Fi chip performs the bulk AES cryptography for Wi-Fi and Bluetooth communications. See Section 7.1.8.3 "Wireless LAN (WLAN)" for more detail.

The functions listed below are used to implement the security protocols supported as well as for the encryption of data at rest:

- Random number generation
- Data encryption/decryption
- Signature generation/verification
- Message digest
- Message authentication
- Key generation
- Key wrapping

The Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1] is a TOE OS kernel extension (KEXT) optimized for library use within the TOE OS kernel. Once the module is loaded into the kernel, its cryptographic functions are made available to TOE OS Kernel services only.

Figure 3 below shows the logical boundary of the Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1] within the TOE.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

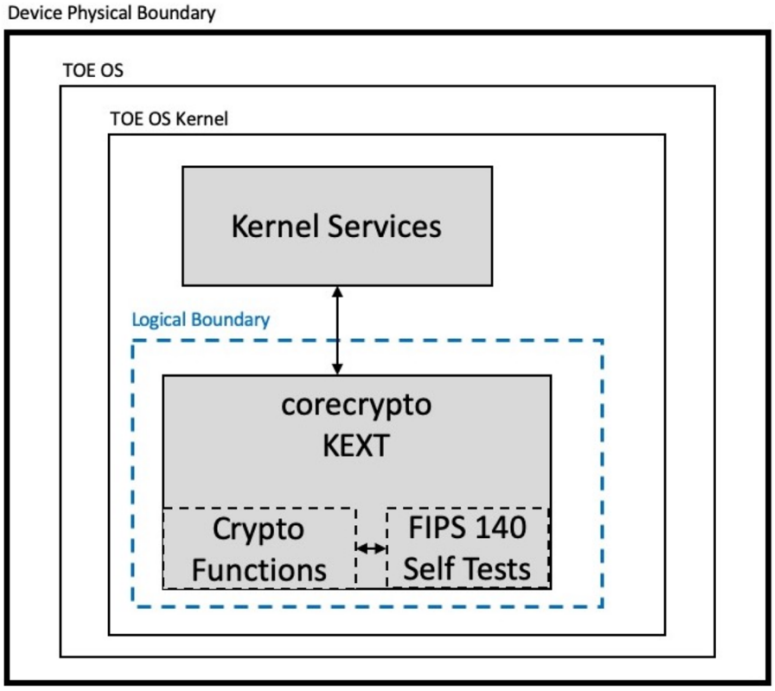

Figure 3: Block Diagram of Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1]

The functions listed below are used to implement the security protocols supported as well as for the encryption of data at rest:

- Random number generation
- Data encryption/decryption
- Signature generation/verification
- Message digest
- Message authentication
- Key generation
- Key wrapping

The Apple corecrypto Module v18 [Apple silicon, SKS, Hardware, SL2] is a single-chip standalone hardware cryptographic module SoC/System-in-Package (SiP) running on a multi-chip device and provides services intended to protect data in transit and at rest. It contains both firmware and hardware cryptographic algorithm implementations. (The Secure Key Store is also known as the SKS.)

The cryptographic services provided by the module are the following:

- Random number generation
- Data encryption/decryption
- Signature generation/verification
- Message digest
- Message authentication
- Key derivation (PBKDF2)
- Key generation
- Key wrapping

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Figure 4 below shows the logical boundary of the Apple corecrypto Module v18 [Apple silicon, SKS, Hardware, SL2] within the TOE.

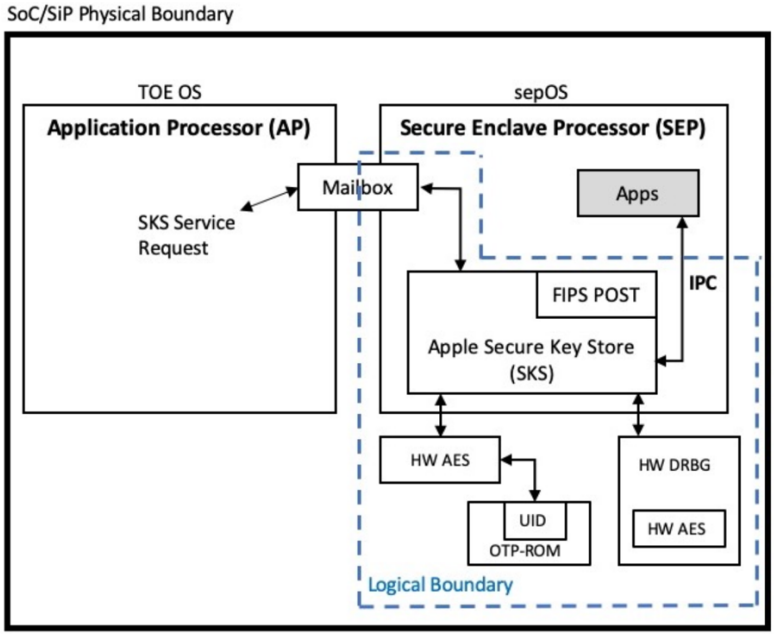

Figure 4: Block Diagram of Apple corecrypto Module v18 [Apple silicon, SKS, Software, SL1]

In the figure above, HW is shorthand for hardware, POST is shorthand Pre-Operational Self Tests, OTP-ROM is shorthand for One Time Programmable Read-Only Memory, and UID is shorthand for Unique ID. Note that the "Apps" in the sepOS are Apple-only developed apps. They are not user-created apps.

1.5.2.2.3. User Data Protection

User data in files is protected using cryptographic functions, ensuring this data remains protected even if the device gets lost or is stolen. Critical data (like passcodes used by apps or application-defined cryptographic keys) can be stored in the keychain, which provides additional protection. Passcode protection and encryption ensure that data at rest remains protected even in the case of the device being lost or stolen.

The Secure Enclave Processor (SEP), a separate coprocessor that executes a stand-alone operating system and has separate memory, provides protection for critical security data such as keys.

Data is protected such that only the app that owns the data can access it.

1.5.2.2.4. Identification and Authentication

The user must authenticate using a passcode or a biometric feature (face, fingerprint) to use the device except when performing the following:

- Accessing Medical ID information
- Answering calls
- Making emergency calls
- Using the cameras (unless their use is generally disallowed)
- Using the flashlight

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- Using the control center
- Using the notification center
- Viewing widgets in Today View and Search

The user is required to use the passcode authentication mechanism under the following conditions:

- Turn on or restart the device
- Press the Home button or swipe up to unlock your device (configurable)
- Update software
- Erase the device
- View or change passcode settings (including biometric enrollment)
- Install iPadOS configuration profiles

The passcode can be configured for a minimum length, for dedicated passcode policies, and for a maximum lifetime. When entered, passcodes are obscured and the frequency of entering passcodes is limited as well as the number of consecutive failed attempts of entering the passcode.

The TOE also enters a locked state after a (configurable) time of user inactivity and the user is required to either enter his passcode or use biometric authentication (fingerprint or face) to unlock the TOE.

The TOE's biometric face authentication is known as Face ID and its fingerprint authentication is known as Touch ID. There are also multiple generations of these BAFs.

External entities connecting to the TOE via a secure protocol (e.g., Transport Layer Security (TLS), Extensible Authentication Protocol Transport Layer Security (EAP-TLS), IPsec) can be authenticated using X.509 certificates. The TOE also supports the usage of Post-quantum Pre-shared Keys in the IKEv2 protocol.

1.5.2.2.5. Security Management

The security function listed in the tables in Section 6 can be managed either by the user or by an authorized administrator through an MDM system. These tables identify the functions that can be managed and indicate if the management can be performed by the user, by the authorized administrator, or both.

1.5.2.2.6. Protection of the TSF

Some of the functions the TOE implements to protect the TSF and TSF data are as follows:

- Protection of cryptographic keys. The keys used for TOE internal key wrapping and for the protection of data at rest are not exportable. There are provisions for fast and secure wiping of key material.
- Use of memory protection and processor states to separate apps and protect the TSF from unauthorized access to TSF resources. In addition, each device includes a separate system called the SEP which is the only system that can use the Root Encryption Key (REK). The SEP is a separate CPU that executes a stand-alone operating system and has separate memory.
- Digital signature protection of the TSF image. All updates to the TSF need to be digitally signed.
- Software/firmware integrity self-test upon startup. The TOE will not go operational when this test fails.
- Digital signature verification for apps.
- Access to defined TSF data and TSF services only when the TOE is unlocked.

1.5.2.2.7. TOE Access

The TSF provides functions to lock the TOE upon request and after an administrator-configurable time of inactivity. Access to the TOE via a wireless network is controlled by user/administrator defined policy.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

1.5.2.2.8. Trusted Path/Channels

The TOE supports the use of the following cryptographic protocols that define a trusted channel between itself and another trusted IT product:

- IEEE 802.11-2012
- IEEE 802.11ac-2013 (a.k.a. Wi-Fi 5)
- IEEE 802.11ax (a.k.a. Wi-Fi 6, Wi-Fi 6E)
- IEEE 802.1X
- EAP-TLS
- TLS
- IPsec
- Bluetooth
- HTTPS

1.5.3. TOE Documentation

The TOE documentation includes the following reference(s):

Reference	Document	Location
Mobile Device Administrator Guidance		
[CCGUIDE]	Apple iOS 18: iPhone and Apple iPadOS 18: iPad Common Criteria Configuration Guide	https://www.niap-ccevs.org/products/11624

Table 1: TOE Documents

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

2. CC Conformance Claim

This Security Target is CC Part 2 extended and CC Part 3 extended. Common Criteria [CC] version 3.1 revision 5 is the basis for this conformance claim.

This Security Target claims exact conformance to the "PP-Configuration for Mobile Device Fundamentals, Biometric enrollment and verification – for unlocking the device, Bluetooth, MDM Agents, Virtual Private Network (VPN) Clients, and WLAN Clients, version 1.1 [PP-Config] (CFG_MDF-BIO-BT-MDMA-VPNC-WLANC_V1.1). The PP configuration includes:

- [MDF]: Base-PP: Protection Profile for Mobile Device Fundamentals. Version 3.3 (PP_MDF_V3.3) as of 2022-09-12.
- [BIO]: PP-Module: collaborative PP-Module for Biometric enrolment and verification - for unlocking the device - [BIOPP-Module]. Version 1.1 (MOD_CPP_BIO_V1.1) as of 2022-09-12.
- [BT]: PP-Module for Bluetooth. Version 1.0 (MOD_BT_V1.0) as of 2021-04-15.
- [Agent]: PP-Module for MDM Agents. Version 1.0 (MOD_MDM_AGENT_V1.0) as of 2019-04-25.
- [VPNC]: PP-Module for Virtual Private Network (VPN) Clients. Version 2.5 (MOD_VPNC_V2.5) as of 2024-06-24.
- [WLANC]: PP-Module for WLAN Clients. Version 1.0 (MOD_WLANC_V1.0) as of 2022-03-31.

In addition, this ST claims conformance to the following functional package:

- [TLSPKG]: Functional Package for Transport Layer Security (TLS). Version 1.1 (PKG_TLS_V1.1) as of 2019-03-01.

The following sections describes the use cases that each document covers and the technical decisions applied.

2.1. Base-PP: Protection Profile for Mobile Device Fundamentals [MDF]

This document claims conformance to the following MDF Use Cases:

- **[Use Case 3] Personally-owned device for personal and enterprise use**: A personally-owned device that is used for both personal activities and enterprise data is commonly called Bring Your Own Device (BYOD). The device may be provisioned for access to enterprise resources after significant personal usage has occurred.
- **[Use Case 4] Personally-owned device for personal and limited enterprise use**: A personally-owned device that is used for both personal activities and enterprise data is commonly called Bring Your Own Device (BYOD). This device may be provisioned for limited access to enterprise resources such as enterprise email.

Table 2 contains the NIAP Technical Decisions (TDs) for this protection profile at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0934	A greater minimum entropy is required when CTR_DRBG is selected in FCS_RBG_EXT.1.2.	Yes	
TD0871	Updating FIPS 186-4 to 186-5 in PP_MDF_V3.3	Yes	
TD0844	Addition of Assurance Package for Flaw Remediation V1.0 Conformance Claim	No	The ST does not claim conformance to Assurance Package for Flaw Remediation V1.0.
TD0724	Format corrections for FAU_GEN.1.1 in MDF 3.3	Yes	
TD0704	Part 3 (Extended) in CC Conformance Claims for MDF 3.3	Yes	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0689	RFC Update in FIA_X509_EXT.1 for MDF PP v3.3	Yes	
TD0677	Correction to Symbol in FCS_RBG_EXT.1 Test EA for MDF 3.3	Yes	

Table 2: NIAP TDs for MDF

2.2. PP-Module: collaborative PP-Module for Biometric enrolment and verification – for unlocking the device [BIO]

This document claims conformance to the following Biometric Use Case:

- **[Use Case 1] Biometric verification for unlocking the computer**: This use case is applicable for any computers such as a desktop, laptop, tablet or smartphone that implement biometric enrolment and verification functionality.

Table 3 contains the NIAP Technical Decisions (TDs) for this cPP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0892	Update for PP_MDF_V3.3 References for Biometric System Configuration	Yes	
TD0714	Minor discrepancy in posted Biometrics cPP-Module	Yes	
TD0700	BIT Technical Decision for incorrect SFR reference in SD	Yes	

Table 3: NIAP TDs for BIO

2.3. PP-Module: Bluetooth [BT]

This document claims conformance to the following Bluetooth Use Case:

- **[Use Case 2] Mobile Device**: This use case is for a Bluetooth TOE that is part of a mobile operating system that runs on a mobile device. Specifically, the Bluetooth TOE is expected to be part of the mobile operating system itself and not a standalone third-party application that is acquired from the mobile vendor's application store.

Table 4 contains the NIAP Technical Decisions (TDs) for this PP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0707	Formatting corrections for MOD_BT_V1.0	Yes	
TD0685	BT missing multiple SFR-to-Obj mappings	Yes	
TD0671	Bluetooth PP-Module updated to allow for new PP and PP-Module Versions	Yes	
TD0650	Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	No	MOD_VPNC_V2.3 and 2.4 are not used by this ST.
TD0645	Bluetooth audit details	Yes	
TD0640	Handling BT devices that do not support encryption	Yes	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0600	Conformance claim sections updated to allow for MOD_VPNC_V2.3	No	MOD_VPNC_V2.3 is not used by this ST.

Table 4: NIAP TDs for BT

2.4. PP-Module: MDM Agents [Agent]

This document claims conformance to the following MDM Agents Use Case:

- **[Use Case 3] Personally owned device for personal and enterprise use**: A personally owned device, which is used, for both personal activities and enterprise data is commonly called Bring Your Own Device (BYOD). The device may be provisioned for access to enterprise resources after significant personal usage has occurred.
- **[Use Case 4] Personally owned device for personal and limited enterprise use**: A personally owned device may also be given access to limited enterprise services such as enterprise email.

Table 5 contains the NIAP Technical Decisions (TDs) for this PP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0755	MDM-Agent Policy Authenticity	Yes	
TD0673	MDM-Agent PP-Module updated to allow for new PP and PP-Module Versions	Yes	
TD0660	Mislabeled SFRs in MDM Agent Auditable Events Table	Yes	
TD0650	Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	No	MOD_VPNC_V2.3 and 2.4 are not used by this ST.
TD0600	Conformance claim sections updated to allow for MOD_VPNC_V2.3	No	MOD_VPNC_V2.3 is not used by this ST.
TD0497	SFR Rationale, Consistency of SPD, and Implicitly Satisfied SFRs	Yes	

Table 5: NIAP TDs for Agent

2.5. PP-Module: Virtual Private Network (VPN) Clients [VPNC]

This document claims conformance to the following VPN Client Use Case:

- **[Use Case 1] TOE to VPN Gateway**: A VPN client allows users on the TOE platform to establish an encrypted IPsec tunnel across a less trusted, often unprotected, public network to a private network.

Table 6 contains the NIAP Technical Decisions (TDs) for this PP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0890	RFC 8784 Optional in VPNC 2.5	Yes	
TD0877	Updating FIPS 186-4 to 186-5 in MOD_VPNC_V2.5	Yes	
TD0875	Correction to FCS_IPSEC_EXT.1.11 TSS Activity	Yes	

Table 6: NIAP TDs for VPNC

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

2.6. PP-Module: WLAN Clients [WLANC]

Table 7 contains the NIAP Technical Decisions (TDs) for this PP-Module at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0920	Clarification for FMT_SMF.1/WLAN Table 3	Yes	
TD0837	Updates to WLAN Client PP-Module allow-lists	Yes	
TD0797	Addition of FCS_WPA_EXT to ECD	No	This TD only affects the ECD of the PP module
TD0710	WPA version restrictions	Yes	
TD0703	Removal of FIA_X509_EXT.2/WLAN evaluation activities for revocation checking	Yes	
TD0667	Move Set Wireless Freq Band to Optional/ Objective	Yes	

Table 7: NIAP TDs for WLANC

2.7. Functional Package for Transport Layer Security (TLS) [TLSPKG]

Table 8 contains the NIAP Technical Decisions (TDs) for this functional package at the time of the evaluation and a statement of applicability to the evaluation.

NIAP TD	TD Description	Applicable?	Non-Applicability Rationale
TD0779	Updated Session Resumption Support in TLS package V1.1	No	The TOE does not contain TLS server functionality.
TD0770	TLSS.2 connection with no client cert	No	The TOE does not contain TLS server functionality.
TD0739	PKG_TLS_V1.1 has 2 different publication dates	Yes	
TD0726	Corrections to (D)TLSS SFRs in TLS 1.1 FP	No	The TOE does not contain (D)TLS server functionality.
TD0513	CA Certificate loading	Yes	
TD0499	Testing with pinned certificates	Yes	
TD0469	Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1	No	The TOE does not contain TLS server functionality.
TD0442	Updated TLS Ciphersuites for TLS Package	Yes	

Table 8: NIAP TDs for TLSPKG

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

3. Security Problem Definition

The threats, assumptions, and organizational security policies (OSPs) are defined in the documents specified in Section 2 "CC Conformance Claim". This security target includes by reference the Security Problem Definition (composed of organizational policies, threat statements, and assumptions) from the following PP/PP-Module:

- Agent MDM Agents PP-Module
- BIO Biometric Enrolment and Verification cPP-Module
- BT Bluetooth PP-Module
- MDF Mobile Device Fundamentals PP
- VPNC VPN Client PP-Module
- WLANC WLAN Client PP-Module

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

4. Security Objectives

The security objectives are defined in the documents specified in Section 2 "CC Conformance Claim". This security target includes by reference the Security Objectives (Objectives for the TOE and Objectives for the Operational Environment) from the following PP/PP-Module:

- Agent MDM Agents PP-Module
- BIO Biometric Enrolment and Verification cPP-Module
- BT Bluetooth PP-Module
- MDF Mobile Device Fundamentals PP
- VPNC VPN Client PP-Module
- WLANC WLAN Client PP-Module

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

5. Extended Components Definition

The extended components definitions are defined in the documents specified in Section 2 "CC Conformance Claim".

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6. Security Requirements

6.1. TOE Security Functional Requirements

The table below summarizes the SFRs for the TOE and the operations performed on the components according to CC part 1. Operations in the SFRs use the following convention:

- Iterations (Iter.) are identified by appending a suffix to the original SFR.
- Refinements (Ref.) added to the text are shown in italic text, deletions are shown as strikethrough text.
- Assignments (Ass.) are shown in bold text.
- Selections (Sel.) are shown in bold text.

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
FAU – Security Audit	FAU_ALT_EXT.2 Agent Alerts		Agent
	FAU_GEN.1 Audit Data Generation		MDF
	FAU_GEN.1(2) Audit Data Generation		Agent
	FAU_GEN.1/BT Audit Data Generation (Bluetooth)		BT
	FAU_GEN.1/VPN Audit Data Generation		VPNC
	FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN)		WLANC
	FAU_SAR.1 Audit Review		MDF
	FAU_SEL.1(2) Security Audit Event Selection		Agent
	FAU_STG.1 Audit Storage Protection		MDF
	FAU_STG.4 Prevention of Audit Data Loss		MDF
FCS – Cryptographic Support	FCS_CKM.1 Cryptographic Key Generation		MDF
	FCS_CKM.1/VPN VPN Cryptographic Key Generation (IKE)		VPNC
	FCS_CKM.1/WPA Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections)		WLANC
	FCS_CKM.2/UNLOCKED Cryptographic Key Establishment		MDF
	FCS_CKM.2/LOCKED Cryptographic Key Establishment		MDF
	FCS_CKM.2/WLAN Cryptographic Key Distribution (Group Temporal Key for WLAN)		WLANC
	FCS_CKM_EXT.1 Cryptographic Key Support		MDF
	FCS_CKM_EXT.2 Cryptographic Key Random Generation		MDF

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
	FCS_CKM_EXT.3 Cryptographic Key Generation		MDF
	FCS_CKM_EXT.4 Key Destruction		MDF
	FCS_CKM_EXT.5 TSF Wipe		MDF
	FCS_CKM_EXT.6 Salt Generation		MDF
	FCS_CKM_EXT.7 Cryptographic Key Support (REK)		MDF
	FCS_CKM_EXT.8 Bluetooth Key Generation		BT
	FCS_COP.1/ENCRYPT Cryptographic Operation		MDF
	FCS_COP.1/HASH Cryptographic Operation		MDF
	FCS_COP.1/SIGN Cryptographic Operation		MDF
	FCS_COP.1/KEYHMAC Cryptographic Operation		MDF
	FCS_COP.1/CONDITION Cryptographic Operation		MDF
	FCS_HTTPS_EXT.1 HTTPS Protocol		MDF
	FCS_IPSEC_EXT.1 IPsec		VPNC
	FCS_IV_EXT.1 Initialization Vector Generation		MDF
	FCS_RBG_EXT.1/HW Random Bit Generation (Hardware)	FCS_RBG_EXT.1	MDF
	FCS_RBG_EXT.1/SW Random Bit Generation (Software)	FCS_RBG_EXT.1	MDF
	FCS_SRV_EXT.1 Cryptographic Algorithm Services		MDF
	FCS_STG_EXT.1 Cryptographic Key Storage		MDF
	FCS_STG_EXT.2 Encrypted Cryptographic Key Storage		MDF
	FCS_STG_EXT.3 Integrity of Encrypted Key Storage		MDF
	FCS_STG_EXT.4 Cryptographic Key Storage		Agent
	FCS_TLS_EXT.1 TLS Protocol		TLSPKG
	FCS_TLSC_EXT.1 TLS Client Protocol		TLSPKG
	FCS_TLSC_EXT.1/WLAN TLS Client Protocol (EAP-TLS for WLAN)		WLANC

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
	FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication		TLSPKG
	FCS_TLSC_EXT.2/WLAN TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN)		WLANC
	FCS_TLSC_EXT.4 TLS Client Support for Renegotiation		TLSPKG
	FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension		TLSPKG
	FCS_WPA_EXT.1 Supported WPA Versions		WLANC
FDP – User Data Protection	FDP_ACF_EXT.1 Access Control for System Services		MDF
	FDP_ACF_EXT.2 Access Control for System Resources		MDF
	FDP_DAR_EXT.1 Protected Data Encryption		MDF
	FDP_DAR_EXT.2 Sensitive Data Encryption		MDF
	FDP_IFC_EXT.1 Subset Information Flow Control		MDF
	FDP_RIP.2 Full Residual Information Protection		VPNC
	FDP_STG_EXT.1 User Data Storage		MDF
	FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)		MDF
	FDP_UPC_EXT.1/BLUETOOTH Inter-TSF User Data Transfer Protection (Bluetooth)		MDF
	FDP_VPN_EXT.1 Split Tunnel Prevention		VPNC
FIA – Identification and Authentication	FIA_AFL_EXT.1 Authentication Failure Handling		MDF
	FIA_BLT_EXT.1 Bluetooth User Authorization		BT
	FIA_BLT_EXT.2 Bluetooth Mutual Authentication		BT
	FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections		BT
	FIA_BLT_EXT.4 Secure Simple Pairing		BT
	FIA_BLT_EXT.6 Trusted Bluetooth Device User Authorization		BT
	FIA_BLT_EXT.7 Untrusted Bluetooth Device User Authorization		BT
	FIA_ENR_EXT.2 Agent Enrollment of Mobile Device into Management		Agent
	FIA_MBE_EXT.1 Biometric Enrolment		BIO

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
	FIA_MBE_EXT.2 Quality of Biometric Templates for Biometric Enrolment		BIO
	FIA_MBV_EXT.1 Biometric Verification		BIO
	FIA_MBV_EXT.2 Quality of Biometric Samples for Biometric Verification		BIO
	FIA_PAE_EXT.1 Port Access Entity Authentication		WLANC
	FIA_PMG_EXT.1 Password Management		MDF
	FIA_PSK_EXT.1 Pre-Shared Key Composition		VPNC
	FIA_PSK_EXT.2 Generated Pre-Shared Keys		VPNC
	FIA_TRT_EXT.1 Authentication Throttling		MDF
	FIA_UAU.5 Multiple Authentication Mechanisms		MDF
	FIA_UAU.6/CREDENTIAL Re-Authenticating (Credential Change)		MDF
	FIA_UAU.6/LOCKED Re-Authenticating (TSF Lock)		MDF
	FIA_UAU.7 Protected Authentication Feedback		MDF
	FIA_UAU_EXT.1 Authentication for Cryptographic Operation		MDF
	FIA_UAU_EXT.2 Timing of Authentication		MDF
	FIA_X509_EXT.1 X.509 Validation of Certificates		MDF
	FIA_X509_EXT.1/WLAN X.509 Certificate Validation		WLANC
	FIA_X509_EXT.2 X.509 Certificate Authentication		MDF
	FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for WLAN)		WLANC
	FIA_X509_EXT.3 Request Validation of Certificates		MDF
	FIA_X509_EXT.6 Certificate Storage and Management		WLANC
FMT – Security Management	FMT_MOF_EXT.1 Management of Security Functions Behavior		MDF
	FMT_POL_EXT.2 Agent Trusted Policy Update		Agent
	FMT_SMF.1 Specification of Management Functions		MDF
	FMT_SMF.1/VPN Specification of Management Functions (VPN)		VPNC

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
	FMT_SMF.1/WLAN Specification of Management Functions (WLAN Client)		WLANC
	FMT_SMF_EXT.1/BT Specification of Management Functions		BT
	FMT_SMF_EXT.2 Specification of Remediation Actions		MDF
	FMT_SMF_EXT.4 Specification of Management Functions		Agent
	FMT_UNR_EXT.1 User Unenrollment Prevention		Agent
FPT – Protection of the TSF	FPT_AEX_EXT.1 Application Address Space Layout Randomization		MDF
	FPT_AEX_EXT.2 Memory Page Permissions		MDF
	FPT_AEX_EXT.3 Stack Overflow Protection		MDF
	FPT_AEX_EXT.4 Domain Isolation		MDF
	FPT_BDP_EXT.1 Biometric Data Processing		BIO
	FPT_JTA_EXT.1 JTAG Disablement		MDF
	FPT_KST_EXT.1 Key Storage		MDF
	FPT_KST_EXT.2 No Key Transmission		MDF
	FPT_KST_EXT.3 No Plaintext Key Export		MDF
	FPT_NOT_EXT.1 Self-Test Notification		MDF
	FPT_PBT_EXT.1 Protection of Biometric Template		BIO
	FPT_STM.1 Reliable Time Stamps		MDF
	FPT_TST_EXT.1 TSF Cryptographic Functionality Testing		MDF
	FPT_TST_EXT.1/VPN TSF Self-Test		VPNC
	FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)		MDF
	FPT_TST_EXT.2/POSTKERNEL TSF Integrity Checking (Post-Kernel)		MDF
	FPT_TST_EXT.3 TSF Integrity Testing		MDF
	FPT_TST_EXT.3/WLAN TSF Cryptographic Functionality Testing (WLAN Client)		WLANC
	FPT_TUD_EXT.1 TSF Version Query		MDF

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Functional Class	Security Functional Requirement	Base Security Functional Component	Source
	FPT_TUD_EXT.2 TSF Update Verification		MDF
	FPT_TUD_EXT.3 Application Signing		MDF
	FPT_TUD_EXT.4 Trusted Update Verification		MDF
	FPT_TUD_EXT.5 Application Verification		MDF
	FPT_TUD_EXT.6 Trusted Update Verification		MDF
FTA – TOE Access	FTA_SSL_EXT.1 TSF- and User-initiated Locked State		MDF
	FTA_TAB.1 Default TOE Access Banners		MDF
	FTA_WSE_EXT.1 Wireless Network Access		WLANC
FTP – Trusted Path/Channels	FTP_BLT_EXT.1 Bluetooth Encryption		BT
	FTP_BLT_EXT.2 Persistence of Bluetooth Encryption		BT
	FTP_BLT_EXT.3/BR Bluetooth Encryption Parameters (BR/EDR)		BT
	FTP_BLT_EXT.3/LE Bluetooth Encryption Parameters (LE)		BT
	FTP_ITC.1/WLAN Trusted Channel Communication (Wire-less LAN)		WLANC
	FTP_ITC_EXT.1 Trusted Channel Communication		MDF
	FTP_ITC_EXT.1(2) Trusted Channel Communication		Agent
	FTP_TRP.1(2) Trusted Path (for Enrollment)		Agent

Table 9: SFRs for the TOE

6.1.1. Security Audit (FAU)

6.1.1.1. FAU_ALT_EXT.2 Agent Alerts

PP Origin: Agent

FAU_ALT_EXT.2.1

The MDM Agent shall provide an alert via the trusted channel to the MDM Server in the event of any of the following audit events:

- successful application of policies to a mobile device,
- **receiving** periodic reachability events,
- **no other events**.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FAU_ALT_EXT.2.2

The MDM Agent shall queue alerts if the trusted channel is not available.

6.1.1.2. FAU_GEN.1 Audit Data Generation

PP Origin: MDF
Applied TDs: TD0724

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

1. Start-up and shutdown of the audit functions
2. All auditable events for the not selected level of audit
3. All administrative actions
4. Start-up and shutdown of the OS
5. Insertion or removal of removable media
6. Specifically defined auditable events in ~~Table 2~~ *Table* 10
7. **no additional auditable events**

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	N/A
FAU_SAR.1	No events specified	N/A
FAU_STG.1	No events specified	N/A
FAU_STG.4	No events specified	N/A
FCS_CKM.1	None	No additional information
FCS_CKM.2/UNLOCKED	No events specified	N/A
FCS_CKM.2/LOCKED	No events specified	N/A
FCS_CKM_EXT.1	None	No additional information
FCS_CKM_EXT.2	No events specified	N/A
FCS_CKM_EXT.3	No events specified	N/A
FCS_CKM_EXT.4	No events specified	N/A
FCS_CKM_EXT.5	None	No additional information
FCS_CKM_EXT.6	No events specified	N/A
FCS_COP.1/ENCRYPT	No events specified	N/A
FCS_COP.1/HASH	No events specified	N/A
FCS_COP.1/SIGN	No events specified	N/A
FCS_COP.1/KEYHMAC	No events specified	N/A
FCS_COP.1/CONDITION	No events specified	N/A

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Requirement	Auditable Events	Additional Audit Record Contents
FCS_IV_EXT.1	No events specified	N/A
FCS_SRV_EXT.1	No events specified	N/A
FCS_STG_EXT.1	Import or destruction of key	Identity of key, role and identity of requester
	None	Identity of key, role and identity of requester
FCS_STG_EXT.2	No events specified	N/A
FCS_STG_EXT.3	Failure to verify integrity of stored key	Identity of key being verified
FDP_ACF_EXT.1	No events specified	N/A
FDP_DAR_EXT.1	Failure to encrypt/decrypt data	No additional information
FDP_DAR_EXT.2	Failure to encrypt/decrypt data	No additional information
FDP_IFC_EXT.1	No events specified	N/A
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database	Subject name of certificate
FIA_PMG_EXT.1	No events specified	N/A
FIA_TRT_EXT.1	No events specified	N/A
FIA_UAU.5	No events specified	N/A
FIA_UAU.7	No events specified	N/A
FIA_UAU_EXT.1	No events specified	N/A
FIA_X509_EXT.1	Failure to validate X.509v3 certificate	Reason for failure of validation
FIA_X509_EXT.2	No events specified	N/A
FMT_MOF_EXT.1	No events specified	N/A
FPT_AEX_EXT.1	No events specified	N/A
FPT_AEX_EXT.2	No events specified	N/A
FPT_AEX_EXT.3	No events specified	N/A
FPT_JTA_EXT.1	No events specified	N/A
FPT_KST_EXT.1	No events specified	N/A
FPT_KST_EXT.2	No events specified	N/A
FPT_KST_EXT.3	No events specified	N/A
FPT_NOT_EXT.1	None	No additional information
FPT_STM.1	No events specified	N/A
FPT_TST_EXT.1	Initiation of self-test	No additional information
	Failure of self-test	No additional information
FPT_TST_EXT.2/ PRE-KERNEL	Start-up of TOE	No additional information
	None	No additional information
FPT_TUD_EXT.1	No events specified	N/A
FTA_SSL_EXT.1	No events specified	N/A
FTA_TAB.1	No events specified	N/A

Table 10: Mandatory Auditable Events (MDF)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:

1. Date and time of the event
2. Type of event
3. Subject identity
4. The outcome (success or failure) of the event
5. Additional information in ~~Table 2~~ *Table 10*
6. **no additional information**

6.1.1.3. FAU_GEN.1(2) Audit Data Generation

PP Origin: Agent
Applied TDs: TD0660

FAU_GEN.1.1(2)

The MDM Agent shall **implement functionality** to generate an MDM Agent audit record of the following auditable events:

a) Startup and shutdown of the MDM Agent;

b) All auditable events for not specified level of audit; and

c) MDM policy updated, any modification commanded by the MDM Server, specifically defined auditable events listed in ~~Table 1~~ *Table 11*, and **no other events**.

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ALT_EXT.2	Success/failure of sending alert.	No additional information.
FAU_GEN.1(2)	None.	N/A
FAU_SEL.1(2)	All modifications to the audit configuration that occur while the audit collection functions are operating.	No additional information.
FCS_STG_EXT.4, ~~FCS_STG_EXT.1(2)~~	None.	
FCS_TLSC_EXT.1	Failure to establish a TLS session.	Reason for failure.
	Failure to verify presented identifier.	Presented identifier and reference identifier.
	Establishment/termination of a TLS session.	Non-TOE endpoint of connection.
FIA_ENR_EXT.2	Enrollment in management.	Reference identifier of MDM Server.
FMT_POL_EXT.2	Failure of policy validation.	Reason for failure of validation.
FMT_SMF_EXT.4	Outcome (Success/failure) of function.	No additional information.
FMT_UNR_EXT.1.1	**Attempt to unenroll**	No additional information.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Requirement	Auditable Events	Additional Audit Record Contents
FTP_ITC_EXT.1(2)	Initiation and termination of trusted channel.	Trusted channel protocol. Non-TOE endpoint of connection.

Table 11: Auditable Events (Agent)

FAU_GEN.1.2(2)

The **TSF** shall record within each MDM Agent audit record at least the following information:

a) Date and time of the event, type of event, subject identity, (if relevant) the outcome (success or failure) of the event, and additional information in ~~Table 1~~ *Table 11*; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP-Module/ST, **no other audit relevant information**.

6.1.1.4. FAU_GEN.1/BT Audit Data Generation

PP Origin: BT

Applied TDs: TD0645, TD0707

FAU_GEN.1.1/BT

The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions

b) All auditable events for the not specified level of audit

c) Specifically defined auditable events in the Auditable Events table (*Table 12*).

Requirement	Auditable Events	Additional Audit Record Contents
FCS_CKM_EXT.8	None.	
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user rejected connection, incorrect pin entry).
	Failed user authorization for local Bluetooth Service.	**Complete** BD_ADDR and **name of device**. Bluetooth profile. Identity of local service with **service ID**.
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	**Complete** BD_ADDR and **name of device**.
	Failure of Bluetooth connection.	Reason for failure.
FIA_BLT_EXT.3 ~~(optional)~~	~~Duplicate connection attempt.~~	~~[selection: complete, last [assignment: integer greater than or equal to 2] octets of the] BD_ADDR of connection attempt.~~
FIA_BLT_EXT.4	None.	
~~FIA_BLT_EXT.5 (if claimed)~~	None.	
FIA_BLT_EXT.6	None.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Requirement	Auditable Events	Additional Audit Record Contents
FIA_BLT_EXT.7	None.	
FTP_BLT_EXT.1	None.	
FTP_BLT_EXT.2	None.	
FTP_BLT_EXT.3/BR	None.	
FTP_BLT_EXT.3/LE (if claimed)	None.	

Table 12: Auditable Events (BT)

Application Note: *FIA_BLT_EXT.3 is excluded from Table 12 because the rejection is performed at the HCI layer. FIA_BLT_EXT.5 is not claimed.*

FAU_GEN.1.2/BT

The TSF shall record within each audit record at least the following information:

a) Date and time of the event

b) Type of event

c) Subject identity

d) The outcome (success or failure) of the event

e) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, Additional information in the Auditable Events table (*Table 12*).

6.1.1.5. FAU_GEN.1/VPN Audit Data Generation

PP Origin: VPNC

FAU_GEN.1.1/VPN

The TSF and **TOE platform** shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the not specified level of audit;

c) All administrative actions;

d) Specifically defined auditable events listed in the Auditable Events tables (*Table 13*).

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1/VPN	No events specified.	N/A
FCS_CKM.1/VPN	No events specified.	N/A
FCS_IPSEC_EXT.1	Decisions to DISCARD or BYPASS network packets processed by the TOE.	Presumed identity of source subject. The entry in the SPD that applied to the decision.
	Failure to establish an IPsec SA.	Identity of destination subject. Reason for failure.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Requirement	Auditable Events	Additional Audit Record Contents
	Establishment/Termination of an IPsec SA.	Identity of destination subject. Transport layer protocol, if applicable. Source subject service identifier, if applicable. Non-TOE endpoint of connection (IP address) for both successes and failures.
FDP_RIP.2	No events specified.	N/A
FDP_VPN_EXT.1	No events specified.	N/A
FIA_PSK_EXT.1	No events specified.	N/A
FIA_PSK_EXT.2	No events specified.	N/A
FMT_SMF.1/VPN	Success or failure of management function.	No additional information.
FPT_TST_EXT.1/VPN	No events specified.	N/A

Table 13: Auditable Events (VPN)

FAU_GEN.1.2/VPN

The TSF and **TOE platform** shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP-Module/ST, information specified in column three of Auditable Events table.

6.1.1.6. FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN)

PP Origin: WLANC

FAU_GEN.1.1/WLAN

The TSF shall **implement functionality** to generate an audit record of the following auditable events:

a) Startup and shutdown of the audit functions;

b) All auditable events for not specified level of audit; and

c) All auditable events for mandatory SFRs specified in ~~Table 2 and selected SFRs in Table 5~~ *Table 14*.

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1/WLAN	No events specified.	N/A
FCS_CKM.1/WPA	No events specified.	N/A
FCS_CKM.2/WLAN	No events specified.	N/A
FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure. Non-TOE endpoint of connection.
	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FCS_WPA_EXT.1	No events specified.	N/A

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Requirement	Auditable Events	Additional Audit Record Contents
FIA_PAE_EXT.1	No events specified.	N/A
FIA_X509_EXT.1/WLAN	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2/WLAN	No events specified.	N/A
FIA_X509_EXT.6	Attempts to load certificates.	None.
	Attempts to revoke certificates.	None.
FMT_SMF.1/WLAN	No events specified.	N/A
FPT_TST_EXT.3/WLAN	Execution of this set of TSF self-tests.	None.
	None.	**None.**
FTA_WSE_EXT.1	All attempts to connect to access points.	For each access point record the **Complete SSID and MAC** of the MAC Address. Success and failures (including reason for failure).
FTP_ITC.1/WLAN	All attempts to establish a trusted channel.	Identification of the non-TOE endpoint of the channel.

Table 14: Auditable Events (WLAN)

Requirement	Auditable Events	Additional Audit Record Contents
FCS_TLSC_EXT.2/WLAN	No events specified.	N/A

Table 15: Auditable Events for Selection-based Requirements (WLANC)

FAU_GEN.1.2/WLAN

The **TSF** shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, (if relevant) the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP-Module/ST, Additional Audit Record Contents as specified in ~~Table 2 and Table 5~~ *Table 14*.

6.1.1.7. FAU_SAR.1 Audit Review

PP Origin: MDF

FAU_SAR.1.1

The TSF shall provide the administrator with the capability to read all audited events and record contents from the audit records.

FAU_SAR.1.2

The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.1.8. FAU_SEL.1(2) Security Audit Event Selection

PP Origin: Agent

FAU_SEL.1.1(2)

The TSF shall **implement functionality** to select the set of events to be audited from the set of all auditable events based on the following attributes:

a) event type
b) success of auditable security events, failure of auditable security events, **no other attributes**.

6.1.1.9. FAU_STG.1 Audit Storage Protection

PP Origin: MDF

FAU_STG.1.1

The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2

The TSF shall be able to prevent unauthorized modifications to the stored audit records in the audit trail.

6.1.1.10. FAU_STG.4 Prevention of Audit Data Loss

PP Origin: MDF

FAU_STG.4.1

The TSF shall overwrite the oldest stored audit records if the audit trail is full.

6.1.2. Cryptographic Support (FCS)

6.1.2.1. FCS_CKM.1 Cryptographic Key Generation

PP Origin: MDF, VPNC
Applied TDs: TD0871, TD0877

FCS_CKM.1.1

The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm

- **ECC schemes using**
  - **"NIST curves" P-256, P-384, and P-521 that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Appendix B.4**
  - **Curve25519 schemes that meet the following: RFC 7748**
- **FFC schemes using**
  - **"safe-prime" groups that meet the following: 'NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"'**

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.2.2. FCS_CKM.1/VPN VPN Cryptographic Key Generation (IKE)

PP Origin: VPNC
Applied TDs: TD0877

FCS_CKM.1.1/VPN

The TSF shall **implement functionality to** generate **asymmetric** cryptographic keys **used for IKE peer authentication** in accordance with:

- **FIPS PUB 186-5, "Digital Signature Standard (DSS)," Appendix B.4 for ECDSA schemes and implementing "NIST curves," P-256, P-384 and P-521**

and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

6.1.2.3. FCS_CKM.1/WPA Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections)

PP Origin: WLANC

FCS_CKM.1.1/WPA

The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm PRF-384 and **PRF-704** (as defined in IEEE 802.11-2012) and specified key sizes 256 bits and **128 bits** using a Random Bit Generator as specified in FCS_RBG_EXT.1.

6.1.2.4. FCS_CKM.2/UNLOCKED Cryptographic Key Establishment

PP Origin: MDF, VPNC

FCS_CKM.2.1/UNLOCKED
The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method

- **RSA-based key establishment schemes that meet the following**
  - **RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1:RSA Cryptography Specifications Version 2.2"**
- **Elliptic curve-based key establishment schemes that meet the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**
- **Finite field-based key establishment schemes that meet the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

6.1.2.5. FCS_CKM.2/LOCKED Cryptographic Key Establishment

PP Origin: MDF

FCS_CKM.2.1/LOCKED

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- **Elliptic curve-based key establishment schemes that meet the following:**
  - **RFC 7748, "Elliptic Curves for Security"**

for the purposes of encrypting sensitive data received while the device is locked.

6.1.2.6. FCS_CKM.2/WLAN Cryptographic Key Distribution (Group Temporal Key for WLAN)

PP Origin: WLANC

FCS_CKM.2.1/WLAN

The TSF shall decrypt Group Temporal Key in accordance with a specified cryptographic key distribution method AES Key Wrap (as defined in RFC 3394) in an EAPOL-Key frame (as defined in IEEE 802.11-2012) for the packet format and timing considerations and does not expose the cryptographic keys.

6.1.2.7. FCS_CKM_EXT.1 Cryptographic Key Support

PP Origin: MDF

FCS_CKM_EXT.1.1

The TSF shall support **immutable hardware** REKs with a **symmetric** key of strength **256 bits**.

FCS_CKM_EXT.1.2

Each REK shall be hardware-isolated from the OS on the TSF in runtime.

FCS_CKM_EXT.1.3

Each REK shall be generated by an RBG in accordance with FCS_RBG_EXT.1.

6.1.2.8. FCS_CKM_EXT.2 Cryptographic Key Random Generation

PP Origin: MDF

FCS_CKM_EXT.2.1

All DEKs shall be **randomly generated** with entropy corresponding to the security strength of AES key sizes of **256** bits.

6.1.2.9. FCS_CKM_EXT.3 Cryptographic Key Generation

PP Origin: MDF

FCS_CKM_EXT.3.1

The TSF shall use **symmetric KEKs of 128-bit, 256-bit security strength corresponding to at least the security strength of the keys encrypted by the KEK**.

FCS_CKM_EXT.3.2

The TSF shall generate all KEKs using one of the following methods:

- Derive the KEK from a Password Authentication Factor according to FCS_COP.1.1/CONDITION and
- **Generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1)**
- **Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by**
  - **concatenating the keys and using a KDF (as described in SP 800-56C)**

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- **encrypting one key with another.**

6.1.2.10. FCS_CKM_EXT.4 Key Destruction

PP Origin: MDF

FCS_CKM_EXT.4.1

The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:

- By clearing the KEK encrypting the target key
- In accordance with the following rules
  - For volatile memory, the destruction shall be executed by a single direct overwrite **consisting of zeros**.
  - For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed by a read-verify.
  - For non-volatile flash memory, that is not wear-leveled, the destruction shall be executed **by a block erase that erases the reference to memory that stores data as well as the data itself**.
  - For non-volatile flash memory, that is wear-leveled, the destruction shall be executed **by a block erase**.
  - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.

FCS_CKM_EXT.4.2

The TSF shall destroy all plaintext keying material and critical security parameters when no longer needed.

6.1.2.11. FCS_CKM_EXT.5 TSF Wipe

PP Origin: MDF

FCS_CKM_EXT.5.1

The TSF shall wipe all protected data by

- **Cryptographically erasing the encrypted DEKs or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1**.

FCS_CKM_EXT.5.2

The TSF shall perform a power cycle on conclusion of the wipe procedure.

6.1.2.12. FCS_CKM_EXT.6 Salt Generation

PP Origin: MDF

FCS_CKM_EXT.6.1

The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1.

Application Note: *The salt is generated using FCS_RBG_EXT.1/HW.*

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.2.13. FCS_CKM_EXT.7 Cryptographic Key Support (REK)

PP Origin: MDF

FCS_CKM_EXT.7.1

A REK shall not be able to be read from or exported from the hardware.

6.1.2.14. FCS_CKM_EXT.8 Bluetooth Key Generation

PP Origin: BT

FCS_CKM_EXT.8.1

The TSF shall generate public/private ECDH key pairs every **new connection attempt**.

6.1.2.15. FCS_COP.1/ENCRYPT Cryptographic Operation

PP Origin: MDF, VPNC

FCS_COP.1.1/ENCRYPT

The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm:

- AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and
- **AES Key Wrap (KW) (as defined in NIST SP 800-38F)**
- **AES-GCM (as defined in NIST SP 800-38D)**
- **AES-CCM (as defined in NIST SP 800-38C)**
- **AES-XTS (as defined in NIST SP 800-38E) mode**
- **AES-CCMP-256 (as defined in NIST SP800-38C and IEEE 802.11ac-2013)**
- **AES-GCMP-256 (as defined in NIST SP800-38D and IEEE 802.11ac-2013)**

and cryptographic key sizes 128-bit key sizes and **256-bit key sizes**.

6.1.2.16. FCS_COP.1/HASH Cryptographic Operation

PP Origin: MDF

FCS_COP.1.1/HASH

The TSF shall perform cryptographic hashing in accordance with a specified cryptographic algorithm SHA-1 and **SHA-256, SHA-384, SHA-512** and message digest sizes 160 and **256, 384, 512 bits** that meet the following: FIPS Pub 180-4.

6.1.2.17. FCS_COP.1/SIGN Cryptographic Operation

PP Origin: MDF
Applied TDs: TD0871

FCS_COP.1.1/SIGN

The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm

- **RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Section 4**

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- **ECDSA schemes using "NIST curves" P-384 and P-256, P-521 that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Section 5**

6.1.2.18. FCS_COP.1/KEYHMAC Cryptographic Operation

PP Origin: MDF

FCS_COP.1.1/KEYHMAC

The TSF shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-SHA-1 and **HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512** and cryptographic key sizes **greater than or equal to 112 bits** and message digest sizes 160 and **256, 384, 512** bits that meet the following: FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code", and FIPS Pub 180-4, "Secure Hash Standard".

6.1.2.19. FCS_COP.1/CONDITION Cryptographic Operation

PP Origin: MDF

FCS_COP.1.1/CONDITION

The TSF shall perform conditioning in accordance with a specified cryptographic algorithm HMAC-**SHA-256** using a salt, and **PBKDF2 with one iterations, repetitive AES-CBC-256 encryption with a duration between 100 and 150 milli-seconds (50,000 repetition minimum)** and output cryptographic key sizes **256** that meet the following: **NIST SP 800-132**.

Application Note: *The number of repetitions is calibrated to take at least 100 to 150 milliseconds with a minimum of 50,000 repetitions. The number of repetitions may be greater in some devices.*

6.1.2.20. FCS_HTTPS.1 HTTPS Protocol

PP Origin: MDF

FCS_HTTPS_EXT.1.1

The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS as defined **in the Functional Package for Transport Layer Security (TLS), version 1.1**.

FCS_HTTPS_EXT.1.3

The TSF shall notify the application and **request application authorization to establish the connection** if the peer certificate is deemed invalid.

6.1.2.21. FCS_IPSEC_EXT.1 IPsec

PP Origin: VPNC
Applied TDs: TD0890

FCS_IPSEC_EXT.1.1

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2

The TSF shall implement **tunnel mode**.

FCS_IPSEC_EXT.1.3

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, **AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC**.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:

- **IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8247, and no other RFCs for hash functions.**

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the **IKEv2** protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and **AES-GCM-128 as specified in RFC 5282, AES-GCM-256 as specified in RFC 5282**.

FCS_IPSEC_EXT.1.7

The TSF shall ensure that **IKEv2 SA lifetimes can be configured by an Administrator based on length of time**. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH Groups

- 19 (256-bit Random ECP), 20 (384-bit Random ECP) according to RFC 5114 and
- **14 (2048-bit MODP), 15 (3072-bit MODP)), 16 (4096-bit MODP), 17 (6144-bit MODP), 18 (8192-bit MODP), and 21 (521-bit ECP) according to RFC 3526**.

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE DH key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least **224, 256, or 384** bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^**112, 128, or 192**.

FCS_IPSEC_EXT.1.11

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall ensure that **IKEv2 performs peer authentication using RSA, ECDSA that use X.509v3 certificates that conform to RFC 4945 and Pre-shared Keys that conform to RFC 8784**.

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the **Fully Qualified Domain Name (FQDN)** and **no other reference identifier type** contained in a certificate does not match the expected values for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.14

The **TSF** shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the **IKEv2 IKE_SA** connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the **IKEv2 CHILD_SA** connection.

6.1.2.22. FCS_IV_EXT.1 Initialization Vector Generation

PP Origin: MDF

FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with *MDF* Table 11: References and IV Requirements for NIST-approved Cipher Modes.

6.1.2.23. FCS_RBG_EXT.1/HW Random Bit Generation (Hardware)

PP Origin: MDF

FCS_RBG_EXT.1.1/HW

The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using **CTR_DRBG (AES)**.

FCS_RBG_EXT.1.2/HW

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from **a TSF-hardware-based noise source** with a minimum of **384 bits** of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1.3/HW

The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

Application Note: *FCS_RBG_EXT.1/HW is for the Secure Enclave RBG.*

6.1.2.24. FCS_RBG_EXT.1/SW Random Bit Generation (Software)

PP Origin: MDF

FCS_RBG_EXT.1.1/SW

The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using **CTR_DRBG (AES)**.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FCS_RBG_EXT.1.2/SW

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a **software-based noise source** with a minimum of **256 bits** of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1.3/SW

The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

Application Note: *FCS_RBG_EXT.1/SW is for the kernel and user space RBG.*

6.1.2.25. FCS_SRV_EXT.1 Cryptographic Algorithm Services

PP Origin: MDF

FCS_SRV_EXT.1.1

The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:

- All mandatory and **selected algorithms with the exception of ECC over Curve25519-based algorithms** in FCS_CKM.2/LOCKED
- The following algorithms in FCS_COP.1/ENCRYPT: AES-CBC, **no other modes**
- All selected algorithms in FCS_COP.1/SIGN
- All mandatory and selected algorithms in FCS_COP.1/HASH
- All mandatory and selected algorithms in FCS_COP.1/KEYHMAC
- **No other cryptographic operations**

6.1.2.26. FCS_STG_EXT.1 Cryptographic Key Storage

PP Origin: MDF

FCS_STG_EXT.1.1

The TSF shall provide **software-based** secure key storage for asymmetric private keys and **symmetric keys, persistent secrets**.

FCS_STG_EXT.1.2

The TSF shall be capable of importing keys or secrets into the secure key storage upon request of **the user, the administrator** and **applications running on the TSF**.

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys or secrets in the secure key storage upon request of **the user, the administrator**.

FCS_STG_EXT.1.4

The TSF shall have the capability to allow only the application that imported the key or secret the use of the key or secret. Exceptions may only be explicitly authorized by **a common application developer**.

FCS_STG_EXT.1.5

The TSF shall allow only the application that imported the key or secret to request that the key or secret be destroyed. Exceptions may only be explicitly authorized by **a common application developer**.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.2.27. FCS_STG_EXT.2 Encrypted Cryptographic Key Storage

PP Origin: MDF

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs, KEKs, **WPA2/WPA3 (PSKs), IPsec (client certificates), Bluetooth keys** and **all software-based key storage** by KEKs that are

- **Protected by the REK with**
  - **encryption by a KEK chaining from a REK**
- **Protected by the REK and the password with**
  - **encryption by a KEK chaining to a REK and the password-derived or biometric-unlocked KEK.**

FCS_STG_EXT.2.2

DEKs, KEKs, **WPA2/WPA3 (PSKs), IPsec (client certificates), Bluetooth keys** and **all software-based key storage** shall be encrypted using one of the following methods:

- **using AES in the Key Wrap (KW) mode.**

6.1.2.28. FCS_STG_EXT.3 Integrity of Encrypted Key Storage

PP Origin: MDF

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted DEKs and KEKs and **long-term trusted channel key material** by

- **an immediate application of the key for decrypting the protected data followed by a successful verification of the decrypted data with previously known information.**

FCS_STG_EXT.3.2

The TSF shall verify the integrity of the **MAC** of the stored key prior to use of the key.

6.1.2.29. FCS_STG_EXT.4 Cryptographic Key Storage

PP Origin: Agent

FCS_STG_EXT.4.1

The MDM Agent shall use the platform provided key storage for all persistent secret and private keys.

6.1.2.30. FCS_TLS_EXT.1 TLS Protocol

PP Origin: TLSPKG

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FCS_TLS_EXT.1.1

The product shall implement **TLS as a client**.

6.1.2.31. FCS_TLSC_EXT.1 TLS Client Protocol

PP Origin: TLSPKG
Applied TDs: TD0442

FCS_TLSC_EXT.1.1

The product shall implement TLS 1.2 (RFC 5246) and **no earlier TLS versions** as a client that supports the cipher suites
- **TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246**
- **TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246**
- **TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288**
- **TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288**
- **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289**
- **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289**
- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**

and also supports functionality for
- **mutual authentication**
- **session renegotiation.**

FCS_TLSC_EXT.1.2

The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3

The product shall not establish a trusted channel if the server certificate is invalid
- **with no exceptions.**

6.1.2.32. FCS_TLSC_EXT.1/WLAN TLS Client Protocol (EAP-TLS for WLAN)

PP Origin: WLANC

FCS_TLSC_EXT.1.1/WLAN

The TSF shall implement TLS 1.2 (RFC 5246) and **TLS 1.1 (RFC 4346)** in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following cipher suites:
- **TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246**
- **TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288**
- **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289**
- **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289**
- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**

FCS_TLSC_EXT.1.2/WLAN

The TSF shall generate random values used in the EAP-TLS exchange using the RBG specified in FCS_RBG_EXT.1.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FCS_TLSC_EXT.1.3/WLAN

The TSF shall use X509 v3 certificates as specified in FIA_X509_EXT.1/WLAN.

FCS_TLSC_EXT.1.4/WLAN

The TSF shall verify that the server certificate presented includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FCS_TLSC_EXT.1.5/WLAN

The TSF shall allow an authorized administrator to configure the list of CAs that are allowed to sign authentication server certificates that are accepted by the TOE.

6.1.2.33. FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication

PP Origin: TLSPKG

FCS_TLSC_EXT.2.1

The product shall support mutual authentication using X.509v3 certificates.

6.1.2.34. FCS_TLSC_EXT.2/WLAN TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN)

PP Origin: WLANC

FCS_TLSC_EXT.2.1/WLAN

The TSF shall present the Supported Elliptic Curves extension in the Client Hello with the following NIST curves: **secp256r1, secp384r1, secp521r1**.

6.1.2.35. FCS_TLSC_EXT.4 TLS Client Support for Renegotiation

PP Origin: TLSPKG

FCS_TLSC_EXT.4.1

The product shall support secure renegotiation through use of the "renegotiation_info" TLS extension in accordance with RFC 5746.

6.1.2.36. FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension

PP Origin: TLSPKG

FCS_TLSC_EXT.5.1

The product shall present the Supported Groups Extension in the Client Hello with the supported groups

- **secp256r1**
- **secp384r1**

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- **secp521r1**

6.1.2.37. FCS_WPA_EXT.1 Supported

PP Origin: WLANC

FCS_WPA_EXT.1.1

The TSF shall support WPA3 and **WPA2** security type.

6.1.3. User Data Protection (FDP)

6.1.3.1. FDP_ACF_EXT.1 Access Control for System Services

PP Origin: MDF

FDP_ACF_EXT.1.1

The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

FDP_ACF_EXT.1.2

The TSF shall provide an access control policy that prevents **application, groups of applications** from accessing **private** data stored by other **application, groups of applications**. Exceptions may only be explicitly authorized for such sharing by **a common application developer**.

6.1.3.2. FDP_ACF_EXT.2 Access Control for System Resources

PP Origin: MDF

FDP_ACF_EXT.2.1

The TSF shall provide a separate **keystore, account credential database** for each application group and only allow applications within that process group to access the resource. Exceptions may only be explicitly authorized for such sharing by **no one**.

6.1.3.3. FDP_DAR_EXT.1 Protected Data Encryption

PP Origin: MDF

FDP_DAR_EXT.1.1

Encryption shall cover all protected data.

FDP_DAR_EXT.1.2

Encryption shall be performed using DEKs with AES in the **XTS** mode with key size **128**, **256** bits.

6.1.3.4. FDP_DAR_EXT.2 Sensitive Data Encryption

PP Origin: MDF

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FDP_DAR_EXT.2.1

The TSF shall provide a mechanism for applications to mark data and keys as sensitive.

FDP_DAR_EXT.2.2

The TSF shall use an asymmetric key scheme to encrypt and store sensitive data received while the product is locked.

FDP_DAR_EXT.2.3

The TSF shall encrypt any stored symmetric key and any stored private key of the asymmetric keys used for the protection of sensitive data according to FCS_STG_EXT.2.1 selection 2.

FDP_DAR_EXT.2.4

The TSF shall decrypt the sensitive data that was received while in the locked state upon transitioning to the unlocked state using the asymmetric key scheme and shall re-encrypt that sensitive data using the symmetric key scheme.

6.1.3.5. FDP_IFC_EXT.1 Subset Information Flow Control

PP Origin: MDF, VPNC
FDP_IFC_EXT.1.1

The TSF shall

- **provide an interface which allows a VPN client to protect all IP traffic using IPsec**
- **provide a VPN client which can protect all IP traffic using IPsec as defined in the PP-Module for Virtual Private Network (VPN) Clients, version 2.4**

with the exception of IP traffic needed to manage the VPN connection, and **AirPrint, cellular services, voicemail, and initial Captive Network communication**, when the VPN is enabled.

6.1.3.6. FDP_RIP.2 Full Residual Information Protection

PP Origin: VPNC

FDP_RIP.2.1

The **TOE** shall enforce that any previous information content of a resource is made unavailable upon the **allocation of the resource to** all objects.

6.1.3.7. FDP_STG_EXT.1 User Data Storage

PP Origin: MDF

FDP_STG_EXT.1.1

The TSF shall provide protected storage for the Trust Anchor Database.

6.1.3.8. FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)

PP Origin: MDF

FDP_UPC_EXT.1.1/APPS

The TSF shall provide a means for non-TSF applications executing on the TOE to use

- Mutually authenticated TLS as defined in the Functional Package for Transport Layer Security (TLS), version 1.1,
- HTTPS,

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

and

- **No other protocol**

to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FDP_UPC_EXT.1.2/APPS

The TSF shall permit the non-TSF applications to initiate communication via the trusted channel.

6.1.3.9. FDP_UPC_EXT.1/BLUETOOTH Inter-TSF User Data Transfer Protection (Bluetooth)

PP Origin: MDF

FDP_UPC_EXT.1.1/BLUETOOTH

The TSF shall provide a means for non-TSF applications executing on the TOE to use

- Bluetooth BR/EDR in accordance with the PP-Module for Bluetooth, version 1.0,

and

- **Bluetooth LE in accordance with the PP-Module for Bluetooth, version 1.0**

to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FDP_UPC_EXT.1.2/BLUETOOTH

The TSF shall permit the non-TSF applications to initiate communication via the trusted channel.

6.1.3.10. FDP_VPN_EXT.1 Split Tunnel Prevention

PP Origin: VPNC

FDP_VPN_EXT.1.1

The TSF shall ensure that all IP traffic (other than IP traffic required to establish the VPN connection) flow through the IPsec VPN client.

6.1.4. Identification and Authentication (FIA)

6.1.4.1. FIA_AFL_EXT.1 Authentication Failure Handling

PP Origin: MDF

FIA_AFL_EXT.1.1

The TSF shall consider password and **no other mechanism** as critical authentication mechanisms.

FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive integer within **2 to 11** of **unique** unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

FIA_AFL_EXT.1.3

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall maintain the number of unsuccessful authentication attempts that have occurred upon power off.

FIA_AFL_EXT.1.4

When the defined number of unsuccessful authentication attempts has exceeded the maximum allowed for a given authentication mechanism, all future authentication attempts will be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

FIA_AFL_EXT.1.5

When the defined number of unsuccessful authentication attempts for the last available authentication mechanism or single critical authentication mechanism has been surpassed, the TSF shall perform a wipe of all protected data.

FIA_AFL_EXT.1.6

The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

6.1.4.2. FIA_BLT_EXT.1 Bluetooth User Authorization

PP Origin: BT

FIA_BLT_EXT.1.1

The TSF shall require explicit user authorization before pairing with a remote Bluetooth device.

6.1.4.3. FIA_BLT_EXT.2 Bluetooth Mutual Authentication

PP Origin: BT

FIA_BLT_EXT.2.1

The TSF shall require Bluetooth mutual authentication between devices prior to any data transfer over the Bluetooth link.

6.1.4.4. FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections

PP Origin: BT

FIA_BLT_EXT.3.1

The TSF shall discard pairing and session initialization attempts from a Bluetooth device address (BD_ADDR) to which an active session already exists.

6.1.4.5. FIA_BLT_EXT.4 Secure Simple Pairing

PP Origin: BT

FIA_BLT_EXT.4.1

The TOE shall support Bluetooth Secure Simple Pairing, both in the host and the controller.

FIA_BLT_EXT.4.2

The TOE shall support Secure Simple Pairing during the pairing process.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.4.6. FIA_BLT_EXT.6 Trusted Bluetooth Device User Authorization

PP Origin: BT

FIA_BLT_EXT.6.1

The TSF shall require explicit user authorization before granting trusted remote devices access to services associated with the following Bluetooth profiles: **none**.

6.1.4.7. FIA_BLT_EXT.7 Untrusted Bluetooth Device User Authorization

PP Origin: BT

FIA_BLT_EXT.7.1

The TSF shall require explicit user authorization before granting untrusted remote devices access to services associated with the following Bluetooth profiles: **all**.

6.1.4.8. FIA_ENR_EXT.2 Agent Enrollment of Mobile Device into Management

PP Origin: Agent

FIA_ENR_EXT.2.1

The MDM Agent shall record the reference identifier of the MDM Server during the enrollment process.

6.1.4.9. FIA_MBE_EXT.1 Biometric Enrolment

PP Origin: BIO
Applied TDs: TD0714

FIA_MBE_EXT.1.1

The TSF shall provide a mechanism to enrol an authenticated user to the biometric system.

6.1.4.10. FIA_MBE_EXT.2 Quality of Biometric Templates for Biometric Enrolment

PP Origin: BIO

FIA_MBE_EXT.2.1

The TSF shall only use biometric samples of sufficient quality for enrolment. Sufficiency of sample data shall be determined by measuring sample with **a developer defined quality assessment method**.

6.1.4.11. FIA_MBV_EXT.1 Biometric Verification

PP Origin: BIO

FIA_MBV_EXT.1.1

The TSF shall provide a biometric verification mechanism using **face, fingerprint**.

FIA_MBV_EXT.1.2

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall provide a biometric verification mechanism with the **FAR** not exceeding **0.01%** for the upper bound of 95% confidence interval and, **FRR** not exceeding **5%** for the upper bound of **95%** confidence interval.

6.1.4.12. FIA_MBV_EXT.2 Quality of Biometric Samples for Biometric Verification

PP Origin: BIO

FIA_MBV_EXT.2.1

The TSF shall only use biometric samples of sufficient quality for verification. Sufficiency of sample data shall be determined by measuring sample with **a developer defined quality assessment method**.

6.1.4.13. FIA_PAE_EXT.1 Port Access Entity Authentication

PP Origin: WLANC

FIA_PAE_EXT.1.1

The TSF shall conform to IEEE Standard 802.1X for a Port Access Entity (PAE) in the "Supplicant" role.

6.1.4.14. FIA_PMG_EXT.1 Password Management

PP Origin: MDF

FIA_PMG_EXT.1.1

The TSF shall support the following for the Password Authentication Factor:

1. Passwords shall be able to be composed of any combination of **upper and lower case letters**, numbers, and special characters: **"!", "@", "#", "$", "%", "^", "&", "*", "(", ")"**;
2. Password length up to **16** characters shall be supported.

6.1.4.15. FIA_PSK_EXT.1 Pre-Shared Key Composition

PP Origin: VPNC

FIA_PSK_EXT.1.1

The TSF shall be able to use pre-shared keys for **IKEv2**.

FIA_PSK_EXT.1.2

The TSF shall be able to accept the following as pre-shared keys: **generated bit-based** keys.

6.1.4.16. FIA_PSK_EXT.2 Generated Pre-Shared Keys

PP Origin: VPNC

FIA_PSK_EXT.2.1

The TSF shall be able to **accept externally generated pre-shared keys**.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.4.17. FIA_TRT_EXT.1 Authentication Throttling

PP Origin: MDF

FIA_TRT_EXT.1.1

The TSF shall limit automated user authentication attempts by **enforcing a delay between incorrect authentication attempts** for all authentication mechanisms selected in FIA_UAU.5.1. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

6.1.4.18. FIA_UAU.5 Multiple Authentication Mechanisms

PP Origin: MDF

FIA_UAU.5.1

The TSF shall provide password and **biometric in accordance with the Biometric Enrollment and Verification, version 1.1** to support user authentication.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the **validation of the user's password, fingerprint, or face.**

Application Note: *The TSS describes authentication rules in more detail.*

6.1.4.19. FIA_UAU.6/CREDENTIAL Re-Authenticating (Credential Change)

PP Origin: MDF

FIA_UAU.6.1/CREDENTIAL

The TSF shall re-authenticate the user via the Password Authentication Factor under the conditions attempted change to any supported authentication mechanisms.

6.1.4.20. FIA_UAU.6/LOCKED Re-Authenticating (TSF Lock)

PP Origin: MDF

FIA_UAU.6.1/LOCKED

The TSF shall re-authenticate the user via an authentication factor defined in FIA_UAU.5.1 under the conditions TSF-initiated lock, user-initiated lock, **no other conditions**.

6.1.4.21. FIA_UAU.7 Protected Authentication

PP Origin: MDF

FIA_UAU.7.1

The TSF shall provide only obscured feedback to the device's display to the user while the authentication is in progress.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.4.22. FIA_UAU_EXT.1 Authentication for Cryptographic Operation

PP Origin: MDF

FIA_UAU_EXT.1.1
The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and **all software-based key storage** at startup.

6.1.4.23. FIA_UAU_EXT.2 Timing of Authentication

PP Origin: MDF

FIA_UAU_EXT.2.1

The TSF shall allow
- **Accessing Medical ID information**
- **Answering calls**
- **Making emergency calls**
- **Using the cameras (unless their use is generally disallowed)**
- **Using the flashlight**
- **Using the control center**
- **Using the notification center**
- **Viewing widgets in Today View and Search**

on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2

The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

6.1.4.24. FIA_X509_EXT.1 X.509 Validation of Certificates

PP Origin: MDF
Applied TDs: TD0689

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The TSF shall validate the revocation status of the certificate using **OCSP as specified in RFC 6960**.
- The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id- kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field. [conditional]
- Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
- OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field. [conditional]

FIA_X509_EXT.1.2

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

6.1.4.25. FIA_X509_EXT.1/WLAN X.509 Certificate Validation

PP Origin: WLANC

FIA_X509_EXT.1.1/WLAN

The TSF shall validate certificates for EAP-TLS in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a certificate in the Trust Anchor Database
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

FIA_X509_EXT.1.2/WLAN

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

6.1.4.26. FIA_X509_EXT.2 X.509 Certificate Authentication

PP Origin: MDF, VPNC

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for mutually authenticated TLS as defined in the

- Functional Package for Transport Layer Security (TLS), version 1.1
- HTTPS
- **IPsec in accordance with the PP-Module for Virtual Private Network (VPN) Clients, version 2.4**

and

- **code signing for system software updates**
- **code signing for mobile applications**
- **code signing for integrity verification**

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the revocation status of a certificate, the TSF shall **not accept the certificate**.

6.1.4.27. FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for WLAN)

PP Origin: WLANC

FIA_X509_EXT.2.1/WLAN

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for EAP-TLS exchanges.

6.1.4.28. FIA_X509_EXT.3 Request Validation of Certificates

PP Origin: MDF

FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2

The TSF shall respond to the requesting application with the success or failure of the validation.

6.1.4.29. FIA_X509_EXT.6 Certificate Storage and Management

PP Origin: WLANC

FIA_X509_EXT.6.1

The TSF shall **store and protect** certificate(s) from unauthorized deletion and modification.

FIA_X509_EXT.6.2

The TSF shall **provide the capability for authorized administrators to load X.509v3 certificates into the TOE** for use by the TSF.

6.1.5. Security Management (FMT)

6.1.5.1. FMT_MOF_EXT.1 Management of Security Functions Behavior

PP Origin: MDF

FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the functions in column 4 of ~~Table 7~~ *Table 16* to the user.

FMT_MOF_EXT.1.2

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall restrict the ability to perform the functions in column 6 of ~~Table 7~~ *Table 16* to the administrator when the device is enrolled and according to the administrator-configured policy.

6.1.5.2. FMT_POL_EXT.2 Agent Trusted Policy Update

PP Origin: Agent
Applied TDs: TD0755

FMT_POL_EXT.2.1

The MDM Agent shall only accept policies and policy updates that are digitally signed by a private key that has been authorized for policy updates by the MDM Server.

FMT_POL_EXT.2.2

The MDM Agent shall not install policies if the signature check fails.

6.1.5.3. FMT_SMF.1 Specification of Management Functions

PP Origin: MDF, VPNC

FMT_SMF.1.1

The TSF shall be capable of performing the following management functions:

Status Markers:
M - Mandatory
O - Optional/Objective
'-' - means that no value (M or O) can be assigned
(Y) - Optional and supported (i.e., Yes)
(N) - Optional and unsupported (i.e., No)

#	Management Function	Impl.	User Only	Admin	Admin Only
1	configure password policy: • Minimum password length • Minimum password complexity • Maximum password lifetime	M	-	M	M
2	configure session locking policy: • Screen-lock enabled/disabled • Screen lock timeout • Number of authentication failures	M	-	M	M
3	enable/disable the VPN protection: • Across device • **no other method**	M	(N)	(Y)	(N)
4	enable/disable **Bluetooth, cellular, satellite, UWB, and Wi-Fi radios**	M	(Y)	(N)	(N)
5(1)	enable/disable **cameras**: • Across device	M	(Y)	(Y)	(N)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

#	Management Function	Impl.	User Only	Admin	Admin Only
5(2)	enable/disable **cameras**: • **on a per-app basis**	M	(Y)	(N)	(N)
6	transition to the locked state	M	-	M	-
7	TSF wipe of protected data	M	-	M	-
8	configure application installation policy by **denying installation of applications**	M	-	M	M
9	import keys or secrets into the secure key storage	M	(Y)	(Y)	-
10	destroy imported keys or secrets and **no other keys/secrets** in the secure key storage	M	(Y)	(Y)	-
11	import X.509v3 certificates into the Trust Anchor Database	M	-	M	(N)
12	remove imported X.509v3 certificates and **no other X.509v3 certificates** in the Trust Anchor Database	M	(N)	(Y)	-
13	enroll the TOE in management	M	(Y)	(N)	(N)
14	remove applications	M	-	M	(N)
15	update system software	M	-	M	(N)
16	install applications	M	-	M	(N)
17	remove Enterprise applications	M	-	M	-
18	enable/disable display notification in the locked state of: • **all notifications**	M	(N)	(Y)	(N)
19	enable data-at-rest protection	M	(N)	(N)	(N)
20	enable removable media's data-at-rest protection	M	(Y)	(N)	(N)
21	enable/disable location services: • Across device • **on a per-app basis**	M	(Y)	(N)	(N)
22	enable/disable the use of **Biometric Authentication Factor**	(Y)	(N)	(Y)	(N)
23	configure whether to allow or disallow establishment of **a TLS trusted channel** if the peer or server certificate is deemed in-valid.	(Y)	(Y)	(N)	(N)
28	wipe Enterprise data	(Y)	(N)	(Y)	-
32	read audit logs kept by the TSF	(Y)	(N)	(Y)	(N)
36	configure the unlock banner	M	-	(Y)	(Y)
41	enable/disable: • Hotspot functionality authenticated by **pre-shared key** • USB tethering authenticated by **no authentication**	(Y)	(N)	(Y)	(N)
44	unenroll the TOE from management	M	(Y)	(N)	(N)
45	enable/disable the Always On VPN protection: • Across device • **no other method** (*mandated by VPNC FMT_SMF.1*)	M	(N)	(Y)	(Y)
47	**enable/disable microphones on a per-app basis**	(Y)	(Y)	(N)	(N)

Table 16: Management Functions (MDF/VPNC)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Application Note: *Most of the administrator management functions are implemented by the specification and installation of configuration profiles. Also, for the enforcement of other functions, such as the password policy, the installation of configuration profiles with dedicated values for some of the payload keys is required.*

For function 19, the TOE always provides data at rest protection of internal memory (i.e., it cannot be managed (enabled or disabled)). The KEKs are protected by the passcode feature in the evaluated configuration.

Function 23 is supported by TLS for the user. Function 23 is not supported by the VPN.

Function 26 has not been included in the table because the TOE does not support a developer mode.

Function 27 has not been included in the table because the TOE (in its evaluated configuration) does not support bypass of local user authentication.

Function 33 has not been included in the table because the feature is not configurable.

All other functions not included in the table are optional.

6.1.5.4.FMT_SMF.1/VPN Specification of Management Functions (VPN)

PP Origin: VPNC

FMT_SMF.1.1/VPN

The TSF shall be capable of performing the following management functions:
- **Specify VPN gateways to use for connections**
- **Specify client credentials to be used for connections**
- **Configure the reference identifier of the peer**

6.1.5.5. FMT_SMF.1/WLAN Specification of Management Functions (WLAN Client)

PP Origin: WLANC
Applied TDs: TD0667, TD0920

FMT_SMF.1.1/WLAN
The TSF shall be capable of performing the following management functions:

Status Markers:
M - Mandatory
O - Optional/Objective
X - Supported
'-' - Prohibited

#	Management Function	Impl.	Admin	User
WL-1	configure security policy for each wireless network: • **specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s),** • security type, • authentication protocol, • client credentials to be used for authentication	M	M	-
WL-2	specify wireless networks (SSIDs) to which the TSF may connect	M	M	-

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

#	Management Function	Impl.	Admin	User
WL-3	enable/disable wireless network bridging capability (for example, bridging a connection between the WLAN and cellular radios to function as a hotspot) authenticated by **passcode**	M	M	X
WL-4	enable/disable certificate revocation list checking	-	-	-
WL-5	disable ad hoc wireless client-to-client connection capability (*a.k.a. Apple AirDrop*)	X	X	X
WL-6	disable roaming capability	X	-	X
WL-7	enable/disable IEEE 802.1X pre-authentication	-	-	-
WL-8	loading X.509 certificates into the TOE	X	X	X
WL-9	revoke X.509 certificates loaded into the TOE	X	X	X
WL-10	enable/disable and configure PMK caching: • set the amount of time (in minutes) for which PMK entries are cached, • set the maximum number of PMK entries that can be cached	-	-	-
WL-11	configure security policy for each wireless network: set wireless frequency band to **2.4 GHz, 5 GHz, 6 GHz**	-	-	-

Table 17: Management Functions (WLANC)

6.1.5.6. FMT_SMF.1/BT Specification of Management Functions

PP Origin: BT

FMT_SMF_EXT.1.1/BT

The TSF shall be capable of performing the following Bluetooth management functions:

Status Markers:

M - Mandatory

X - Supported

'-' – Prohibited

Function	Impl.	User Only	Admin	Admin Only
BT-1. Configure the Bluetooth trusted channel. • Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes;	M	X	-	-
BT-2. Change the Bluetooth device name (separately for BR/EDR and LE);	-	-	-	-
BT-3. Provide separate controls for turning the BR/EDR and LE radios on and off;	-	-	-	-
BT-4. Allow/disallow the following additional wireless technologies to be used with Bluetooth: no other wireless technologies;	-	-	-	-
BT-5. Configure allowable methods of Out of Band pairing (for BR/EDR and LE);	-	-	-	-

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Function	Impl.	User Only	Admin	Admin Only
BT-6. Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes separately;	-	-	-	-
BT-7. Disable/enable the Connectable mode (for BR/EDR and LE);	-	-	-	-
BT-8. Disable/enable the Bluetooth list of Bluetooth service and/or profiles available on the OS (for BR/EDR and LE);	-	-	-	-
BT-9. Specify minimum level of security for each pairing (for BR/EDR and LE);	-	-	-	-

Table 18: Management Functions (BT)

Application Note: *The optional management functions are not supported.*

6.1.5.7. FMT_SMF.2 Specification of Remediation Actions

PP Origin: MDF

FMT_SMF_EXT.2.1

The TSF shall offer
- **remove Enterprise applications**
- **remove all device-stored Enterprise resource data**

upon unenrollment and **no other triggers**.

6.1.5.8. FMT_SMF.4 Specification of Management Functions

PP Origin: Agent
Applied TDs: TD0755

FMT_SMF_EXT.4.1

The MDM Agent shall be capable of interacting with the platform to perform the following functions:

- **Import the certificates to be used for authentication of MDM Agent communications**
- **administrator-provided management functions in MDF PP**
- **no additional functions**.

FMT_SMF_EXT.4.2

The MDM Agent shall be capable of performing the following functions:

- Enroll in management
- Configure whether users can unenroll from management
- **no other functions**.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.5.9. FMT_UNR_EXT.1 User Unenrollment Prevention

PP Origin: Agent

FMT_UNR_EXT.1.1

The MDM Agent shall provide a mechanism to enforce the following behavior upon an attempt to unenroll the mobile device from management:

- **prevent the unenrollment from occurring**
- **apply remediation actions**

6.1.6. Protection of the TSF (FPT)

6.1.6.1. FPT_AEX_EXT.1 Application Address Space Layout Randomization

PP Origin: MDF

FPT_AEX_EXT.1.1

The TSF shall provide address space layout randomization ASLR to applications.

FPT_AEX_EXT.1.2

The base address of any user-space memory mapping will consist of at least 8 unpredictable bits.

6.1.6.2. FPT_AEX_EXT.2 Memory Page Permissions

PP Origin: MDF

FPT_AEX_EXT.2.1

The TSF shall be able to enforce read, write, and execute permissions on every page of physical memory.

6.1.6.3. FPT_AEX_EXT.3 Stack Overflow Protection

PP Origin: MDF

FPT_AEX_EXT.3.1

TSF processes that execute in a non-privileged execution domain on the application processor shall implement stack-based buffer overflow protection.

6.1.6.4. FPT_AEX_EXT.4 Domain Isolation

PP Origin: MDF

FPT_AEX_EXT.4.1

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

6.1.6.5. FPT_BDP_EXT.1 Biometric Data Processing

PP Origin: BIO

FPT_BDP_EXT.1.1

Processing of plaintext biometric data shall be inside the SEE in runtime.

FPT_BDP_EXT.1.2

Transmission of plaintext biometric data between the capture sensor and the SEE shall be isolated from the main computer operating system on the TSF in runtime.

6.1.6.6. FPT_JTA_EXT.1 JTAG Disablement

PP Origin: MDF

FPT_JTA_EXT.1.1

The TSF shall **disable access through hardware** to JTAG.

6.1.6.7. FPT_KST_EXT.1 Key Storage

PP Origin: MDF, BIO
FPT_KST_EXT.1.1

The TSF shall not store any plaintext key material or biometric data in readable non-volatile memory.

6.1.6.8. FPT_KST_EXT.2 No Key Transmission

PP Origin: MDF, BIO

FPT_KST_EXT.2.1

The TSF shall not transmit any plaintext key material or biometric data outside the security boundary of the TOE.

6.1.6.9. FPT_KST_EXT.3 No Plaintext Key Export

PP Origin: MDF

FPT_KST_EXT.3.1

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall ensure it is not possible for the TOE users to export plaintext keys.

6.1.6.10. FPT_NOT_EXT.1 Self-Test Notification

PP Origin: MDF

FPT_NOT_EXT.1.1

The TSF shall transition to non-operational mode and no other actions when the following types of failures occur:
- failures of the self-tests
- TSF software integrity verification failures
- **no other failures**

6.1.6.11. FPT_PBT_EXT.1 Protection of Biometric Template

PP Origin: BIO
Applied TDs: TD0714

FPT_PBT_EXT.1.1

The TSF shall protect the biometric template **using a password as an additional factor**.

6.1.6.12. FPT_STM.1 Reliable Time Stamps

PP Origin: MDF

FPT_STM.1.1

The TSF shall be able to provide reliable time stamps for its own use.

6.1.6.13. FPT_TST_EXT.1 TSF Cryptographic Functionality Testing

PP Origin: MDF

FPT_TST_EXT.1.1

The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of all cryptographic functionality.

6.1.6.14. FPT_TST_EXT.1/VPN TSF Self-Test

PP Origin: VPNC

FPT_TST_EXT.1.1/VPN

The **TOE** shall run a suite of self tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2/VPN

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The **TOE** shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the **cryptographic services specified in** FCS_COP.1/HASH , FCS_COP.1/ SIGN , FCS_COP.1/KEY-HMAC , or FIA_X509_EXT.1.

6.1.6.15. FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)

PP Origin: MDF

FPT_TST_EXT.2.1/PREKERNEL

The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel stored in mutable media prior to its execution through the use of **a digital signature using a hardware-protected asymmetric key**.

6.1.6.16. FPT_TST_EXT.2/POSTKERNEL TSF Integrity Checking (Post-Kernel)

PP Origin: MDF

FPT_TST_EXT.2.1/POSTKERNEL

The TSF shall verify the integrity of **applications** stored in mutable media prior to its execution through the use of **a digital signature using a hardware-protected asymmetric key**.

6.1.6.17. FPT_TST_EXT.3 TSF Integrity Testing

PP Origin: MDF

FPT_TST_EXT.3.1

The TSF shall not execute code if the code signing certificate is deemed invalid.

6.1.6.18. FPT_TST_EXT.3/WLAN TSF Cryptographic Functionality Testing (WLAN Client)

PP Origin: WLANC

FPT_TST_EXT.3.1/WLAN

The **TOE** shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.3.2/WLAN

The **TOE** shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic services.

6.1.6.19. FPT_TUD_EXT.1 TSF Version Query

PP Origin: MDF

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FPT_TUD_EXT.1.1

The TSF shall provide authorized users the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2

The TSF shall provide authorized users the ability to query the current version of the hardware model of the device.

FPT_TUD_EXT.1.3

The TSF shall provide authorized users the ability to query the current version of installed mobile applications.

6.1.6.20. FPT_TUD_EXT.2 TSF Update Verification

PP Origin: MDF

FPT_TUD_EXT.2.1

The TSF shall verify software updates to the Application Processor system software and **no other processor system software** using a digital signature verified by the manufacturer trusted key prior to installing those updates.

FPT_TUD_EXT.2.2

The TSF shall **never update** the TSF boot integrity **key**.

FPT_TUD_EXT.2.3

The TSF shall verify that the digital signature verification key used for TSF updates **matches an immutable hardware public key**.

6.1.6.21. FPT_TUD_EXT.3 Application Signing

PP Origin: MDF

FPT_TUD_EXT.3.1

The TSF shall verify mobile application software using a digital signature mechanism prior to installation.

6.1.6.22. FPT_TUD_EXT.4 Trusted Update Verification

PP Origin: MDF

FPT_TUD_EXT.4.1

The TSF shall not install code if the code signing certificate is deemed invalid.

6.1.6.23. FPT_TUD_EXT.5 Application Verification

PP Origin: MDF

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

FPT_TUD_EXT.5.1

The TSF shall by default only install mobile applications cryptographically verified by **a built-in X.509v3 certificate**.

6.1.6.24. FPT_TUD_EXT.6 Trusted Update Verification

PP Origin: MDF

FPT_TUD_EXT.6.1

The TSF shall verify that software updates to the TSF are a current or later version than the current version of the TSF.

6.1.7. TOE Access (FTA)

6.1.7.1. FTA_SSL_EXT.1 TSF- and User-initiated Locked State

PP Origin: MDF

FTA_SSL_EXT.1.1

The TSF shall transition to a locked state after a time interval of inactivity.

FTA_SSL_EXT.1.2

The TSF shall transition to a locked state after initiation by either the user or the administrator.

FTA_SSL_EXT.1.3

The TSF shall, upon transitioning to the locked state, perform the following operations:
a) Clearing or overwriting display devices, obscuring the previous contents;
b) **Zeroize the decrypted class key for the NSFileProtectionComplete class.**

6.1.7.2. FTA_TAB.1 Default TOE Access Banners

PP Origin: MDF

FTA_TAB.1.1

Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE.

6.1.7.3. FTA_WSE_EXT.1 Wireless Network Access

PP Origin: WLANC

FTA_WSE_EXT.1.1

The TSF shall be able to attempt connections only to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF.1/WLAN.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

6.1.8. Trusted Path/Channels (FTP)

6.1.8.1. FTA_BLT_EXT.1 Bluetooth Encryption

PP Origin: BT

FTP_BLT_EXT.1.1

The TSF shall enforce the use of encryption when transmitting data over the Bluetooth trusted channel for BR/ EDR and **LE**.

FTP_BLT_EXT.1.2

The TSF shall use key pairs per FCS_CKM_EXT.8 for Bluetooth encryption.

6.1.8.2. FTA_BLT_EXT.2 Persistence of Bluetooth Encryption

PP Origin: BT

FTP_BLT_EXT.2.1

The TSF shall **terminate the connection** if the remote device stops encryption while connected to the TOE.

6.1.8.3. FTA_BLT_EXT.3/BR Bluetooth Encryption Parameters (BR/EDR)

PP Origin: BT

FTP_BLT_EXT.3.1/BR

The TSF shall set the minimum encryption key size to **128 bits** for BR/EDR and not negotiate encryption key sizes smaller than the minimum size.

6.1.8.4. FTA_BLT_EXT.3/LE Bluetooth Encryption Parameters (LE)

PP Origin: BT

FTP_BLT_EXT.3.1/LE

The TSF shall set the minimum encryption key size to **128 bits** for LE and not negotiate encryption key sizes smaller than the minimum size.

6.1.8.5. FTP_ITC.1/WLAN Trusted Channel Communication (Wireless LAN)

PP Origin: WLANC

FTP_ITC.1.1/WLAN

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall use 802.11-2012, 802.1X, and EAP-TLS to provide a trusted communication channel between itself and a wireless access point that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2/WLAN

The TSF shall permit the TSF to initiate communication via the trusted channel.

FTP_ITC.1.3/WLAN

The TSF shall initiate communication via the trusted channel for wireless access point connections.

6.1.8.6. FTP_ITC_EXT.1 Trusted Channel Communication

PP Origin: MDF, VPNC

FTP_ITC_EXT.1.1

The TSF shall use
- 802.11-2012 in accordance with the PP-Module for Wireless LAN Clients, version 1.0,
- 802.1X in accordance with the PP-Module for Wireless LAN Clients, version 1.0,
- EAP-TLS in accordance with the PP-Module for Wireless LAN Clients, version 1.0,
- Mutually authenticated TLS in accordance with the Functional Package for Transport Layer Security (TLS), version 1.1
  and
- **IPsec in accordance with the PP-Module for Virtual Private Network (VPN) Clients, version 2.5**
- **HTTPS**

protocols to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FTP_ITC_EXT.1.2

The TSF shall permit the TSF to initiate communication via the trusted channel.

FTP_ITC_EXT.1.3

The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and **OTA updates**.

6.1.8.7. FTP_ITC_EXT.1(2) Trusted Channel Communication

PP Origin: Agent

FTP_ITC_EXT.1.1(2)

The TSF shall use **HTTPS** to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FTP_ITC_EXT.1.2(2)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TSF shall permit the TSF and the MDM Server and **no other IT entities** to initiate communication via the trusted channel.

FTP_ITC_EXT.1.3(2)

The TSF shall initiate communication via the trusted channel for all communication between the MDM Agent and the MDM Server and **no other communication**

6.1.8.8. FTP_TRP.1(2) Trusted Path (for Enrollment)

PP Origin: Agent

FTP_TRP.1.1(2)

The TSF shall use **HTTPS** to provide a trusted communication path between itself and another trusted IT product that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data from modification, disclosure.

FTP_TRP.1.2(2)

The TSF shall permit MD users to initiate communication via the trusted path.

FTP_TRP.1.3(2)

The TSF shall require the use of the trusted path for all MD user actions.

6.2. Security Functional Requirements Rationale

The SFR rationale is defined in the documents specified in Section 2 "CC Conformance Claim".

6.3. Security Assurance Requirements

The following table shows the SARs for the TOE, which are taken from the MDF PP.

Security Assurance Class	Security Assurance Requirement	Source
ALC Life-cycle Support	ALC_TSU_EXT.1 Timely Security Updates	MDF
	ALC_CMC.1 Labelling of the TOE	
	ALC_CMS.1 TOE CM coverage	
ASE Security Target Evaluation	ASE_CCL.1 Conformance claims	
	ASE_ECD.1 Extended components definition	
	ASE_INT.1 ST introduction	
	ASE_OBJ.1 Security objectives for the operational environment	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Security Assurance Class	Security Assurance Requirement	Source
	ASE_REQ.1 Stated security requirements	
	ASE_SPD.1 Security problem definition	
	ASE_TSS.1 TOE summary specification	
ADV Development	ADV_FSP.1 Basic functional specification	
AGD Guidance Documents	AGD_OPE.1 Operational user guidance	
	AGD_PRE.1 Preparative procedures	
ATE Tests	ATE_IND.1 Independent testing - conformance	
AVA Vulnerability Assessment	AVA_VAN.1 Vulnerability survey	

Table 19: SARs

6.4. Security Assurance Requirements Rationale

The SAR rationale is defined in the documents specified in Section 2 "CC Conformance Claim".

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7. TOE Summary Specification

Table 20 provides a mapping of SFRs to the TSS information.

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FAU – Security Audit		
FAU_ALT_EXT.2	Describes how the alerts are implemented, how the candidate policy updates are obtained; and the actions that take place for successful (policy update installed) and unsuccessful (policy update not installed) cases. Identifies the software components that are performing the processing. Describes how reachability events are implemented, and if configurable is selected in FMT_SMF_EXT.4.2. Clearly indicates who (MDM Agent or MDM Server) initiates reachability events. Describes under what circumstances, if any, the alert may not be generated, how alerts are queued, and t	7.1.8 "Trusted Path/Channels (FTP)" 7.1.9.2 "MDM Agent Alerts" Table 31 "MDM Agent Status Commands" 7.1.9.2.3 "Alerts on receiving periodic reachability events" 7.1.9.2.1 "Queuing of Alerts"
FAU_GEN.1	Lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. Provides that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2.	7.1.9.1 "Audit Records" Table 10 "Mandatory Auditable Events (MDF)"
FAU_GEN.1(2)	Provides a format for audit records. Provides all format types, with brief description of each field.	7.1.9.1 "Audit Records" Table 11 "Auditable Events (Agent)"
FAU_GEN.1/BT	See TSS requirements for FAU_GEN.1.	7.1.9.1 "Audit Records" Table 12 "Auditable Events (BT)"
FAU_GEN.1/VPN	Describes the auditable events and the component that is responsible for each type of auditable event.	7.1.9.1 "Audit Records" Table 13 "Auditable Events (VPN)" 7.1.8.4.2 “IPsec VPN Client”
FAU_GEN.1/WLAN	Provides a format for audit records, including each audit record format type, along with a brief description of each field.	7.1.9.1 "Audit Records"
FAU_SAR.1	There are no TSS evaluation activities for this SFR.	
FAU_SEL.1(2)	There are no TSS evaluation activities for this SFR.	
FAU_STG.1	Lists the location of all logs and the access controls of those files such that unauthorized modification and deletion are prevented.	7.1.9.1 "Audit Records"
FAU_STG.4	Describes the size limits on the audit records, the detection of a full audit trail, and the action(s) taken by the TSF when the audit trail is full. The action(s) results in the deletion or overwrite of the oldest stored record.	7.1.9.1 "Audit Records" 7.1.5.2 "Configuration Profiles"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS – Cryptographic Support		
FCS_CKM.1	Identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, it identifies the usage for each scheme.	Table 23 "Explanation of usage for cryptographic functions in the cryptographic modules"
FCS_CKM.1/VPN	Describes how the key generation functionality is invoked.	7.1.8.4.2 "IPsec VPN Client"
FCS_CKM.1/WPA	Describes how the primitives defined and implemented by this EP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. Provides a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed.	7.1.8.3 "Wireless LAN (WLAN)" A.2 "Wi-Fi Alliance Certificates"
FCS_CKM.2/UNLOCKED	Demonstrates that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. Identifies the usage for each scheme.	7.1.2.1 "Overview of Key Management" 7.1.2.2 "Password based key derivation" 7.1.2.6 "Explanation of usage for cryptographic functions"
FCS_CKM.2/LOCKED	There are no TSS evaluation activities for this SFR.	
FCS_CKM.2.1/WLAN	Describes how the Group Temporal Key (GTK) is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this EP.	7.1.8.3 "Wireless LAN (WLAN)"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_CKM_EXT.1	Shows that a REK is supported by the TOE. Includes a description of the protection provided by the TOE for a REK. Includes a description of the method of generation of a REK. Describes how any reading, import, and export of that REK is prevented. Describes how encryption/decryption/derivation actions are isolated so as to prevent applications and system-level processes from reading the REK while allowing encryption/decryption/derivation by the key. Describes how the OS is prevented from accessing the memory containing REK key material, which software is allowed access to the REK, how any other software in the execution environment is prevented from reading that key material, and what other mechanisms prevent the REK key material from being written to shared memory locations between the OS and the separate execution environment. Describes the key derivation function and the approved derivation mode and the key expansion algorithm according to FCS_CKM_EXT.3.2. Documents that the generation of a REK meets the FCS_RBG_EXT.1.1 and FCS_RBG_EXT.1.2 requirements. Describes the generation mechanism including what triggers a generation, how the functionality described by FCS_RBG_EXT.1 is invoked, and whether a separate instance of the RBG is used for REK(s).	7.1.1.1 "The Secure Enclave Processor (SEP)" 7.1.2.1 "Overview of Key Management" Figure 5 "Key Hierarchy in the TOE OS" The proprietary Entropy Assessment Report (EAR) (on file with NIAP) has analyzed the random bit generator (RBG) used in the production environment for compliance to the requirements defined in FCS_RBG_EXT.1.
FCS_CKM_EXT.2	Describes how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs.	Figure 5 "Key Hierarchy in the TOE OS" 7.1.1 "Hardware Protection Functions" 7.1.2 "Cryptographic Support (FCS)"
FCS_CKM_EXT.3	Describes the formation of all key encryption keys (KEKs) and that the key sizes match those described by the ST author. Describes that each key (DEKs, software-based key storage, and KEKs) is encrypted by keys of equal or greater security strength using one of the selected methods. If a KDF is used, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP800-108.	7.1.2.1 "Overview of Key Management" Figure 5 "Key Hierarchy in the TOE OS" This RBG in the Secure Enclave has been analyzed for compliance with the requirements of FCS_RBG_EXT.1/HW in the proprietary EAR, which has been provided to NIAP.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_CKM_EXT.4	Lists each type of plaintext key material (DEKs, software-based key storage, KEKs, trusted channel keys, passwords, etc.) and its generation and storage location. Describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state, etc.). Lists, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase). Describes the clearing procedure in terms of the memory in which the data are stored.	7.1.2.1 "Overview of Key Management" Table 21 "Summary of keys and persistent secrets in the TOE OS" Table 22 "Summary of keys and persistent secrets used by the MDM Agent"
FCS_CKM_EXT.5	Describes how the device is wiped; and the type of clearing procedure that is performed (cryptographic erase or overwrite) and, if overwrite is performed, the overwrite procedure (overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase). Describes the clearing procedure in terms of the memory in which the data are stored.	7.1.2.1 "Overview of Key Management" Figure 5 "Key Hierarchy in the TOE OS"
FCS_CKM_EXT.6	Contains a description regarding the salt generation, including which algorithms on the TOE require salts. The salt is generated using an RBG described in FCS_RBG_EXT.1. For PBKDF derivation of KEKs, this assurance activity may be performed in conjunction with FCS_CKM_EXT.3.2.	7.1.1 "Hardware Protection Functions"
FCS_CKM_EXT.7	See FCS_CKM_EXT.1 in this table.	See FCS_CKM_EXT.1 in this table.
FCS_CKM_EXT.8	Describes the criteria used to determine the frequency of generating new ECDH public/private key pairs and does not permit the use of static ECDH key pairs.	7.1.8.2 "Bluetooth"
FCS_COP.1/ENCRYPT	There are no TSS evaluation activities for this SFR.	
FCS_COP.1/HASH	Documents the association of the hash function with other TSF cryptographic functions.	7.1.2.6 "Explanation of usage for cryptographic functions"
FCS_COP.1/SIGN	There are no TSS evaluation activities for this SFR.	
FCS_COP.1/KEY-HMAC	Specifies the following values used by the keyed-hash message authentication code (HMAC) function: key length, hash function used, block size, and output MAC length used.	Table 23 "Explanation of usage for cryptographic functions in the cryptographic modules" 7.1.2 "Cryptographic Support (FCS)" 7.1.3.8 "Keyed Hash"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_COP.1/CONDITION	Describes the method by which the password is first encoded and then fed to the SHA algorithm. Describes the settings for the algorithm (padding, blocking, etc.) and are supported by the selections in this component as well as the selections concerning the hash function itself. Describes how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.3.	7.1.2.1 "Overview of Key Management" 7.1.2.2 "Password based key derivation"
FCS_HTTPS_EXT.1	There are no TSS evaluation activities for this SFR.	
FCS_IPSEC_EXT.1.1	Describes how the IPsec capabilities are implemented and how a packet is processed. Details the relationship between the client and the underlying platform, including which aspects are implemented by the client, and those that are provided by the underlying platform. Describes how the client interacts with the platforms network stack. If the security policy database (SPD) is implemented by the client, then the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. Describes the rules that are available and the resulting actions available after matching a rule. Describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS, DISCARD, and PROTECT actions is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. The description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no security association (SA) is established on the interface or for that particular packet) as well as packets that are part of an established SA. If the SPD is implemented by the underlying platform, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.	7.1.8.4 "VPN" 7.1.8.4.1 "AlwaysOn VPN" 7.1.8.4.2 “IPsec VPN Client”
FCS_IPSEC_EXT.1.2	States that the VPN can be established to operate in tunnel mode and/or transport mode (as selected).	7.1.8.4 "VPN" 7.1.8.4.2 “IPsec VPN Client”
FCS_IPSEC_EXT.1.3	Describes how a packet is processed against the SPD and that if no "rules" are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.	7.1.8.4 "VPN" 7.1.8.4.2 “IPsec VPN Client”

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_IPSEC_EXT.1.4	States that the algorithms AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256 are implemented.	7.1.8.4 "VPN" 7.1.8.4.2 “IPsec VPN Client”
FCS_IPSEC_EXT.1.5	States that IKEv2 is implemented.	
FCS_IPSEC_EXT.1.6	Identifies the algorithms used for encrypting the IKEv2 payload (AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256).	
FCS_IPSEC_EXT.1.7	There are no TSS evaluation activities for this SFR.	
FCS_IPSEC_EXT.1.8	Lists the supported DH groups. Describes how a particular DH group is specified/ negotiated with a peer.	7.1.8.4 "VPN" 7.1.8.4.2 “IPsec VPN Client” 7.1.8.4.3 “IKEv2, cryptography and authentication mechanisms”
FCS_IPSEC_EXT.1.9 FCS_IPSEC_EXT.1.10	Describes, for each DH group supported, the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. Indicates that the random number generated that meets the requirements in this PP-Module is used, and that the length of "x" and the nonces meet the stipulations in the requirement.	7.1.8.4 "VPN" 7.1.8.4.3 “IKEv2, cryptography and authentication mechanisms”
FCS_IPSEC_EXT.1.11 FCS_IPSEC_EXT.1.12 FCS_IPSEC_EXT.1.13	The evaluator shall ensure that the TSS describes whether peer authentication is performed using RSA, ECDSA, or both. If any selection with pre-shared keys is chosen in the selection, the evaluator shall check to ensure that the TSS describes how those selections work in conjunction with authentication of IPsec connections. The evaluator shall ensure that the TSS describes how the TOE compares the peer’s presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which fields of the certificate are used as the presented identifier (DN, Common Name, or SAN), and if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer’s presented certificate.	7.1.8.4 "VPN" 7.1.8.4.2 “IPsec VPN Client” 7.1.8.4.3 “IKEv2, cryptography and authentication mechanisms”
FCS_IPSEC_EXT.1.14	Describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. Describes the checks that are done when negotiating IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.	7.1.8.4 "VPN" 7.1.8.4.3 “IKEv2, cryptography and authentication mechanisms”
FCS_IV_EXT.1	Describes the encryption of all keys. Describes that the formation of the IVs for each key encrypted by the same KEK meets FCS_IV_EXT.1.	7.1.2.1 "Overview of Key Management" Figure 5 "Key Hierarchy in the TOE OS"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_RBG_EXT.1/HW FCS_RBG_EXT.1/SW	There are no TSS evaluation activities for this SFR.	A proprietary Entropy Assessment Report (EAR) has been produced and is on file with NIAP.
FCS_SRV_EXT.1	There are no TSS evaluation activities for this SFR.	
FCS_STG_EXT.1	Describes that the TOE implements the required secure key storage. Contains a description of the key storage mechanism that justifies the selection of "mutable hardware" or "software-based".	7.1.2.1 "Overview of Key Management" 7.1.3.6 "Keychain Data Protection"
FCS_STG_EXT.2	Includes a key hierarchy description of the protection of each DEK for data at rest, of software-based key storage, of long-term trusted channel keys, and of KEK related to the protection of the DEKs, long-term trusted channel keys, and software-based key storage. This description includes a diagram of the hierarchy implemented by the TOE indicates how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs (FCS_CKM_EXT.2), the key size (FCS_CKM_EXT.2 and FCS_CKM_EXT.3) for each key, how each KEK is formed (generated, derived, or combined according to FCS_CKM_EXT.3), the integrity protection method for each encrypted key (FCS_STG_EXT.3), and the IV generation for each key encrypted by the same KEK (FCS_IV_EXT.1). States in the key hierarchy description in that each DEK and software-stored key is encrypted according to FCS_STG_EXT.2.	7.1.2.1 "Overview of Key Management" Figure 5 "Key Hierarchy in the TOE OS"
FCS_STG_EXT.3	States in the key hierarchy description that each encrypted key is integrity protected according to one of the options in FCS_STG_EXT.3.	7.1.2.1 "Overview of Key Management"
FCS_STG_EXT.4	Lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST, for what purpose it is used, and, for each platform listed as supported in the ST, how it is stored. States that the MDM Agent calls a platform-provided API to store persistent secrets and private keys.	7.1.2.1 "Overview of Key Management" 7.1.2.2 "Password based key derivation" 7.1.2.4 "Storage of Persistent Secrets and Private Keys by the MDM Agent"
FCS_TLS_EXT.1	There are no TSS evaluation activities for this SFR.	
FCS_TLSC_EXT.1.1	Provides a description of the implementation of this protocol and the cipher suites supported.	7.1.8.1 "EAP-TLS and TLS"
FCS_TLSC_EXT.1.2	Describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported and whether IP addresses and wildcards are supported. Identifies whether and the manner in which certificate pinning is supported or used by the TOE.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FCS_TLSC_EXT.1.3	Describes how and when user or administrator authorization is obtained. Describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.	
FCS_TLSC_EXT.1/WLAN	Describes the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The cipher suites specified include those listed for this component.	7.1.4.2 "X.509v3 Certificates" 7.1.8.1 "EAP-TLS and TLS"
FCS_TLSC_EXT.2	Describes for FIA_X509_EXT.2.1 the use of client-side certificates for TLS mutual authentication. Describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.	7.1.8.1.1 "TLS mutual authentication" 7.1.4.2 "X.509v3 Certificates"
FCS_TLSC_EXT.2/WLAN	Describes the Supported Groups Extension and indicates whether the required behavior is performed by default or may be configured.	7.1.8.1 "EAP-TLS and TLS"
FCS_TLSC_EXT.4	There are no TSS evaluation activities for this SFR.	
FCS_TLSC_EXT.5	Describes the Supported Groups Extension.	7.1.8.1 "EAP-TLS and TLS"
FCS_WPA_EXT.1	There are no TSS evaluation activities for this SFR.	
FDP – User Data Protection		
FDP_ACF_EXT.1.1	Lists all system services available for use by an application. Describes how applications interface with these system services, and the means by which these system services are protected by the TSF. Describes which of the following categories each system service falls in. • No applications are allowed access • Privileged applications are allowed access • Applications are allowed access by user authorization • All applications are allowed access Describes how privileges are granted to third-party applications. Describes for both types of privileged applications, how and when the privileges are verified and how the TSF prevents unprivileged applications from accessing those services. Identifies for any services for which the user may grant access, whether the user is prompted for authorization when the application is installed, or during runtime.	7.1.3.1 "Protection of Files" 7.1.3.2 "Application Access to Files" 7.1.3.5 "Restricting Applications Access to Services"
FDP_ACF_EXT.1.2	Describes which data sharing is permitted between applications, which data sharing is not permitted, and how disallowed sharing is prevented.	
FDP_ACF_EXT.2	There are no TSS evaluation activities for this SFR.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FDP_DAR_EXT.1	Indicates which data is protected by the DAR implementation and what data is considered TSF data. This data includes all protected data.	7.1.2.1 "Overview of Key Management" 7.1.3.6 "Keychain Data Protection" Figure 5 "Key Hierarchy in the TOE OS"
FDP_DAR_EXT.2.1	Describes which data stored by the TSF is treated as sensitive. Describes the mechanism that is provided for applications to use to mark data and keys as sensitive. Contains information reflecting how data and keys marked in this manner are distinguished from data and keys that are not.	7.1.3.6 "Keychain Data Protection" Table 24 "Keychain to File-system Mapping"
FDP_DAR_EXT.2.2	Describes the process of receiving sensitive data while the device is in a locked state. Indicates if sensitive data that may be received in the locked state are treated differently than sensitive data that cannot be received in the locked state. Describes the key scheme for encrypting and storing the received data, which must involve an asymmetric key and must prevent the sensitive data at rest from being decrypted by wiping all key material used to derive or encrypt the data.	
FDP_DAR_EXT.2.3	Includes the symmetric encryption keys in the key hierarchy section for (DEKs) used to encrypt sensitive data. Includes the protection of any private keys of the asymmetric pairs. Describes that any private keys that are not wiped and are stored by the TSF are stored encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived or biometric-unlocked KEK.	
FDP_DAR_EXT.2.4	Includes a description of the actions taken by the TSF for the purposes of DAR upon transitioning to the unlocked state. Describes that these actions minimally include decrypting all received data using the asymmetric key scheme and re-encrypting with the symmetric key scheme used to store data while the device is unlocked.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FDP_IFC_EXT.1	Describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. Indicates which traffic does not go through the VPN and which traffic does and that a configuration exists for each baseband protocol in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) or needed for the correct functioning of the TOE is not encapsulated by the VPN protocol (IPsec). Describes any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or, LTE).	7.1.8.4.1 "AlwaysOn VPN"
FDP_RIP.2	Describes the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.	7.1.8.4.6 "Residual information protection and packet processing"
FDP_STG_EXT.1	Describes the Trust Anchor Database implemented that contain certificates used to meet the requirements of this PP. Contains information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access in accordance with the permissions established in FMT_SMF.1 and FMT_MOF_EXT.1.	7.1.4.2 "X.509v3 Certificates"
FDP_UPC_EXT.1/APPS FDP_UPC_EXT.1/BLUETOOTH	Describes that all protocols listed in the TSS are specified and included in the requirements in the ST.	7.1.8 "Trusted Path/Channels (FTP)"
FDP_VPN_EXT.1	The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does and that a configuration exists for each baseband protocol in which only the traffic identified by the ST author is necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec). The ST author shall also identify in the TSS section any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi, LTE).	7.1.8.4.1 "AlwaysOn VPN"
FIA – Identification and Authentication		

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FIA_AFL_EXT.1	Describes that a value corresponding to the number of unsuccessful authentication attempts since the last successful authentication is kept for each Authentication Factor interface. Describes if and how this value is maintained when the TOE loses power, either through a graceful powered off or an ungraceful loss of power and that if the value is not maintained, the interface is after another interface in the boot sequence for which the value is maintained. Describes how the unsuccessful authentication attempts for each mechanism selected in FIA_UAU.5.1 is handled. Describes if each authentication mechanism utilizes its own counter or if multiple authentication mechanisms utilize a shared counter. If multiple authentication mechanisms utilize a shared counter, the evaluator shall verify that the TSS describes this interaction. Describes how the process used to determine if the authentication attempt was successful and that that the counter would be updated even if power to the device is cut immediately following notifying the TOE user if the authentication attempt was successful or not.	7.1.5.2 "Configuration Profiles" 7.1.4 "Identification and Authentication (FIA)"
FIA_BLT_EXT.1	Describes when user permission is required for Bluetooth pairing, and that this description mandates explicit user authorization via manual input for all Bluetooth pairing, including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed.	7.1.8.2 "Bluetooth"
FIA_BLT_EXT.2	Describes how data transfer of any type is prevented before the Bluetooth pairing is completed. Specifically calls out any supported radio frequency communication (RFCOMM) and L2CAP data transfer mechanisms.	
FIA_BLT_EXT.3	Describes how Bluetooth connections are maintained such that two devices with the same Bluetooth device address are not simultaneously connected and such that the initial connection is not superseded by any following connection attempts.	
FIA_BLT_EXT.4	Describes the secure simple pairing process.	
FIA_BLT_EXT.6	Describes all Bluetooth profiles and associated services for which explicit user authorization is required before a remote device can gain access. Describes any difference in behavior based on whether or not the device has a trusted relationship with the TOE for that service (i.e. whether there are any services that require explicit user authorization for untrusted devices that do not require such authorization for trusted devices). Describes the method by which a device can become 'trusted'.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FIA_BLT_EXT.7	See TSS requirements for FIA_BLT_EXT.6.	
FIA_ENR_EXT.2	Describes which types of reference identifiers are acceptable and how the identifier is specified.	7.1.4.3 "MDM Server Reference ID" Table 25 "MDM Server Reference Identifiers"
FIA_MBE_EXT.1	Explains how the TOE meets FIA_MBE_EXT.1 at high level description and describes how the TOE enrolls a user.	7.1.5.3 "Biometric Authentication Factors (BAFs)"
FIA_MBE_EXT.2	Explains how the TOE meets FIA_MBE_EXT.2 at high level description. If standard quality metrics are selected and assigned, the TSS shall include information (e.g. name of quality metrics and section numbers that define the metrics in the standard) to identify quality metrics that the TOE implements. If a developer defined quality assessment is selected, the TSS shall include an overview of the quality metrics used for the assessment. Between the TSS and the Biometric Management Design (BMD), describes how the TOE generates templates of sufficient quality from samples at enrolment.	7.1.4.1.2 "Biometric Sample Quality"
FIA_MBV_EXT.1	Explains how the TOE meets FIA_MBV_EXT.1 at high level description and, between the TSS and the Biometric Management Design (BMD), describes how the TOE verifies a user with one's biometric characteristics.	7.1.5.3 "Biometric Authentication Factors (BAFs)" 7.1.4.1.1 "Accuracy of Biometric Authentication"
FIA_MBV_EXT.2	Explains how the TOE meets FIA_MBV_EXT.2 at high level description. If standard quality metrics are selected and assigned, it includes information (e.g. name of quality metrics and section numbers that define the metrics in the standard) to identify quality metrics that the TOE implements. If a developer defined quality assessment is selected, it includes an overview of the quality metrics used for the assessment. Between the TSS and the Biometric Management Design (BMD), describes how the TOE checks the quality of the samples captured.	7.1.4.1.2 "Biometric Sample Quality"
FIA_PAE_EXT.1	There are no TSS evaluation activities for this SFR.	
FIA_PMG_EXT.1	There are no TSS evaluation activities for this SFR.	
FIA_PSK_EXT.1	Describes the use pre-shared keys, generated bit-based keys, as part of the VPN key management protocol, here ikev2.	7.1.4 "Identification and Authentication (FIA)"
FIA_PSK_EXT.2	Describes how the VPN key management protocol, here ikev2, is able to accept externally generated pre-shared keys.	7.1.4 "Identification and Authentication (FIA)"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FIA_TRT_EXT.1	Describes the method by which authentication attempts are not able to be automated. Describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that this delay totals at least 500 milliseconds over 10 attempts for all authentication mechanisms selected in FIA_UAU.5.1.	7.1.4 "Identification and Authentication (FIA)"
FIA_UAU.5	Describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.	
FIA_UAU.6/CREDENTIAL	There are no TSS evaluation activities for this SFR.	
FIA_UAU.6/LOCKED	There are no TSS evaluation activities for this SFR.	
FIA_UAU.7	Describes the means of obscuring the authentication entry, for all authentication methods specified in FIA_UAU.5.1.	7.1.4 "Identification and Authentication (FIA)"
FIA_UAU_EXT.1	Describes the process for decrypting protected data and keys and that this process requires the user to enter a Password Authentication Factor and, in accordance with FCS_CKM_EXT.3, derives a KEK, which is used to protect the software-based secure key storage and (optionally) DEK(s) for sensitive data, in accordance with FCS_STG_EXT.2.	7.1.2.1 "Overview of Key Management"
FIA_UAU_EXT.2	Describes the actions allowed by unauthorized users in the locked state.	7.1.4 "Identification and Authentication (FIA)"
FIA_X509_EXT.1 FIA_X509_EXT.1/WLAN	Describes where the check of validity of the certificates takes place. Describes the certificate path validation algorithm.	7.1.4.2 "X.509v3 Certificates"
FIA_X509_EXT.2	Describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates. Describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. Describes any distinctions between trusted channels.	
FIA_X509_EXT.2/WLAN	Describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates. Describes any distinctions between trusted channels.	7.1.8.3 "Wireless LAN (WLAN)"
FIA_X509_EXT.3	There are no TSS evaluation activities for this SFR.	
FIA_X509_EXT.6	Describes all certificate stores implemented that contain certificates used to meet the requirements. The description contains information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access.	7.1.4.2 "X.509v3 Certificates"
FMT – Security Management		

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FMT_MOF_EXT.1.1	Describes those management functions that may only be performed by the user and that the TSS does not include an administrator API for any of these management functions.	Table 16 "Management Functions (MDF/VPNC)"
FMT_MOF_EXT.1.2	Describes those management functions that may be performed by the administrator, including how the user is prevented from accessing, performing, or relaxing the function (if applicable), and how applications/APIs are prevented from modifying the Administrator configuration. Describes any functionality that is affected by administrator-configured policy and how.	Table 16 "Management Functions (MDF/VPNC)" 7.1.5.2 "Configuration Profiles"
FMT_POL_EXT.2	Describes how the candidate policies are obtained by the MDM Agent; the processing associated with verifying the digital signature of the policy updates; and the actions that take place for successful (signature was verified) and unsuccessful (signature could not be verified) cases. Identifies the software components that are performing the processing.	7.1.5.2 "Configuration Profiles"
FMT_SMF.1	Describes all management functions, what role(s) can perform each function, and how these functions are (or can be) restricted to the roles identified by FMT_MOF_EXT.1.	7.1.5 "Specification of Management Functions (FMT)" Table 16 "Management Functions (MDF/VPNC)"
	Function 1: Defines the allowable policy options: the range of values for both password length and lifetime, and a description of complexity to include character set and complexity policies.	7.1.4 "Identification and Authentication (FIA)" 7.1.5.2 "Configuration Profiles"
	Function 2: Defines the range of values for both timeout period and number of authentication failures for all supported authentication mechanisms.	
	Function 3: There are no TSS evaluation activities for this SFR.	
	Function 4: Describes each radio and an indication of if the radio can be enabled/disabled along with what role can do so. Describes the frequency ranges at which each radio operates is included in the TSS. Describes the point in the boot sequence the radios are powered on and indicates if the radios are used as part of the initialization of the device.	7.1.5.5 "Radios" Appendix A.1 "Devices Covered by this Evaluation" Table 16 "Management Functions (MDF/VPNC)"
	Function 5: Describes each collection device and an indication if it can be enabled/disabled along with what role can do so.	7.1.5.2 "Configuration Profiles" 7.1.5.6 "Audio and Visual collection devices" Table 16 "Management Functions (MDF/VPNC)"
	Function 6: There are no TSS evaluation activities for this SFR.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
	Function 7: There are no TSS evaluation activities for this SFR.	
	Function 8: Describes the allowable application installation policy options based on the selection included in the ST.	7.1.5.2 "Configuration Profiles"
	Function 9: Describes each category of keys/secrets that can be imported into the TSF's secure key storage.	7.1.2.1 "Overview of Key Management" Table 21 "Summary of keys and persistent secrets in the TOE OS"
	Function 10: See Function 9	
	Function 11: There are no TSS evaluation activities for this SFR.	
	Function 12: Describes each additional category of X.509 certificates and their use within the TSF.	7.1.4.2 "X.509v3 Certificates"
	Function 13: Describes each management function that will be enforced by the enterprise once the device is enrolled.	7.1.5.2 "Configuration Profiles"
	Function 14: Indicates which applications can be removed along with what role can do so	
	Function 15: There are no TSS evaluation activities for this SFR.	
	Function 16: There are no TSS evaluation activities for this SFR.	
	Function 17: There are no TSS evaluation activities for this SFR.	
	Function 18: There are no TSS evaluation activities for this SFR.	
	Function 19: There are no TSS evaluation activities for this SFR.	
	Function 20: There are no TSS evaluation activities for this SFR.	
	Function 21: There are no TSS evaluation activities for this SFR.	
	Function 22: States if the TOE supports a BAF. Describes the procedure to enable/disable the BAF.	7.1.5.3 "Biometric Authentication Factors (BAFs)"
	Function 23: There are no TSS evaluation activities for this SFR.	
	Function 28: There are no TSS evaluation activities for this SFR.	
	Function 30: There are no TSS evaluation activities for this SFR.	
	Function 32: There are no TSS evaluation activities for this SFR.	
	Function 36: There are no TSS evaluation activities for this SFR.	
	Function 37: There are no TSS evaluation activities for this SFR.	
	Function 41: Contains guidance to enable/disable a Hotspot functionality and USB tethering.	7.1.5.5 Radios

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
	Function 44: There are no TSS evaluation activities for this SFR.	
	Function 45: Contains guidance to configure the VPN as Always-On.	7.1.8.4.1 "AlwaysOn VPN"
	Function 47: Describes all assigned security management functions and their intended behavior.	7.1.5.2 "Configuration Profiles" 7.1.5.6 "Audio and Visual collection devices"
FMT_SMF_EXT.1/BT	Describes the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE.	7.1.8.2 "Bluetooth"
	Function BT-1: There are no TSS evaluation activities for this SFR.	
FMT_SMF.1/VPN	Describes the client credentials and how they are used by the TOE.	7.1.5.7 "VPN Certificate Credentials" 7.1.8.4.3 "IKEv2, cryptography and authentication mechanims"
FMT_SMF.1/WLAN	There are no TSS evaluation activities for this SFR.	
FMT_SMF_EXT.2	Describes all available remediation actions, when they are available for use, and any other administrator-configured triggers, and how the remediation actions are provided to the administrator.	7.1.2.1 "Overview of Key Management" 7.1.5.4 "Device Unenrollment"
FMT_SMF_EXT.4	Describes the any assigned functions and that these functions are documented as supported by the platform. Lists any differences between management functions and policies for each supported mobile device. Describes the methods in which the MDM Agent can be enrolled. Makes clear if the MDM Agent supports multiple interfaces for enrollment and configuration.	7.1.5.1 "Device Enrollment" 7.1.5.4 "Device Unenrollment"
FMT_UNR_EXT.1	Describes the mechanism used to prevent users from unenrolling or the remediation actions applied when unenrolled.	7.1.5.4 "Device Unenrollment"
FPT – Protection of the TSF		
FPT_AEX_EXT.1	Describes how the 8 bits are generated and provides a justification as to why those bits are unpredictable.	7.1.6.5 "Domain Isolation"
FPT_AEX_EXT.2	Describes of the memory management unit (MMU), and documents the ability of the MMU to enforce read, write, and execute permissions on all pages of virtual memory.	
FPT_AEX_EXT.3	Describes the stack-based buffer overflow protections implemented in the TSF software which runs in the non-privileged execution mode of the application processor. Contains an inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not. Provides a rationale for those binaries and libraries that are not protected in this manner.	7.1.6.5 "Domain Isolation" 7.1.6.8 "Inventory of TSF Binaries and Libraries"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FPT_AEX_EXT.4	Describes the mechanisms that are in place that prevents non-TSF software from modifying the TSF software or TSF data that governs the behavior of the TSF. Describes how the TSF ensures that the address spaces of applications are kept separate from one another. Describes the method by which actions prescribed by Unstructured Supplementary Service Data (USSD) or Man-Machine Interface (MMI) codes (not supported) are prevented. Documents any TSF data which may be accessed and modified over a wired interface in auxiliary boot modes. Describes data, which is modified in support of update or restore of the device. Describes the means by which unauthorized and undetected modification (that is, excluding cryptographically verified updates per FPT_TUD_EXT.2) of the TSF data over the wired interface in auxiliary boots modes is prevented.	7.1.6.5 "Domain Isolation"
FPT_BDP_EXT.1	Explains how the TOE meets FPT_BDP_EXT.1 at high level description. Contains the following: a. All TSF modules and physical interconnections are within the defined boundary of the SEE and any entities outside the SEE including the main computer operating system can't interfere with transmission between and processing of these modules. b. All plaintext biometric data (whether generated by the biometric capture sensor or by the evaluation processes of the TSF) is retained in volatile memory within the SEE and any entities outside the SEE including the main computer operating system can't access these data. Any TSFIs which may exist, do not reveal plaintext biometric data to any entities outside the SEE. The evaluator shall examine TSFIs of TSF modules provided by the biometric capture sensor (e.g. SDK) because they may include testing or debug codes and the developer who integrated the sensor into the TOE may apply changes to those modules.	7.1.5.3 "Biometric Authentication Factors (BAFs)" 7.1.2.3 "No plaintext key transmission and export"
FPT_JTA_EXT.1	Explains the location of the Joint Test Action Group (JTAG) ports on the TSF, to include the order of the ports (i.e. Data In, Data Out, Clock, etc.). Describes how access to the JTAG is controlled by a signing key. Describes when the JTAG can be accessed, i.e. what has the access to the signing key.	7.1.6.2 "Joint Test Action Group (JTAG) Disablement"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FPT_KST_EXT.1	Contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data. Describes how the cryptographic functions in the cryptographic support (FCS) requirements are being used to perform the encryption functions, including how the KEKs, DEKs, and stored keys are unwrapped, saved, and used by the TOE so as to prevent plaintext from being written to non-volatile storage. Describes, for each power-down scenario how the TOE ensures that all keys in non-volatile storage are not stored in plaintext. Describes how other functions available in the system ensure that no unencrypted key material is present in persistent storage. Describes that key material is not written unencrypted to the persistent storage. For each BAF selected in FIA_UAU.5.1, describes the activities that happen on biometric authentication, relating to the decryption of DEKs, stored keys, and data. In addition, how the system ensures that the biometric keying material is not stored unencrypted in persistent storage.	7 "TOE Summary Specification" 7.1.2.1 "Overview of Key Management" 7.1.1.1 "The Secure Enclave Processor (SEP)"
FPT_KST_EXT.2	Describes the TOE security boundary. Contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data. Describes how other functions available in the system ensure that no unencrypted key material is transmitted outside the security boundary of the TOE. Describes that key material is not transmitted outside the security boundary of the TOE. For each BAF selected in FIA_UAU.5.1 contains a description of the activities that happen on biometric authentication, including how any plaintext material, including critical security parameters and results of biometric algorithms, are protected and accessed. Describes how functions available in the biometric algorithms ensure that no unencrypted plaintext material, including critical security parameters and intermediate results, is transmitted outside the security boundary of the TOE or to other functions or systems that transmit information outside the security boundary of the TOE.	7 "TOE Summary Specification" 7.1.2.1 "Overview of Key Management" 7.1.2.3 "No plaintext key transmission and export" 7.1.1.1 "The Secure Enclave Processor (SEP)"

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FPT_KST_EXT.3	Provides a statement of their policy for handling and protecting keys. Describes a policy in line with not exporting either plaintext DEKs, KEKs, or keys stored in the secure key storage.	7.1.1.1 "The Secure Enclave Processor (SEP)" 7.1.2.3 "No plaintext key transmission and export"
FPT_NOT_EXT.1	Describes critical failures that may occur and the actions to be taken upon these critical failures.	7.1.6.9 "Self-Tests"
FPT_PBT_EXT.1	Explains how the TOE meets FPT_PBT_EXT.1 at high level description. Between the TSS and guidance, identify any TSFI through which the user can access (e.g. revoke) the templates and that those TSFI require the use of a Non-Biometric Authentication Factor (NBAF).	7.1.5.3 "Biometric Authentication Factors (BAFs)"
FPT_STM.1	Lists each security function that makes use of time. Describes how the time is maintained and considered reliable in the context of each of the time related functions. Identifies whether the TSF uses an NTP server or the carrier's network time as the primary time sources.	7.1.6.7 "Time"
FPT_TST_EXT.1	Specifies the self-tests that are performed at start-up. This description must include an outline of the test procedures conducted by the TSF. Includes any error states that they TSF may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation Indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator. The self-tests include algorithm self-tests. The algorithm self-tests will typically be conducted using known answer tests.	7.1.6.9 "Self-Tests"
FPT_TST_EXT.1/VPN	Details the self-tests that are run by the TSF on start-up; this description includes an outline of what the tests are actually doing and makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. Identifies and describes any of the tests that are performed by the TOE platform, describes how the integrity of stored TSF executable code is cryptographically verified when it is loaded for execution. Makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. Describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.	7.1.6.9 "Self-Tests" describes in detail the self-tests that are run by the TSF on start-up

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FPT_TST_EXT.2/ PREKERNEL FPT_TST_EXT.2/ POSTKERNEL FPT_TST_EXT.3	Describes the boot procedures, including a description of the entire bootchain, of the software for the TSF's Application Processor. Describes that before loading the bootloader(s) for the operating system and the kernel, all bootloaders and the kernel software itself is cryptographically verified. For each additional category of executable code verified before execution, describes how that software is cryptographically verified. Contains a justification for the protection of the cryptographic key or hash, preventing it from being modified by unverified or unauthenticated software. Describes the protection afforded to the mechanism performing the cryptographic verification.	7.1.6.1 "Secure Boot"
FPT_TST_EXT.3/WLAN	Details the self-tests that are run by the TSF on start-up. Makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. Describes how to verify the integrity of stored TSF executable code when it is loaded for execution. Makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised, describing the actions that take place for successful and unsuccessful cases.	7.1.6.9 "Self-Tests"
FPT_TUD_EXT.1	There are no TSS evaluation activities for this SFR.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FPT_TUD_EXT.2	Describes all TSF software update mechanisms for updating the system software. Includes a description of the digital signature verification of the software before installation and that installation fails if the verification fails. All software and firmware involved in updating the TSF is described and, if multiple stages and software are indicated, that the software/firmware responsible for each stage is indicated and that the stage(s) which perform signature verification of the update are identified. Describes the method by which the digital signature is verified and that the public key used to verify the signature is either hardware-protected or is validated to chain to a public key in the Trust Anchor Database. If hardware-protection is selected, the method of hardware-protection is described and the justification why the public key may not be modified by unauthorized parties. Describes that software updates to system software running on other processors (i.e., SEP) is verified, the evaluator shall verify that these other processors are listed in the TSS and that the description includes the software update mechanism for these processors, if different than the update.	7.1.6.3 "Secure Software Update"
FPT_TUD_EXT.3	Describes how mobile application software is verified at installation and uses a digital signature.	7.1.6.9.4 "Application integrity"
FPT_TUD_EXT.4	There are no TSS evaluation activities for this SFR.	
FPT_TUD_EXT.5	Describes how mobile application software is verified at installation using a digital signature by a code signing certificate.	7.1.4.2 "X.509v3 Certificates"
FPT_TUD_EXT.6	Describes the mechanism that prevents the TSF from installing software updates that are an older version that the currently installed version.	7.1.6.3 "Secure Software Update"
FTA – TOE Access		
FTA_SSL_EXT.1	Describes the actions performed upon transitioning to the locked state. Describes the information allowed to be displayed to unauthorized users.	7.1.6.6 "Device Locking" 7.1.5.2 "Configuration Profiles" 7.1.2.1 "Overview of Key Management"
FTA_TAB.1	Describes when the banner is displayed.	7.1.7.3 "Lock Screen/Access Banner Display"
FTA_WSE_EXT.1	Specifically defines all of the attributes that can be used to specify acceptable networks (access points).	7.1.7.2 "Restricting Access to Wireless Networks"
FTP – Trusted Channels/Paths		

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	TSS Requirements From Assurance Activities	TSS Section/Reference
FTP_BLT_EXT.1	Describes the use of encryption, the specific Bluetooth protocol(s) it applies to, and whether it is enabled by default. The evaluator shall verify that the TSS includes the protocol used for encryption of the transmitted data and the key generation mechanism used.	7.1.8.2 "Bluetooth"
FTP_BLT_EXT.2	Describes the TSF's behavior if a remote device stops encryption while connected to the TOE.	
FTP_BLT_EXT.3/BR	Specifies the minimum key size for BR/EDR encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.	
FTP_BLT_EXT.3/LE	Specifies the minimum key size for LE encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.	
FTP_ITC.1/WLAN	Describes the details of the TOE connecting to an access point in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. All protocols listed in the TSS are specified and included in the requirements in the ST.	7.1.8.3 "Wireless LAN (WLAN)" 7.1.8.1 "EAP-TLS and TLS"
FTP_ITC_EXT.1	Describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. All protocols listed in the TSS are specified and included in the requirements in the ST. Describes which trusted channel protocol is initiated by the TOE and is used for OTA updates.	Table 30 "Protocols used for trusted channels" 7.1.6.3 "Secure Software Update"
FTP_ITC_EXT.1(2)	The TSS indicates the methods of MDM Agent-Server communication along with how those communications are protected. Describes that all protocols listed in the TSS in support of remote TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.	7.1.4.2 "X.509v3 Certificates" 7.1.4.3 "MDM Server Reference ID"
FTP_TRP.1(2)	Specifies that for mobile device enrollment, the mobile device initiates a connection over HTTPS to the MDM server.	7.1.4.3 "MDM Server Reference ID"

Table 20: SFR to TSS Mappings

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1. TOE Security Functionality

7.1.1. Hardware Protection Functions

7.1.1.1. The Secure Enclave Processor (SEP)

The SEP, which is part of the Secure Enclave, is a coprocessor fabricated in all of the Apple processors listed in Appendix A.1 "Devices Covered by this Evaluation". It utilizes its own secure boot and personalized software update separate from the application processor. It provides all cryptographic operations for Data Protection key management and maintains the integrity of Data Protection even if the kernel has been compromised.

The SEP execution environment shares the main random access memory (RAM) with the application processor. The memory controller provides memory separation between the two processors by distinguishing between the origin of memory fetch or store requests performed by the processors. In addition, the memory controller encrypts/obfuscates (using AES-XEX) all SEP data in RAM using a key shared only between the SEP and the memory controller. This adds an additional level of protection between the SEP memory and the application processor. If the memory separation between the two processors were violated to where the application processor could access the SEP RAM, the application processor would only see encrypted data.

The Secure Enclave includes a hardware random bit generator. Its microkernel is based on the L4 family, a second-generation microkernel generally used to implement UNIX-like operating systems, with modifications by Apple. Communication between the SEP and the application processor is isolated to an interrupt-driven mailbox and shared memory data buffers. Note that only a small, dedicated amount of memory used for communication between the SEP and the main system is shared. The main system has no access to other memory areas of the SEP and no keys or key material may be exported.

Each SEP is provisioned during fabrication with its own 256-bit Unique ID (UID). This UID is used as a key by the device, is not accessible to other parts of the system, and is not known to Apple. When the device starts up, an ephemeral key is created, entangled with its UID, and used to encrypt the SEP's portion of the device's memory space.

Additionally, data that is saved to the file system by the TOE OS is encrypted with a key entangled with the UID and an anti-replay counter. The SEP manages the wrapping and unwrapping of the KEKs associated with the stored data.

The UID also serves as the REK for the whole device.

In addition to the UID, the Group Key (GID) and Apple's root certificate are provisioned during manufacturing. The GID is only unique per device type and is used in the secure software update process. Apple's root certificate is used to verify the integrity and authenticity of software during the secure boot process and for updates of the system software.

The Secure Enclave has its own physical noise source and random bit generator, which is used for generating the 128-bit salt value for the password-based key generation function (PBKDF, specifically PBKDF2). The PBKDF2 salt is regenerated each time the passcode changes. The salt value is stored AES encrypted with the UID in the system keybag. (The PBKDF2 is discussed in Section 7.1.2.2 "Password based key derivation".)

Other salt values used for functions in the TOE OS are generated using the TRNG in the Secure Enclave. This includes nonces used in the generation of digital signature algorithm (DSA) signatures as well as nonces required for the Wi-Fi and TLS protocol.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.2. Cryptographic Support (FCS)

7.1.2.1. Overview of Key Management

Each TOE comes with a unique 256-bit AES key called the UID. This key is stored (i.e., etched into the silicon) in the SEP and is not accessible by the application processor. Even the software in the SEP cannot read the UID. It can only request encryption and decryption operations performed by a dedicated AES engine accessible only from the SEP. The UID is generated during production using the hardware [SP800-90A-Rev1] DRBG (CTR_DRBG(AES)) of the Secure Enclave and then etched into the silicon.

The UID is used to derive two other keys, called "Key 0x89B" and "Key 0x835". These two keys are derived during the first boot by encrypting defined constants with the UID. "Key 0x89B" and "Key 0x835" are used to wrap two other keys: the "Media key" (the file system key, wrapped by "Key 0x89B") and the "Class D key" (the device key, wrapped by "Key 0x835") in accordance with the requirements of [SP800-38F].

Both the "Media key" and the "Class D key" are stored in block 0 of the flash memory, which is also called the "effaceable storage". This area of flash memory can be wiped very quickly. Both the "Media key" and the "Class D key" are generated using the Secure Enclave's TRNG (used to seed the CTR_DRBG) when the TOE OS is first installed or after the device has been wiped.

All keys are generated using an internal entropy source, seeding a deterministic random bit generator (DRBG) (CTR_DRBG). System entropy is generated from timing variations during boot and additionally from interrupt timing once the device has booted. Keys generated inside the SEP use the TRNG to seed the CTR_DRBG.

The Media key is used for the encryption of file system metadata.

The Class D key is used within the key hierarchy to directly wrap the class keys that can be used when the device is locked. All class keys are generated in the Secure Enclave and passed to the TOE OS kernel in wrapped form only. For class keys that can only be used when the device is unlocked, the class keys are wrapped with the XOR of the Class D key and the passcode key.

Every time a file on the data partition is created, a new 256-bit AES key (the "per-file" key) is created using the hardware random bit generator of the Secure Enclave (i.e., FCS_RBG_EXT.1/HW). Files are encrypted using this key with AES in Xor-Encrypt-Xor-based tweaked-codebook mode with ciphertext stealing (XTS) where the initialization vector (IV) is calculated with the block offset into the file, encrypted with the SHA-1 hash of the per-file key, and follows [SP800-38E]. On devices with an A14 and later SoC and M1 and later SoC, the encryption uses AES-256 in XTS mode in the Secure Enclave. On A9 through A13 devices, the encryption uses AES-128 in XTS mode in the Secure Enclave where the 256-bit per file key is split to provide a 128-bit tweak and a 128-bit cipher key. (This is a general statement and not all aforementioned Apple SoCs may be used by the TOE devices.)

Each per-file key is wrapped (in the SEP) with the class key of the file's class and then stored in the metadata of the file. Key wrapping uses AES key wrapping per [RFC3394].

Class keys themselves are wrapped either with device key only (for the class NSFileProtectionNone) or are wrapped with a key derived from the device key and the passcode key using XOR. This key wrapping is also performed within the SEP.

Each file belongs to one of the following classes with its associated class key.

NSFileProtectionComplete

Some files may need to be written while the device is locked. A good example of this is a mail attachment downloading in the background. This behavior is achieved by using asymmetric elliptic curve cryptography (ECDH over Curve25519). The TOE OS implements this by generating a device-wide asymmetric key pair and then protects the private key of this pair by encrypting it with the class key for the NSFileProtectionCompleteUnlessOpen class. Note that this class key can only be unwrapped when the device is unlocked because it requires the passcode to be entered which then is used in the key derivation function (KDF) that generates the key encryption key (KEK) for this class key as described above. The device-wide asymmetric key pair is generated within the SEP.

When receiving data to be protected when the device is in the locked state, the application can create a file with the file attribute NSFileProtectionCompleteUnlessOpen. In this case, the TOE OS generates another asymmetric key pair within

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

the SEP (per file object used to store the data). The device-wide public key and the file object private key are then used to generate a shared secret (using one-pass Diffie-Hellman (DH) as described in [SP800-56A-Rev3]). The KDF is Concatenation Key Derivation Function (Approved Alternative 1) as described in 5.8.1 of [SP800-56A-Rev3]. AlgorithmID is omitted. PartyUInfo and PartyVInfo are the ephemeral and static public keys, respectively. SHA-256 is used as the hashing function. The key generated in that fashion is used as the symmetric key to encrypt the data. The object private key and the shared secret are cleared when the file is closed and only the object public key is stored with the file object.

To read the file, the per file object shared secret is regenerated using the device-wide private key and the per file object public key.

Unwrapping of the device-wide private key can only be performed when the correct passcode has been entered, because the device-wide private key is wrapped with a key that can only be unwrapped with a class key that itself can only be unwrapped when the passcode is available. FDP_DAR_EXT.2.2 requires an asymmetric key scheme to be used to encrypt and store sensitive data received while the TOE is locked. The key scheme implemented by the TOE uses elliptic curve Diffie Hellman (ECDH) over Curve25519. When the correct passcode has been entered, the files with sensitive data received while the device was in the locked state get the per-file key re-wrapped with the NSFileProtectionCompleteUnlessOpenclass key. It is up to the application to check when the device is unlocked and then cause the TOE OS to re-wrap the file encryption key with the class key for the NSFileProtectionComplete class by changing the file's NSFileProtectionKey attribute to NSFileProtectionComplete.

NSFileCompleteUntilFirstUserAuthentication

This class behaves in the same way as Complete Protection, except that the decrypted class key is not removed from memory when the device is locked. The protection in this class has similar properties to desktop full-volume encryption and protects data from attacks that involve a reboot. This is the default class for all third-party app data not otherwise assigned to a Data Protection class.

NSFileProtectionNone

This class key is wrapped only with the device key and is kept in Effaceable Storage. Because all the keys needed to decrypt files in this class are stored on the device, the encryption only affords the benefit of fast remote wipe. If a file is not assigned a Data Protection class, it is still stored in encrypted form (as is all data on a TOE device).

Keychain data is protected using a class structure similar to the one used for files. Those classes have behaviors equivalent to the file Data Protection classes but use distinct keys.

In addition, there are keychain classes with the additional extension "ThisDeviceOnly". Class keys for those classes are wrapped with a key that is also derived from the Device Key, which, when copied from a device during backup and restored on a different device, will make them useless.

The keys for both file and keychain Data Protection classes are collected and managed in keybags. The TOE OS uses the following keybags: system (a.k.a. user and device keybags), backup, escrow, and iCloudBackup. The keys are stored in the system keybag with some keys stored in the escrow keybag. The escrow keybag is used for device update and by MDM, which are both relevant functions defined in [MDF].

The system keybag is where the wrapped class keys used in normal operation of the device are stored. For example, when a passcode or biometric authentication factor is entered, the NSFileProtectionComplete key is loaded from the system keybag and unwrapped. It is a binary plist stored in the No Protection class but whose contents are encrypted with a key held in Effaceable Storage. In order to give forward security to keybags, this key is wiped and regenerated each time a user changes their passcode.

The AppleKeyStore kernel extension manages the system keybag and can be queried regarding a device's lock state. It reports that the device is unlocked only if all the class keys in the system keybag are accessible and have been unwrapped successfully.

Table 21 summarizes the storage for keys in persistent storage.

Key/Persistent Secret	Purpose	Storage (for all devices)
UID	REK for device Key entanglement	SEP

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Key/Persistent Secret	Purpose	Storage (for all devices)
Salt (128 bits)	Additional input to one-way functions	AES encrypted in the system key-bag
Key 0x89B	Wrapping of Media key	RAM
Key 0x835	Wrapping of Class D key	RAM
Class D key	Wrapping the Class keys useable when device is locked	Block 0 of the flash memory (Ef-faceable storage)
Media key	Used for the encryption of file system metadata	Block 0 of the flash memory (Ef-faceable storage)
NSFileProtectionCompleteUn-lessOpen device-wide asymmetric key pair	Writing files while the device is locked	Stored in wrapped form in persis-tent storage
NSFileProtectionCompleteUntil-FirstUserAuthentication	Protection until first user authentica-tion (in memory after)	
NSFileProtectionCompleteUn-lessOpen	Writing files while the device is locked: KDF static public keys	
AfterFirstUnlock	Keychain item that cannot be ac-cessed after a restart until the device has been unlocked once by the user	
AfterFirstUnlockThisDeviceOnly	AfterFirstUnlock on the device that created it: *does not* migrate to a new device.	
WhenUnlocked	Keychain item that can be accessed only while the device is unlocked by the user.	
WhenUnlockedThisDeviceOnly	WhenUnlocked on the device that created it: *does not* migrate to a new device.	
NSFileProtectionNone	Reading and writing files at any time	
NSFileProtectionComplete class key	Reading and writing files prevented while device is locked or booting.	
Individual keys for files and keychains	Used to protect files and keychains.	
Biometric templates (Touch ID and Face ID)	Used to authenticate User using de-vice biometric sensor.	
DH Group parameters	Used as part of IKE/IPsec key estab-lishment	RAM
User IPsec/TLS X.509v3 Certificate Keys	Used to authenticate IKE/IPsec & TLS sessions	Persistently stored encrypted in the platform keychain
CA IPsec/TLS X.509v3 Certificate Public Keys	Used in X.509v3 certificate validation	
IKEv2 IKE_SA Encryption Keys	Used to encrypt IKE/IPsec traffic	RAM
IKEv2 IKE_SA Integrity Keys	Used to verify the integrity of IKE/ IPsec traffic.	
IKEv2 CHILD_SA Encryption Keys	Used to encrypt IKE/IPsec traffic	
IKEv2 CHILD_SA Integrity Keys	Used to verify the integrity of IKE/ IPsec traffic.	
TLS ECDH keys	Used as part of TLS key establishment	
TLS AES session keys	Used to encrypt TLS traffic	

Table 21: Summary of Keys and Persistent Secrets in the TOE OS

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.2.2. Password based Key Derivation

The TOE implements PBKDF2 to derive a 256-bit key from a user's passcode. The PBKDF2 is implemented as specified in [SP800-132] following "Option 2b" defined in section 5.4 of the standard. It uses HMAC-SHA-256 as the pseudorandom function (PRF).

The input to the PBKDF2 is the 128-bit random salt generated by the SEP (as described in Section 7.1.1.1), the user's passcode without any pre-processing, and an iteration count of one. The output is the 256-bit key mentioned above.

Next, the output of the PBKDF2 is repeatedly encrypted with the AES-CBC-256 hardware cipher using the 256-bit UID as the encryption key to generate 256 bits of data with each loop iteration. The loop is performed as often as needed to reach a duration between 100 and 150 milliseconds on that device.

The output after all AES iterations have completed forms the 256-bit root encryption key used to unwrap the user's keybag that holds the class keys for the file system data protection. Only when the unwrapping of the user's keybag is successful is the user considered authenticated.

Note: The number of AES-CBC-256 iterations is calibrated to take at least 100 to 150 milliseconds with a minimum of 50,000 iterations. The number of iterations is device-specific and may be greater than 50,000 on some devices.

7.1.2.3. No Plaintext Key Transmission and Export

The TOE security boundary is the device. The TOE does not transmit or export plaintext key material outside of the TOE security boundary. Plaintext key material is never logged.

Biometric credential data is confined to the Secure Enclave. The Secure Enclave is the Separate Execution Environment (SEE). Biometric keying material, enrollment and authentication templates, the features an algorithm uses to perform biometric authentication for enrollment or verification, threshold values, intermediate calculations, and final match scores never leave the Secure Enclave. The capture sensor and the SEE are isolated from the device's main operating system in runtime.

Plaintext keys such as plaintext data encryption keys (DEKs), key encryption keys (KEKs), and keys stored in the secure key storage are never exported. As described in Section 7.1.2.1, the Secure Enclave preserves the security of these keys.

7.1.2.4. Storage of Persistent Secrets and Private Keys by the MDM Agent

The MDM Agent calls the TOE OS API on the device in order to store keys and persistent secrets in the keychain which are therefore stored in wrapped form in persistent storage, as described above.

Table 22 summarizes the keys, authentication token, and persistent secrets stored for the MDM Agent. They are used on all devices listed in this Security Target.

Key/Persistent Secret	Purpose	Storage (for all devices)
TLS keys	Protecting MDM Protocol communications with the MDM Server	Stored on the device in wrapped form in persistent storage
Device Push Token	The Device Push Token is received when registering with the Apple Push Notification Service (APNS) in order to have an unambiguous identifier in APNS.	The token is not stored on the device but sent to the MDM Server. The MDM Server stores it to be able to contact the device.
UDID	Unique Device ID	Stored in wrapped form in persistent storage
PushMagic	The magic string that must be included in the push notification message. This value is generated by the device.	
Device identity certificate	The device presents its identity certificate for authentication when it connects to the check-in server.	

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Key/Persistent Secret	Purpose	Storage (for all devices)
Certificate Payload	Transferring certificates via payloads.	
Profile encryption key	A profile can be encrypted so that it can only be decrypted using a private key previously installed on a device.	
GUID	Volume Purchase Program (VPP) Account Protection A random UUID should be standard 8-4-4-4-12 formatted UUID string and must be unique for each installation of your product	

Table 22: Summary of Keys and Persistent Secrets used by the MDM Agent

Figure 5 provides an overview on the key management hierarchy implemented in the TOE OS.

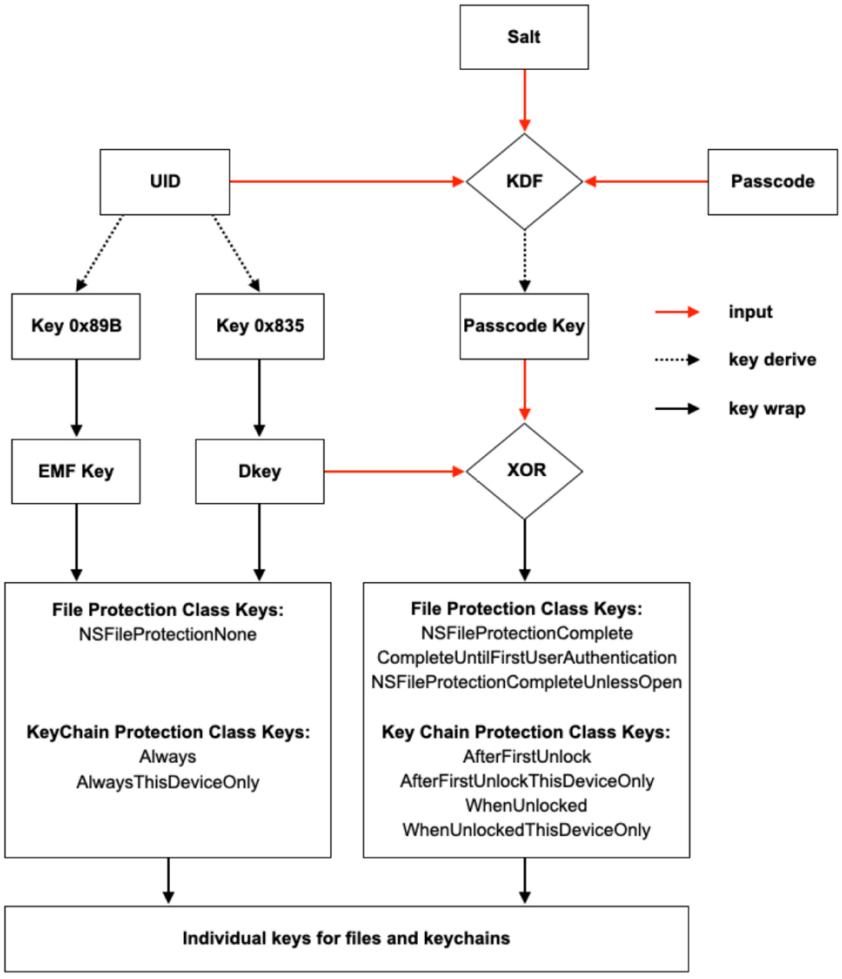

Figure 5: Key Hierarchy in the TOE OS

The Data Protection API can be used by applications to define the class to which a new file belongs by using the

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

NSFileProtectionKey attribute and setting its value to one of the classes described above. When the device is locked, a new file can only be created in the classes NSFileProtectionNone and NSFileProtectionCompleteUnlessOpen.

The UID is burned into the SEP at manufacture. It can not be read by firmware or software and it is used only by the SEP hardware AES engine. The "Key 0x89B" and "Key 0x835" keys are both derived by encrypting defined values (identical for all devices) with the UID. Both keys are held in RAM only. All other keys shown in the figure are stored in wrapped form in persistent storage and unwrapped when needed.

The following bullet points summarize storage of persistent secrets and private keys by the MDM Agent.

- The keys discussed in this section are managed by and/or maintained in the SEP. The TOE OS and SEP interact with each other using a mailbox system detailed in Section 7.1.1.1.
- All file system items and all keychain items are stored in encrypted form only.
- File system metadata is encrypted using the Media key.
- Files and keychain items are encrypted with individual keys. Those keys are wrapped with the class key of the class, the file, or the keychain to which the item belongs.
- Files and keychain items belonging to the classes 'NSFileProtectionNone' (files) or 'kSecAttrAccessibleAlways' (keychain items) are encrypted with keys that are wrapped with the Class D key only. Those items can be accessed (decrypted) before the user is authenticated. For all other classes, the passcode key (which is derived from the user's passcode) is used in the generation of the wrapping key used for those classes, and therefore, decrypting those items is only possible when the user has entered the correct passcode.
- When a wipe command is issued, protected data is wiped by erasing the top-level KEKs. Since all data at rest is encrypted with one of those keys, the device is wiped.

The TOE performs the following activities to protect the keys used for file encryption.

Every time the TOE is booted, it does the following.

- An ephemeral AES key (256-bit) is created in the SEP using the random bit generator of the Secure Enclave.
- The (wrapped) Class D key and (wrapped) Media key (both 256-bit keys) are loaded by the TOE OS kernel from the effaceable storage and sent to the SEP.
- The SEP unwraps the Class D key and the Media key.
- The SEP wraps the Class D key with the newly generated ephemeral key.
- The SEP stores the ephemeral key in the storage controller. This area is not accessible by the TOE OS kernel.

When the TOE OS accesses a file, the following operations are performed.

- The TOE OS kernel first extracts the file metadata (which are encrypted with the Media key) and sends them to the SEP.
- The SEP decrypts the file metadata and sends it back to the TOE OS kernel.
- The TOE OS kernel determines which class key to use and sends the class key (which is wrapped with the Class D key, or with the XOR of the Class D key and the passcode key) and the file key (which is wrapped with the class key) to the SEP.
- The SEP unwraps the file key and re-wraps it with the ephemeral key and sends this wrapped key back to the TOE OS kernel.
- The TOE OS kernel sends the file access request (read or write) together with the wrapped file key to the storage controller.
- The storage controller uses its internal implementation of AES, decrypts the file key, and then decrypts (when the operation is read) or encrypts (when the operation is write) the data during its transfer from/to the flash memory.

The following bullet points summarize the storage location for key material.

- The UID is stored in the firmware of the SEP in a section not accessible by any program in the SEP or the application processor. The SEP can only be used to encrypt and decrypt data (with AES-256) using the UID as the key.
- "Key 0x89B" and "Key 0x835" are held in RAM only.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- The Media key and Class D key are stored in the effaceable area, and the class keys are stored in internal, non-volatile memory. All of them are stored in wrapped form only. As explained, they are never available in plaintext in the application processor system.
- File keys and keychain item keys are stored in internal, non-volatile memory, but in wrapped form only. As explained, they are never available in plaintext in the application processor system.
- The system and the applications can store private keys in keychain items. They are protected by the encryption of the keychain item.
- Symmetric keys used for TLS, HTTPS, or Wi-Fi sessions are held in RAM only. Similarly, ECDH asymmetric keys used for TLS and HTTPS are held in RAM only. They are generated and managed using one of the two libraries, Apple corecrypto Module v18 [Apple silicon, User, Software, SL1] and Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1], or by the AES implementation within the Wi-Fi chip. The functions of those libraries, such as memset(0), also perform the clearing of those keys after use.

7.1.2.5. Randomness Extraction and Expansion Step

"Concatenating the keys and using a KDF (as described in SP800-56C)" is selected in FCS_CKM_EXT.3 Cryptographic Key Generation.

The TOE implements the KDF following the specification in RFC 5869. The KDF defined in this RFC complies with the extraction and expansion KDFs specified in [SP800-56C-Rev2]. This RFC exactly specifies the order of the concatenation of the input data used for the extraction steps as well as the data concatenation and the counter maintenance of the expansion phase.

Extraction:

```
HKDF-Extract(salt, IKM) -> PRK
   Options:
    Hash A hash function; HashLen denotes the length of the
      hash function output in octets.
   Inputs:
    salt Optional salt value (a non-secret random value);
      if not provided, it is set to a string of HashLen zeros.
    IKM Input keying material.
   Output:
    PRK A pseudorandom key (of HashLen octets).
   The output PRK is calculated as follows:
    PRK = HMAC-Hash(salt, IKM)
```

Expansion:

```
HKDF-Expand(PRK, info, L) -> OKM
   Options:
    Hash A hash function; HashLen denotes the length of the
      hash function output in octets.
   Inputs:
    PRK A pseudorandom key of at least HashLen octets
      (usually, the output from the extract step).
    info Optional context and application specific information
      (can be a zero-length string).
    L Length of output keying material in octets
      (<= 255*HashLen).
   Output:
    OKM Output keying material (of L octets).
   The output OKM is calculated as follows:
    N = ceil(L/HashLen)
    T = T(1) | T(2) | T(3) | ... | T(N)
    OKM = first L octets of T
   where:
    T(0) = empty string (zero length)
    T(1) = HMAC-Hash(PRK, T(0) | info | 0x01)
    T(2) = HMAC-Hash(PRK, T(1) | info | 0x02)
    T(3) = HMAC-Hash(PRK, T(2) | info | 0x03)
    ...
```

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

```
(where the constant concatenated to the end of each T(n)
```

The implementation of the KDF uses HMAC-SHA-256 for both the extraction as well as the expansion phase. The salt length and the output key length of the KDF are each 256 bits.

7.1.2.6. Explanation of Usage for Cryptographic Functions

Table 23 enumerates the various cryptographic functions specified in the SFRs and maps them to their modules. The corecrypto modules are abbreviated as: User Space, Kernel Space, and SKS.

SFR	Cryptographic Function	Algorithm	Modes/Notes	Key/Curve Size	Module
FCS_CKM.1	Asymmetric key pair generation	ECDSA KeyGen and KeyVer [FIPS186-5] (ECC scheme)	Notes: Used for TLS, Bluetooth, and memory encryption scheme.	P-256 P-384 P-521 Curve25519	User Space Kernel Space SKS
		Safe-primes [SP800-56A-Rev3] (FFC scheme)	Notes: Used for IPsec.	MODP-2048 MODP-3072 MODP-4096 MODP-6144 MODP-8192	User Space
FCS_CKM.1/VPN	Asymmetric key pair generation	ECDSA KeyGen and KeyVer [FIPS186-5] (ECC scheme)	Notes: Used for IPsec.	P-256 P-384 P-521	User Space
FCS_CKM.2/UNLOCKED	Key establishment	RSA Key Establishment (RSAES-PKCS1-v1_5) [RFC8017]	Notes: Used for TLS.	Modulo: 2048 3072 4096	User Space
		ECC Key Establishment (KAS-ECC-SSC) [SP800-56A-Rev3]	Scheme: ephemeralUnified. Notes: Used for TLS and IPsec.	P-256 P-384 P-521	User Space SKS
		FFC Key Establishment (KAS-FFC-SSC) [SP800-56A-Rev3]	Scheme: dhEphem. Notes: Used for IPsec.	MODP-2048 MODP-3072 MODP-4096 MODP-6144 MODP-8192	User Space
FCS_CKM.2/LOCKED	Key establishment	ECC Scheme [RFC7748]		Curve25519	SKS
FCS_CKM_EXT.3	Key derivation	HKDF [SP800-56C-Rev2]	HMAC-SHA-256	256-bit	User Space

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	Cryptographic Function	Algorithm	Modes/Notes	Key/Curve Size	Module
FCS_COP.1/ENCRYPT	Symmetric encryption/ decryption	AES [FIPS197]	CCM, GCM [SP800-38C] (CCM), [SP800-38D] (GCM)	128-bit 256-bit	User Space Kernel Space SKS
			CBC, XTS [SP800-38A] (CBC), [SP800-38E] (XTS)	128-bit 256-bit	User Space Kernel Space SKS
			KW [SP800-38F] (KW)	128-bit 256-bit	User Space Kernel Space SKS
			CBC [SP800-38A] (CBC)	encrypt only 128-bit 256-bit	SKS Hardware
			CCMP [SP800-38C] (CCM)	128-bit 256-bit	Wi-Fi core/chip
			GCMP [SP800-38D] (GCM)	256-bit	Wi-Fi core/chip
FCS_COP.1/HASH	Hashing	SHS [FIPS180-4]	SHA-1 SHA-256 SHA-384 SHA-512 (byte-oriented mode) Notes: Used for digital signatures (FCS_COP.1/ SIGN), HMACs (FCS_COP.1/ KEYHMAC, and KDFs (FCS_CKM_EXT.3, FCS_COP.1/CONDITION).		User Space Kernel Space SKS
FCS_COP.1/SIGN	Digital signature generation; Digital signature verification	RSA SigGen [FIPS186-5]	Using SHA-256 SHA-384 SHA-512	Modulo: 2048 3072 4096	User Space Kernel Space
		RSA SigVer [FIPS186-5]	Using SHA-1 SHA-256 SHA-384 SHA-512	Modulo: 2048 3072 4096	User Space Kernel Space
		ECDSA SigGen [FIPS186-5]	Using SHA-256 SHA-384 SHA-512	P-256 P-384 P-521	User Space Kernel Space SKS
		ECDSA SigVer [FIPS186-5]	Using SHA-1 SHA-256 SHA-384 SHA-512	P-256 P-384 P-521	User Space Kernel Space SKS

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	Cryptographic Function	Algorithm	Modes/Notes	Key/Curve Size	Module
FCS_COP.1/KEYHMAC	Keyed-hash	HMAC [FIPS198-1]	HMAC-SHA-1 Block size: 512 Output MAC: 160	Greater than or equal to 112 bits	User Space Kernel Space SKS
			HMAC-SHA-256 Block size: 512 Output MAC: 256	Greater than or equal to 112 bits	User Space Kernel Space SKS
			HMAC-SHA-384 Block size: 1024 Output MAC: 384	Greater than or equal to 112 bits	User Space Kernel Space SKS
			HMAC-SHA-512 Block size: 1024 Output MAC: 512	Greater than or equal to 112 bits	User Space Kernel Space SKS
FCS_COP.1/CONDITION	Salted HMAC-SHA and PBKDF2	HMAC-SHA-256	HMAC-SHA-256 Block size: 512 Output MAC: 256	Greater than or equal to 112 bits	User Space Kernel Space SKS
FCS_RBG_EXT.1/HW	Random bit generation; Symmetric key generation	CTR_DRBG(AES) [SP800-90A-Rev1]	AES-256		SKS Hardware
FCS_RBG_EXT.1/SW	Random bit generation; Symmetric key generation	CTR_DRBG(AES) [SP800-90A-Rev1]	AES-256		User Space Kernel Space

Table 23: Explanation of Usage for Cryptographic Functions in the Cryptographic Modules

7.1.3. User Data Protection (FDP)

The Core System Services available for user data protection are those of Protection of Files and Application access to Files, described below in Section 7.1.3.1 and Section 7.1.3.2. These are applicable to all applications on the TOE that are all allowed access to these two System Services.

A further set of high-level system services is presented to applications and monitored by the TOE OS allowing users to grant access to these services, or not.

7.1.3.1. Protection of Files

When a new file is created on a TOE device, it's assigned a class by the app that creates it. Each class uses different policies to determine when the data is accessible. As described above, each class has a dedicated class key that is stored in wrapped form. Note that for the classes other than 'No Protection' to work, the user must have an active passcode lock set for the device.

The basic classes and policies are described below.

Complete Protection (referred to as "class A" in some documents)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Files in this class can only be accessed when the device is unlocked.

Protected Unless Open (referred to as "class B" in some documents)
This class is for files that may need to be written while the device is locked.

Protected Until First User Authentication (referred to as "class C" in some documents)
This class is for files that are protected until the user has successfully authenticated. Unlike the 'Complete Protection' class, the class key for this class is not wiped when the device is locked, but after a re-boot the user has to authenticate before files in this class can be accessed. So, once the user has authenticated after reboot the key is available until the device is shutdown or rebooted.

No Protection (referred to as "class D" in some documents)
Files in this class can always be accessed. The files themselves are encrypted using a file specific key, but this key can be unwrapped without using the passcode key derived from the user's passcode or biometric authentication factor.

Note: Class A, class B, and class C keys require that the user has defined a passcode. If the user has not defined a passcode, then only class D keys exist.

All data in files is considered private data because all files are encrypted. Sensitive data is data protected with a class A or class B key because this data is not accessible when the device is locked.

7.1.3.2. Application Access to Files

An app's interactions with the file system are limited mostly to the directories inside the app's sandbox. During installation of a new app, the TOE OS creates multiple containers for the app. Each container has a specific role. The bundle container holds the app's bundle, whereas the data container holds data for both the application and the user. The data container is further divided into multiple directories that the app can use to sort and organize its data. The app may also store its data in additional containers, such as the "iCloud Drive" and "On My [Device]" containers. The files in these containers are accessible to all apps.

When a built-in application is removed, all of its files, including any related user data and configuration files, are also removed. For any third-party applications, deleting an app (as opposed to "offloading" an application) deletes both the application and all related data from the mobile device.

7.1.3.3. Declaring the Required Device Capabilities of an Application

All applications must declare the device-specific capabilities they need to run. The value of the UIRequiredDeviceCapabilities key is either an array or dictionary that contains additional keys identifying features your app requires (or specifically prohibits). If you specify the value of the key using an array, the presence of a key indicates that the feature is required; the absence of a key indicates that the feature is not required and that the app can run without it. If a dictionary is specified instead, each key in the dictionary must have a Boolean value that indicates whether the feature is required or prohibited. A value of true indicates the feature is required, and a value of false indicates that the feature must not be present on the device.

7.1.3.4. App Groups

Apps and extensions owned by a given developer account can share content when configured to be part of an App Group. It is up to the developer to create the appropriate groups on the Apple Developer Portal and include the desired set of apps and extensions. Once configured to be part of an App Group, apps have access to the following:

- A shared container for storage, which will stay on the device as long as at least one app from the group is installed
- Shared preferences
- Shared keychain items

The Apple Developer Portal guarantees that App Group IDs are unique across the app ecosystem.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The TOE provides the following separate resources for each app group and allows only applications within that group to access the resources:

- Account credential database
- Keystore

7.1.3.5. Restricting Application Access to Services

The TOE allows a user to restrict the services an application can access. The services that can be restricted on a per-app basis are as follows.

Applications prompt the mobile device user to grant permission for the application to use system services when they are installed. Subsequently, mobile device users can perform access control for applications using the following system services through the Settings » Privacy & Security interface:

- Location Services
- Tracking
- Contacts
- Calendars
- Reminders
- Photos
- Bluetooth
- Local Network
- Nearby Interactions
- Microphone
- Speech Recognition
- Camera
- Health Data
- Research Sensor & Usage Data
- HomeKit
- Media & Apple Music
- Files and Folders
- Motion & Fitness
- Focus

7.1.3.6. Keychain Data Protection

Many apps need to handle passwords and other short but sensitive bits of data, such as keys and login tokens. The TOE OS keychain provides a secure way to store these items.

The keychain is implemented as an SQLite database stored on the file system. There is only one database; the securityd daemon determines which keychain items each process or app can access. Keychain access APIs result in calls to the daemon, which queries the app's "keychain-access-groups" and the "application-identifier" entitlement. Rather than limiting access to a single process, access groups allow keychain items to be shared between apps.

Keychain items can only be shared between apps from the same developer. This is managed by requiring third-party apps to use access groups with a prefix allocated to them through the Apple Developer Program or in the TOE OS via application groups. The prefix requirement and application group uniqueness are enforced through code signing, Provisioning Profiles, and the Apple Developer Program. The TOE OS provides a user interface (UI) in the Settings dialog that allows importing of keys for use for Apple-provided applications such as Safari or VPN.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Keychain data is protected using a class structure similar to the one used in file Data Protection. These classes have behaviors equivalent to file Data Protection classes but use distinct keys and are part of APIs that are named differently.

Table 24 shows the keychain classes and their equivalent file system classes.

Availability	File Data Protection Class	Keychain Data Protection Class
When unlocked	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
While locked	NSFileProtectionComplete UnlessOpen	N/A
After first unlock	NSFileProtectionComplete UntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Always	NSFileProtectionNone	kSecAttrAccessibleAlways
Passcode enabled	N/A	kSecAttrAccessibleWhen PasscodeSetThisDeviceOnly

Table 24: Keychain to File-system Mapping

The class kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly behaves the same as kSecAttrAccessibleWhenUnlocked; however, it's available only when the device is configured with a passcode.
Other keychain classes have a "ThisDeviceOnly" counterpart. Keychain items in those classes cannot be moved to a different device using backup and restore; keychain items in those classes are bound to the device.

Among the data stored in keychain items are digital certificates used for setting up VPN connections and certificates and private keys installed by the configuration profile.

Keychains can use access control lists (ACLs) to set policies for accessibility and authentication requirements. Items can establish conditions that require user presence by specifying that they cannot be accessed unless authenticated by entering the device's passcode. ACLs are evaluated inside the SEP and are released to the kernel only if their specified constraints are met.

7.1.3.7. VPN

VPN packet processing is handled by the TOE.

Because all the TSF binaries and libraries are protected from stack-based buffer overflow (see Section 7.1.6.5 "Domain Isolation"), it can be determined that no data will be reused when processing network packets.

Note: To protect the device from vulnerabilities in network processor firmware, network interfaces including Wi-Fi and baseband have limited access to application processor memory. When USB or secure digital input output (SDIO) is used to interface with the network processor, the network processor cannot initiate Direct Memory Access (DMA) transactions to the application processor. When peripheral component interconnect express (PCIe) is used, each network processor is on its own isolated PCIe bus. An input–output memory management unit (IOMMU) on each PCIe bus limits the network processor's DMA access to pages of memory containing its network packets or control structures.

7.1.3.8. Keyed Hash

The TSF performs keyed-hash message authentication in accordance with HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512. It uses key sizes greater than or equal to 112 bits and message digest sizes 160 and 256, 384, 512 bits.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.4. Identification and Authentication (FIA)

The user must authenticate using a passcode or a biometric feature (face or fingerprint) to use the device except when performing the following:

- Accessing Medical ID information
- Answering calls
- Making emergency calls
- Using the cameras (unless their use is generally disallowed)
- Using the flashlight
- Using the control center
- Using the notification center
- Viewing widgets in Today View, and
- Search

All passcode entries are obscured by a dot symbol for each character as the user input occurs. Biometric authentication inputs do not produce feedback to the user unless an input is rejected.

TOE devices support either Face ID (face) or Touch ID (fingerprint) BAFs. Historically, there are multiple generations of these BAFs as improvements were made over time. Appendix A.1 provides the BAF and BAF generation for each TOE device.
For Face ID, when an invalid facial sample is given or cannot be authenticated, the user needs to swipe up before a second attempt can occur and passcode entry will be presented to the user as an option. After five invalid Face ID attempts, the device will vibrate and passcode entry must be used.

For Touch ID, when an invalid fingerprint sample is given or cannot be authenticated, a simple error message is returned to the user to try again. If three invalid fingerprint samples are presented, then the device will offer passcode entry. After five invalid biometric samples are presented, passcode authentication is required.

The following passcode policies can be defined for managed devices:

- The minimum length of the passcode
- The minimum number of special characters a valid passcode must contain
- The maximum number of consecutive failed attempts to enter the passcode (which can be value between 2 and 11, the default is 11)
- The length of minutes for which the device can be idle before it gets locked by the system (which is between 30 seconds and Never, the default is 30 seconds)
- The maximum number of days a passcode can remain unchanged
- The size of the passcode history (the maximum value is 50)

Those parameters for the passcode policy can be defined in the Passcode Payload section of a configuration profile defined by a system administrator for a managed device. For details, see Section 7.1.5.

Devices that support Touch ID do not support Face ID and vice versa. The passcode and the device's BAF **cannot** be combined for two-factor authentication.

In addition, the following behavior applies to BAFs. A passcode must be supplied for additional security validation under any of the following conditions:

- The device has just been turned on or restarted.
- The device has not been unlocked for more than 48 hours.
- The passcode has not been used to unlock the device in the last 156 hours (six and a half days) and Face ID has not unlocked the device in the last 4 hours.
- The device has received a remote lock command.
- After there have been five unsuccessful attempts to match.
- After power off/Emergency SOS/Medical ID has been initiated.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

The number of failed passcode authentication attempts is maintained in a system file, which will persist in the event of graceful or ungraceful loss of power to the TOE. The counter maintaining the number of failed consecutive logon attempts is increased by one immediately once the TOE has identified that the passcode is incorrect. The increment of the counter is completed before the UI informs the user about the failed logon attempt. Face ID and Touch ID use a separate failed attempt counter from the passcode counter that is not maintained over a power loss or reboot (i.e., the Face ID and Touch ID failed attempt counter is reset after a power loss or reboot).

The time between consecutive authentication attempts, including biometric authentication factors, is at least the time it takes the PBKDF2 function to execute. This is calibrated to be at least 80 milliseconds between consecutive attempts. In addition, for Touch ID, the TOE enforces a 5-second delay between repeated failed authentication attempts. When a user exceeds the number of consecutive failed passcode login attempts, the user's partition is erased (by erasing the encryption key). The OS partition is mounted READONLY upon boot and is never modified during the use of the TOE except during a software update or restore. Note that entering the same incorrect passcode multiple times consecutively causes the passcode failed unlock counter to increment only once for those multiple attempts even though these are all passcode failed unlock attempts. Different passcodes must be entered in order for the passcode failed unlock counter to increment. (This description only applies to the passcode failed unlock counter and does not apply to biometric authentication.)

Additionally, authentication credentials, which include biometric samples, are not stored on the TOE in any location. Successful authentication attempts are achieved exclusively by a successful key derivation that decrypts the keybag in the SEP containing the respective class keys.

The use of pre-shared keys, in the form of externally generated bit-based RSA and/or ECDSA keys, to perform peer authentication as part of IKEv2 is exposed in section 7.1.8.4.3 *IKEv2, cryptography and authentication mechanisms*.

7.1.4.1. Biometric Authentication

7.1.4.1.1. Accuracy of Biometric Authentication

Touch ID

Performance assessment of Touch ID verification is conducted in the form of an offline test. Fingerprint data used in these tests were collected separately for each sensor generation using multiple devices, the production version of the sensor, and its firmware. The software (i.e., primarily the biometric algorithms) is based on production code corresponding to the given TOE OS release.

For efficiency reasons, the test is not run on the production hardware, but it is instead emulated on a different platform in a cloud computation infrastructure. A special testing step is performed to assure equality of results between the emulated and production run on the same input data.

The False Acceptance Rate (FAR) test protocols differ between sensor generations as follows:

- For Gen.1 (short for generation 1), a full cross-comparison scheme is used. Each user is enrolled, and each template is attacked by all other user data.
- For Gen.4 (short for generation 4), a partial cross-comparison is done. In this scheme test population is split between users and imposters. Users are enrolled, and each template is attacked by all imposter data.

Face ID

Performance assessment is designed to cover representative Face ID usage cases (e.g., variation of environment, unlock poses and distance, accessory changes) across a representative age, gender and ethnicity distribution.

Data collection for the assessment is submitted to the following controls:

- Check the number of actions in each visit to ensure diversity of actions completed by each participant

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- Image quality assessment
- Score distribution plots from Face ID modules to ensure that scores are in the expected range and data is of high quality.
- Image visualization to remove subjects with identity contamination from evaluation

iPad Pro (3rd Gen) and later allow for facial recognition of a user wearing a mask. This feature is known as the "Face ID with a Mask" setting. For devices that support the "Face ID with a Mask" setting, this setting must be **disabled** in the evaluated configuration.

7.1.4.1.2. Biometric Sample Quality

In the TOE, sample quality is inspected before it is passed to the matcher algorithm for both Touch ID and Face ID. In general, the inspection is based on the following criteria:

For Touch ID:
- Finger motion
- Sensor coverage
- Fixed pattern noise (FPN)

For Face ID:
- **Pose**: Pose angles
- **Distance**: Within a specific range
- **Occlusion**: Visible face region
- **Attention**: Subject must be looking at the device

If the sample quality passes verification during enrollment, the TOE saves it as an enrollment template. When a user authenticates, an authentication template is generated. If a properly formatted template contains unusual data properties, incorrect syntax, low quality, or unrealistic modality, the TOE rejects the template.

The validation of the discussed mechanism is performed regularly for each major TOE OS release. The test is based on specialized datasets containing different levels of coverage and different artifacts. These samples are fed to the biometric system and it is confirmed whether the sample is correctly passed or rejected from the processing as expected.

Additionally, the biometric system is tested by feeding artificially created images containing different geometric patterns.

7.1.4.2. X.509v3 Certificates

There are multiple X.509v3 certificates used by the TOE. First, there is the Apple certificate used to verify the integrity and authenticity of software updates. This certificate is installed in ROM during manufacturing.

Other certificates used for setting up trusted channels or decrypt/verify protected email can be imported by a user (if allowed by the policy) or installed using configuration profiles. In addition, root certificates can be downloaded from a website.

Certificates can be installed for the following:
- IPsec
- TLS
- EAP-TLS, other supported EAP protocols
- Configuration profile validation

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Note that only IPsec, TLS, and EAP-TLS are addressed by [MDF]. Certificates have a certificate type that defines their respective application area. This ensures that only certificates defined for a specific application area are used. In addition, the database containing trust anchors for all certificates is protected via integrity check and write protection.
The certificate types supported by the TOE are as follows:

- AppleX509Basic
- AppleSSL
- AppleSMIME
- AppleEAP
- AppleIPsec
- AppleCodeSigning
- AppleIDValidation
- AppleTimeStamping

External entities can be authenticated using a digital certificate. Out of the box, the TOE includes multiple preinstalled root certificates.

Code signing certificates need to be assigned by Apple and can be imported into a device. The issue of such a certificate can be by app developers or by enterprises that want to deploy apps from their MDM to managed devices. All apps must have a valid signature that can be verified by a code signing certificate before they are installed on a device.

The TOE OS can update certificates wirelessly if any of the preinstalled root certificates become compromised. To disable this, there is a restriction that prevents over-the-air certificate updates.

The list of supported certificate and identity formats are:

- X.509 certificates with RSA and ECDSA keys, and
- File extensions .cer, .crt, .der, .p12, and .pfx.

To use a root certificate that is not preinstalled, such as a self-signed root certificate created by the organization managing the TOE, they can be distributed using one of the following methods:

- When reviewed and accepted by the user
- Using the configuration profile
- Downloaded from a website

When attempting to establish a connection using a peer certificate (i.e., a certificate received from the other endpoint), the peer certificate is first checked to ensure it is valid as per [RFC5280]. Certificates are validated against the Subject Alternative Name (SAN). Wildcards are supported. The Common Name (CN) is ignored. If the SAN does not match the corresponding domain name system (DNS) or IP Address of the server being accessed, validation and, subsequently, the connection will fail. If the certificate is valid, the attempt to establish the connection continues. If the certificate is invalid, the next step is up to the application. The application should provide an indication to the user that the certificate is invalid and options to accept or reject.

As part of the certificate chain validation, the validity period of each certificate in the chain is verified. If the certificate is marked as an extended validation certificate, the TOE performs an OCSP lookup to verify the validity (revocation status) of the certificate (except for WLAN certificate validation, which does not support OCSP). If a connection cannot be established to the OCSP server to determine the revocation status of a certificate, the TOE considers the certificate as revoked. The basicContraints extension and the CA flag are checked. CA certificates must have the basicContraints extension, the CA flag set to TRUE, and include the caSigning purpose. The extendedKeyUsage (EKU) is validated against the rules defined in FIA_X509_EXT.1 (which is a superset of the rules in FIA_X509_EXT.1/WLAN). Finally, the signature of the issuer of the certificate is verified. Only when all checks succeed, the certificate is considered valid and the next certificate in the certificate chain is checked.

The certificate chain searches for the certificates in the TOE's trust store. The trust store is a combination of the trust store delivered with the TOE and the certificates stored in the keychain and marked as trustworthy. The trust store stores

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

and protects certificates from unauthorized deletion and modifications. Authorized administrators can load certificates into the trust store. Certificates from the trusted store are validated using the previously described checks at the time that they are used. Certificate path validation terminates with a certificate in the trust store.

TLS is implemented as a stack that can be utilized by third-party applications. The API informs the calling application that the certificate is not valid. For example, Safari (e.g., HTTPS connection) will display a message to the user that the peer certificate validation failed and allow the user to examine the certificate with the option to allow the connection or not.

The TOE can be configured to disable the user option to accept invalid TLS certificates using the "Allow user to accept untrusted TLS certificates" setting.

7.1.4.3. MDM Server Reference ID

The initial MDM payload contains a mandatory ServerURL string. The URL that the device contacts to retrieve device management instructions must begin with the https:// URL scheme, and may contain a port number (for example ":1234"). Thus, the enrollment is initiated by the device and performed over an HTTPS connection.

When the TOE connects to the MDM server, it uses a TLS authentication, and the MDM Server Reference ID corresponds to the DNS or IP Address as described in Section 7.1.4.2.

Note that during the device enrollment, the MDM enrollment service returns a JavaScript Object Notation (JSON) dictionary with the keys to mobile devices shown in Table 25.

Key	Value
server_name	An identifiable name for the MDM Server
server_uuid	A system-generated server identifier
admin_id	Apple ID of the person who generated the current tokens that are in use
facilitator_id	Legacy equivalent to the admin_id key. This key is deprecated and may not be returned in future responses.
org_name	The organization name
org_email	The organization email address
org_phone	The organization phone
org_address	The organization address

Table 25: MDM Server Reference Identifiers

7.1.5. Specification of Management Functions (FMT)

Since all the mobile devices specified in this Security Target use the same operating system, there are no differences between supported management functions and policies between the different mobile devices.

Table 16, Table 17, and Table 18 describe the management functions of the devices as well as the MDM Agent that are available when the device is enrolled in MDM.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.5.1. Device Enrollment

The methods by which an MDM Agent can be enrolled in to MDM are as follows:[1]

- Using the Apple Business Manager
- Using the Apple Configurator 2
- Distributing an Enrollment Profile via email or a website

A more detailed description is found in Section 7.1.4.3.

The MDM Server provides the agent with an MDM certificate. This certificate is used as the authentication token by the MDM Agent. Similarly, the Apple Configurator 2 installs a certificate when the device allows Apple Configurator 2 to manage it. This certificate is also used as the authentication token.

7.1.5.2. Configuration Profiles

The TOE can be configured using configuration profiles[2] that are installed on the TOE. Configuration Profiles are XML files that may contain settings for multiple configurable parameters.
Different payloads and keys can be defined and configuration profiles are processed by the TOE OS.

When PayloadRemovalDisallowed key is present and set to true in the configuration profile, it prevents the user from deleting the profile unless the profile has a removal password and the user provides it. Only the MDM Server can remove such profiles.
Configuration Profiles can be deployed as follows:

- Using the Apple Configurator 2 tool
- Opening a file on the device:
    - Via an email message
    - Via a webpage
    - Via a files folder
- Using over-the-air configuration
- Using over-the-air configuration via an MDM Server

To preserve the integrity, authenticity, and confidentiality of configuration profiles, they can be required to be digitally signed and encrypted. When the signature of the MDM payload is checked, there are the following possible outcomes:

- Signature verified
- Signature failed
- No signature present or signature cannot be verified due to other reasons (e.g., missing CA certificate)

If the signature is successfully verified, then the payload is deployed.

If the signature fails verification, then the payload is not deployed.

If there is no signature or the signature cannot be verified due to other reasons, then the TOE checks the origin of the payload (i.e., its connection). If the connection to the MDM payload origin is trusted (i.e., the certificates can be validated and its signature checks out), the MDM payload is processed as trusted and deployed. This includes the Apple

[1] Lockdown Mode prevents the installation of configuration profiles and the enrollment in MDM and device supervision.
[2] Lockdown Mode prevents the installation of configuration profiles and the enrollment in MDM and device supervision.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Configurator 2 tool as an MDM origin. (When you enroll the device into the Apple Configurator 2 tool, the device starts trusting the Apple Configurator 2 tool (i.e., it trusts its certificate).) Otherwise, the payload is not deployed.

Managed items relevant for this Security Target that have to be configured using configuration profiles are as follows:

- The password policy—the administrator can define this using the Passcode Payload and:
  - Define the minimum password length
  - Define requirements for the password complexity
  - Define the maximum password lifetime
  - Define the maximum time of inactivity after which the device is locked automatically
  - Define the maximum number of consecutive authentication failures after which the device is wiped
- The VPN policy as follows:
  - Specify that VPN is always on
  - Define the authentication method (certificate)
  - Specification of certificates or shared keys
- The Wi-Fi policy as follows:
  - The EAP types allowed including security type and authentication protocol (WL-1)
  - CAs from which to accept WLAN authentication server certificates (WL-1)
  - Client credentials to be used for authentication (WL-1)
  - The service set identifiers (SSIDs) allowed to connect to (WL-2)
  - The encryption type(s) allowed
  - Enabling/disabling Wi-Fi hotspot functionality (WL-3)
  - Disabling ad hoc wireless client-to-client connections a.k.a. AirDrop (WL-5)
  - Loading X.509 certificates into the TOE (WL-8)
  - Revoking X.509 certificates loaded into the TOE (WL-9)
- General restrictions as follows:
  - Allowing or disallowing specific services (e.g., remote backup) or using devices like the cameras
  - Allowing or disallowing notifications when locked
  - Allowing or disallowing a prompt when an untrusted certificate is presented in a TLS/HTTPS
  - connection
  - Restricting Files USB drive access to encrypted Apple File System (APFS)

The microphones cannot be disabled in general, but a user can restrict access to the microphones on a per-app basis.

Other functions that can be enabled/disabled by an administrator are:

- The installation of applications by a user
- The possibility to perform backups to iCloud
- The ability to submit diagnostics automatically,
- The ability to use fingerprint authentication (Touch ID) or facial authentication (Face ID) for user
- authentication
- The ability to see notifications on the lock screen

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- The ability to take screen shots
- The ability to accept untrusted TLS certificates
- The ability to perform unencrypted backups (via iTunes)

Further restrictions can be enforced for enrolled devices. Those include:

- The ability to modify the account
- The ability to modify the cellular data usage
- The ability to pair with a host other than the supervision host
- The ability for the user to install configuration profiles or certificates interactively
- The ability for the user to use 'Enable Restrictions' interface

A user can access available management functions via the menus of the graphical user interface.

Configuration profiles can also be deployed such that users are unable to override or remove restrictions set in place by Administrators or MDM Administrators. Depending on the behavior defined in the configuration profile, users will be unable to access, perform, or relax management functions defined in Table 16, Table 17, and Table 18. In the most restrictive mode, users will not be able to access the options to alter the above functionality at all. In less restrictive modes, the user is only able to select more secure options.

7.1.5.3. Biometric Authentication Factors (BAFs)

The enrollment and management of biometric authentication factors and credentials are detailed here.

Enrollment for Touch ID is typically accomplished during initial device configuration but can also be performed using the Settings » Touch ID & Passcode menus. Up to five fingerprints may be enrolled, named, and deleted from this menu. Users may place a finger on the Touch ID sensor to determine which biometric credential entry the finger is mapped to. Users may also disable Touch ID selectively for applications or entirely from the Settings » Touch ID & Passcode menu and turning off one or all of the options.

Enrollment for Face ID is typically accomplished during initial device configuration but can also be performed using the Settings » Face ID & Passcode menu by selecting the Set up Face ID option. Users can enroll an alternate appearance for Face ID, for a total of two enrollments. Users may remove Face ID biometric samples from the Settings » Face ID & Passcode and selecting the Reset Face ID option; this action resets both alternate appearances. Users may also turn off Face ID from the Settings » Touch ID & Passcode menu.

When enrolling, naming, and deleting BAFs, the passcode must be successfully entered before changes can be made.

An enrollment authentication template is created from the biometric data and stored inside the SEP. The authentication template cannot be retrieved from the SEP. Instead, the SEP can be asked to compare biometric data to the stored authentication template and will return a success or failure result.

The biometric templates are protected by the passcode. To remove a biometric template or create a new enrollment authentication template, the user must first supply a valid passcode. This can be performed using the Settings » Face ID & Passcode for Face ID and the Settings » Touch ID & Passcode for Touch ID.

7.1.5.4. Device Unenrollment

The TOE supports both preventing device unenrollment from an MDM Server from occurring as well as applying remediation actions when a device is unenrolled.

Unenrollment options for the mobile device user is done through specifying the key "PayloadRemovalDisallowed" (found under Configuration Profiles » Profile-Specific Payload Keys » TopLevel » object TopLevel). This is an optional key. If

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

present and set to true, the user cannot delete the profile unless the profile has a removal password and the user provides it.

It is up to the mobile device administrator to ensure that this key is set appropriately.

In supervised mode, the MDM payload can be locked to the device.

In addition, the additional ability to restrict the installation and removal of configuration profiles from other sources is achieved using the AccessRights key which has a value of a logical OR of the following bit flags:

Required

- 1—Allow inspection of installed configuration profiles
- 2—Allow installation and removal of configuration profiles
- 4—Allow device lock and passcode removal
- 8—Allow device erase
- 16—Allow query of device information (device capacity, serial number)
- 32—Allow query of network information (phone/SIM numbers, MAC addresses)
- 64—Allow inspection of installed provisioning profiles
- 128—Allow installation and removal of provisioning profiles
- 256—Allow inspection of installed applications
- 512—Allow restriction-related queries
- 1024—Allow security-related queries
- 2048—Allow manipulation of settings
- 4096—Allow app management

Note that the AccessRights key may not be zero. If bit 2 is specified, then bit 1 must also be specified. If bit 128 is specified, then bit 64 must also be specified.

When a device is unenrolled, the TOE's remediation action is to delete all MDM payloads regardless of whether they are locked to the device or not. This includes the following:

- Removal of Enterprise applications
- Removal of all device-stored Enterprise resource data

There are no administrator-configurable unenrollment triggers.

7.1.5.5. Radios

The following radios are found in the TOE:

- Bluetooth
- Cellular (Not supported by all devices models. See Appendix A.1)
- Satellite (Not supported by all devices. See Appendix A.1.)
- Ultra-Wideband (UWB) (Not supported by all devices. See Appendix A.1.)
- Wi-Fi

These are more fully described, including the frequencies employed, in Appendix A.1 "Devices Covered by this Evaluation".

As indicated in Table 16, users can enable/disable these radios.

Depending on configuration settings, all radios can be enabled at boot time before the user interacts with the device. Any communication requiring credentials, such as a Wi-Fi passcode, can only commence after the user unlocks the device for the first time. This is due to the storage of the credentials in the keychain with the protection class of requiring the

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

user having initially unlocked the device. Communication requiring no credentials such as unprotected Wi-Fi may commence before the user unlocks the device in case the radio is enabled as per the system configuration. The radios are not required as part of the initialization of the device (i.e., the device will boot with the radios disabled).

Additionally, the user can enable/disable:

- Hotspot functionality: A Personal Hotspot lets you share the cellular data connection of your iPad with another device (Go to Settings > Personal Hotspot, or Settings > Cellular > Personal Hotspot to turn it on or off). This requires that the iPad is equipped with Cellular and that a pre-shared key is authenticated. This connection can be established over WiFi, Bluetooth.
- USB tethering is enabled by connecting a Mac on your iPad using a USB cable. The procedure requires the iPad user to have enabled the hotspot functionality (above) and "Trust This Computer", as no authentication is required when this connection is established. The Mac will then be able to use the cellular data setup on the iPad.

7.1.5.6. Audio and Visual Collection Devices

The following audio and visual collection devices are found in the TOE:

- Cameras
- Microphones

Table 16 describes the roles that can enable/disable them.

7.1.5.7. VPN Certificate Credentials

For the VPN, X.509v3 certificate-based authentication is allowed in the evaluated configuration. These credentials (X.509 certificates) are used by the device when connecting to the IPsec VPN infrastructure.

7.1.5.8. Removal of Applications

The following table indicates which application types can be removed along with the role that can remove them.

Application Type	Role Allowed to Remove the Application
Built-in applications (e.g., Camera, Calendar, Clock, Contacts, Settings, Messages, Safari, Wallet)	Nobody (Some built-in apps allow their icon to be removed, but the app stays installed)
User-installed applications	User
MDM-installed applications	MDM-Administrator, User

Table 26: Removal of Applications

7.1.6. Protection of the TSF (FPT)

7.1.6.1. Secure Boot

Each step of the startup process contains components that are cryptographically signed by Apple to ensure integrity and that proceed only after verifying the chain of trust. This includes the bootloaders, kernel, kernel extensions, and baseband firmware. This secure boot chain helps ensure that the lowest levels of software are not tampered with.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

When a TOE device is turned on, its application processor immediately executes code from read-only memory known as Boot ROM. This immutable code, known as the hardware root of trust, is laid down during chip fabrication and is implicitly trusted. The Boot ROM code contains the Apple Root CA public key, which is used to verify that the iBoot bootloader is signed by Apple before allowing it to load. Because the Apple Root CA public key resides in immutable code, no software (including unverified or unauthorized software) can modify this public key. This is the first step in the chain of trust where each step ensures that the next is signed by Apple. When the iBoot finishes its tasks, it verifies and runs the TOE OS kernel.

A failure of the Boot ROM to load iBoot results in the device entering Device Firmware Update (DFU) mode. In the case of a failure in iBoot to load or verify the next step, startup is halted and the device enters into Recovery mode.

In DFU mode, the TOE device executes the Boot ROM code, which is implicitly trusted immutable code. In Recovery mode, the TOE device executes the iBoot bootloader, which has been cryptographically verified during the aforementioned secure boot procedure.

7.1.6.2. Joint Test Action Group (JTAG) Disablement

The TOE devices use a 2-staged interface that resembles the functionality of JTAG but does not implement the JTAG protocol.

The Apple development environment that is JTAG-like is based on the following:

- To use this JTAG-like interface, a development-fused device is required. In a development-fused device, certain hardware fuses in the device are not blown during the manufacturing process that are blown in a production-fused device. Only with these development interface-related fuses intact, the JTAG-like interface is technically reachable.
- When having a development-fused device, the Apple developers are given a special cable that contains some additional computing logic. This cable establishes a serial channel with the mobile device's JTAG-like interface reachable on development-fused devices. This special cable connects to the development machine's USB port and allows subsequent access by development tools. The serial link allows access to the serial console of the mobile device. The serial console, however, does not allow access on a production-fused device. On a development-fused device, the root account is enabled and an SSH server is listening. The SSH server is accessible via the serial link and allows the developer to access the root account for development including uploading of software or modifying of installed software.

7.1.6.3. Secure Software Update

Software updates to the TOE are released regularly to address emerging security concerns and also provide new features; these updates are provided for all supported devices simultaneously. A request is sent to the mobile device to pull the update from the servers.

Mobile device users receive TOE OS update notifications on the mobile device, through the Finder application on macOS 10.15 or later, through iTunes on macOS 10.14 or earlier and on a PC. Note that the iTunes application is not available on macOS 10.15 or later.

Updates are delivered wirelessly, encouraging rapid adoption of the latest security fixes, as well as downloadable through the aforementioned iTunes and Finder applications.

The startup process, described in Section 7.1.6.1 "Secure Boot" above, helps ensure that only Apple-signed code can be installed on a device. The Apple Root CA public key, which resides in immutable code as described in Section 7.1.6.1, is used in verifying the signed updates.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

To prevent devices from being downgraded to older versions that lack the latest security updates, the TOE OS uses a process called System Software Authorization. If downgrades were possible, an attacker who gains possession of a device could install an older version of OS and exploit a vulnerability that's been fixed in the newer version.

The SEP also utilizes System Software Authorization to ensure the integrity of its software and prevent downgrade installations.

The TOE OS software updates can be installed using Finder on macOS 10.15 or later, using iTunes on macOS 10.14 or earlier and on a PC, or over-the-air (OTA) on the device via HTTPS trusted channel. With iTunes and Finder, a full copy of the latest OS is downloaded and installed. OTA software updates download only the components required to complete an update, improving network efficiency, rather than downloading the entire OS. Additionally, software updates can be cached on a local network server running the caching service on macOS Server so that the TOE devices do not need to access Apple servers to obtain the necessary update data. Software updates may also be cached on macOS version 10.13.0 and higher running the built-in caching service (in the client software).

During a TOE OS upgrade, Finder/iTunes (or the device itself, in the case of OTA software updates) connects to the Apple installation authorization server and sends it a list of cryptographic measurements for each part of the installation bundle to be installed (for example, iBoot, the kernel, and OS image), a random anti-replay value (nonce), and the device's unique Exclusive Chip Identification (ECID).

The authorization server checks the presented list of measurements against versions for which installation is permitted and, if it finds a match, adds the ECID to the measurement and signs the result. The server passes a complete set of signed data to the device as part of the upgrade process. Adding the ECID "personalizes" the authorization for the requesting device. By authorizing and signing only for known measurements, the server ensures that the update takes place exactly as provided by Apple.

The boot-time, chain-of-trust evaluation verifies that the signature comes from Apple and that the measurement of the item loaded, combined with the device's ECID, matches what was covered by the signature.

These steps ensure that the authorization is for a specific device and that an old OS version from one device cannot be copied to another. The nonce prevents an attacker from saving the server's response and using it to tamper with a device or otherwise alter the system software.

Note that this ensures the integrity and authenticity of software updates. A TLS trusted channel is provided for this process.

7.1.6.4. Security Updates

Apple does not disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are generally available. Once an issue has been confirmed and a security update has been made available, references containing technical details are made available and Common Vulnerabilities and Exposures (CVEs) are released.

As part of the Apple release schedule, security updates are provided to the carriers for their own network testing. Depending on the severity of the vulnerability, the security updates are created and provided by Apple to the carriers from a few days up to a week. The carriers' testing can take from 2 weeks up to one month.

The instructions of reporting security issues are available on the web at "Report a security or privacy vulnerability" page (https://support.apple.com/en-us/102549). Users can report security issues related to the TOE on the web at Apple Security Research (https://security.apple.com/). Alternatively, they can send email to "product-security@apple.com", using Apple Product Security PGP key (https://support.apple.com/en-us/102148) to encrypt the email.

Apple publishes security notifications and announcements on the web at "Apple security releases" page (https://support.apple.com/100100). That page provides the information about security updates including the specific CVEs

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

addressed by each update. The security notifications and announcements are also distributed through the security-announce mailing list (https://lists.apple.com/mailman/listinfo/security-announce/).

Apple normally supports their devices with security updates for about 7 years. Products are considered vintage when Apple stopped distributing them for sale more than 5 and less than 7 years ago. Even if the older devices do not get the latest OS release, they will still get latest security updates. Products are considered obsolete when Apple stopped distributing them for sale more than 7 years ago. Users can visit this website (https://support.apple.com/en-us/102772) to find out if their products are supported anymore.

7.1.6.5. Domain Isolation

When a TOE device with cellular capabilities (i.e., a dialer) is in the lock state, only the Emergency Call dialer (Emergency SOS) is available to make a call. The Emergency Call interface does not support the entering of Unstructured Supplementary Service Data (USSD) or Man-Machine Interface (MMI) codes. On TOE devices, USSD/MMI codes start with one or more of the following characters: number sign (#), asterisk (*). The Emergency Call interface ignores (i.e., does not send when the Call button is pressed) codes that start with one or more of these characters, thereby preventing the code actions from executing.

The TOE employs the Boot Progress Register (BPR), which is set of hardware flags in the Secure Enclave, to track if the TOE device has entered auxiliary boot modes, namely, DFU mode and Recovery mode. After a Boot Progress Register flag is set, it cannot be cleared. This allows later software to get a trusted indicator of the state of the system. When the TOE has booted into either DFU mode or Recovery mode, what an unauthenticated user can do is to connect the TOE device to a Mac or PC through USB and reinstall the system. In that case, all contents and settings on the device will be erased.

All applications are executed in their own domain (or 'sandbox'), which isolates the application from other applications and the rest of the system. Stack-based buffer overflow protection is implemented for every sandbox. They are also restricted from accessing files stored by other applications or from making changes to the device configuration. Each application has a unique home directory for its files, which is randomly assigned when the application is installed. If a third-party application needs to access information other than its own, it does so only by using services explicitly provided by the TOE OS.

Stack-based buffer overflow protection implementations in the TOE OS include the following:

- Automatic reference counting (ARC): a memory management system that handles the reference count of objects automatically at compile time
- Address space layout randomization (ASLR): discussed below
- Stack-smashing protection: by utilizing a canary on the stack (Apple recommends that developers compile applications using the -fstack-protector-all compiler flag.)

System files and resources are also shielded from the user's apps. The majority of the TOE OS runs as the non-privileged user "mobile," as do all third-party apps. The entire TOE OS partition is mounted as read-only. Unnecessary tools, such as remote login services, are not included in the system software, and APIs do not allow apps to escalate their own privileges to modify other apps or the TOE OS itself.

Access by third-party apps to user information and features is controlled using declared entitlements. Entitlements are key value pairs that are signed in to an app and allow authentication beyond runtime factors like a UNIX user ID. Since entitlements are digitally signed, they cannot be changed. Entitlements are used extensively by system apps and daemons to perform specific privileged operations that would otherwise require the process to run as root. This greatly reduces the potential for privilege escalation by a compromised system application or daemon.

Address space layout randomization (ASLR) protects against the exploitation of memory corruption bugs. All TSF binaries and libraries use ASLR to ensure that all memory regions are randomized upon launch. Xcode, the TOE OS development environment, automatically compiles third-party programs with ASLR support turned on. Address space layout randomization is used for every sandbox used to execute applications in. There are 8 bits of randomness, taken from the Secure

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Enclave's TRNG, involved in the randomization; the seed for the RNG comes from the seed that also feeds the approved DRBG for cryptographic use.

In addition, the Memory Management Unit (MMU) supports memory address translation using a translation table maintained by the OS kernel. For each page, the MMU maintains flags that allow or deny the read, write, or execution of data. Execution, in this case, allows the CPU to fetch instructions from a given page.

7.1.6.6. Device Locking

The TOE is locked after a configurable time of user inactivity or upon request of the user. When the device is locked, the class key for the 'Complete Protection' class is wiped 10 seconds after locking, making files in that class inaccessible. This also applies for the class key of the 'Accessible when unlocked' keychain class.

The lock screen of a device can be defined and set for supervised devices by an administrator using Apple Configurator 2 or an MDM service.

The TOE can be locked remotely either via the iCloud "Lost Mode" function or by an MDM system if the device is enrolled in management.

7.1.6.7. Time

The following security functional requirements make use of time (FPT_STM.1):

- FAU_GEN.1.2 and similar SFRs (Audit record time stamp)
- FIA_TRT_EXT.1 (Authentication throttling)
- FIA_UAU.7 (Protected authentication feedback)
- FIA_X509_EXT.1.1 (TOE certificate expiration validation)
- FIA_X509_EXT.3.1 (Application certificate expiration validation)
- FMT_SMF.1 Function 1 (Password lifetime)
- FMT_SMF.1 Function 2 (Screen-lock timeout management)
- FTA_SSL_EXT.1 (Lock state inactivity timeout)

When the device starts and the "Set Automatically" setting is set on the device, the following services are used to synchronize the real-time clock on the device.

The devices set time by GPS, unless GPS is unavailable in which case the Apple NTP server will be used.

In the evaluated configuration, the "Set Automatically" setting must be set.

When configured and maintained according to the Network, Identity and Time Zone (NITZ), Global Positioning Satellites (GPS), Network Time Protocol (NTP) standards, or the cellular carrier time service, the time may be considered reliable.

7.1.6.8. Inventory of TSF Binaries and Libraries

The inventory of TSF binaries and libraries is provided in Appendix A.4 "Inventory of TSF Binaries and Libraries".
All user space binaries (applications as well as shared libraries) are subject to address space layout randomization. The logic is implemented in the binary loader and agnostic of a particular file or its contents.

7.1.6.9. Self-Tests

Self-tests are performed by the three cryptographic modules included in the TOE.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

These tests are sufficient to demonstrate that the TSF is operating correctly because they include each of the cryptographic modules included in the TSF. If the self-tests fail, then the TSF will not operate. In addition, the secure boot process begins in hardware and builds a chain of trust through software using the self-tested corecrypto modules, where each step ensures that the next is properly vetted before handing over control to that TSF executable. Secure boot of the devices ensures that the lowest levels of software are not tampered with and that only trusted operating system software from Apple loads at startup. In the devices, security begins in immutable code called the Boot ROM, which is laid down during chip fabrication and known as the hardware root of trust and continues through the loading of the TSF executables.

7.1.6.9.1. Apple corecrypto Module v18 [Apple Silicon, User, Software, SL1]

The module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start-up. The FIPS Self-Tests application runs all required module self-tests. This application is invoked by the TOE OS startup process upon device power on.

The execution of an independent application for invoking the self-tests in the libcorecrypto.dylib library makes use of features of the TOE OS architecture: the module, implemented in libcorecrypto.dylib, is linked by the libcommoncrypto.dylib library, which is linked by the libSystem.dylib library. The libSystem.dylib is a library that must be loaded into every application for operation. The library is stored in the kernel cache and, therefore, is not available in the file system as directly visible files. The TOE OS ensures that there is only one physical instance of the library and maps it to all applications linking to that library. In this way, the module always stays in memory. Therefore, the self-test during start-up is sufficient as it tests the module instance loaded in memory, which is subsequently used by every application on the TOE OS.

All self-tests performed by the module are listed and described in this section.

Power-Up Self-Tests

The following tests are performed each time the Apple corecrypto Module v18 [Apple silicon, User, Software, SL1] starts and must be completed successfully for the module to operate in the approved mode. If any of the following tests fail, then the device powers itself off. To rerun the self-tests on demand, the user must reboot the device. If the following tests succeed, then the device continues with its normal power-up process.

Cryptographic Algorithm Tests

Algorithm	Modes	Test
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, GCM, XTS	KAT Separate encryption/decryption operations are performed
DRBG (CTR_DRBG)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
RSA	SigGen, SigVer	KAT Separate signature generation/verification operations are performed
	Encrypt, Decrypt	KAT Separate encryption/ decryption operations are performed
ECDSA	SigGen, SigVer	KAT Separate signature generation/verification operations are performed
Diffie-Hellman "Z" computation	N/A	KAT
EC Diffie-Hellman "Z" computation	N/A	KAT

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Table 27: Cryptographic Algorithm Tests

Software/Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple corecrypto Module v18 [Apple silicon, User, Software, SL1]. The module's HMAC-SHA-256 is used as a FIPS-approved algorithm for the integrity test. If the test fails, then the device powers itself off. If the test succeeds, then the device continues with its normal power- up process.

Critical Function Tests

No other critical function test is performed on power up.

Conditional Tests

The following items describe the conditional tests supported by the Apple corecrypto Module v18 [Apple silicon, User, Software, SL1]:

- Pair-wise Consistency Test

  The module does generate asymmetric keys and performs all required pair-wise consistency tests, the encryption/decryption, as well as signature verification tests with the newly generated key pairs.

- SP800-90A Assurance Tests

  The module performs a subset of the assurance tests as specified in section 11 of [SP800-90A-Rev1]; in particular, it complies with the mandatory documentation requirements and performs known-answer tests and prediction resistance.

- Critical Function Test

  No other critical function test is performed conditionally.

7.1.6.9.2. Apple corecrypto Module v18 [Apple Silicon, Kernel, Software, SL1]

The module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. The FIPS Self-Tests functionality runs all required module self-tests. This functionality is invoked by the TOE OS Kernel startup process upon device initialization. If the self-tests succeed, then the module instance is maintained in the memory of the TOE OS Kernel on the device and made available to each calling kernel service without reloading. All self-tests performed by the module are listed and described in this section.

Power-Up Self-Tests

The following tests are performed each time the Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1] starts and must be completed successfully for the module to operate in the approved mode. If any of the following tests fail, then the device shuts down automatically. To run the self-tests on demand, the user may reboot the device. If the followings tests succeed, then the device continues with its normal power-up process.

Cryptographic Algorithm Tests

Algorithm	Modes	Test
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, XTS	KAT Separate encryption/decryption operations are performed
DRBG (CTR_DRBG)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
ECDSA	SigGen, SigVer	KAT Separate signature generation/verification operations are performed

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Algorithm	Modes	Test
RSA	SigGen, SigVer	KAT Separate signature generation/verification operations are performed

Table 28: Cryptographic Algorithm Tests

Software/Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1]. The module's HMAC-SHA-256 is used as a FIPS-approved algorithm for the integrity test. If the test fails, then the device powers itself off. If the test succeeds, then the device continues with its normal power-up process.

Critical Function Tests

No other critical function test is performed on power up.

Conditional Tests

The following items describe the conditional tests supported by the Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1]:

- Continuous Random Number Generator Test

  The module performs a continuous random number generator test, whenever CTR_DRBG is invoked. In addition, the seed source implemented in the TOE OS kernel also performs a continuous self-test.

- Pair-wise Consistency Test

  The module generates asymmetric ECDSA key pairs and performs all required pair-wise consistency tests (signature generation and verification) with the newly generated key pairs.

- SP800-90A Assurance Tests

  The module performs a subset of the assurance tests as specified in section 11 of [SP800-90A-Rev1]; in particular, it complies with the mandatory documentation requirements and performs known-answer tests and prediction resistance.

- Critical Function Test

  No other critical function test is performed conditionally.

7.1.6.9.3. Apple corecrypto Module v18 [Apple Silicon, Secure Key Store, Hardware, SL1]

[FIPS140-3] requires that the module perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start-up. In addition, the noise source feeding the random bit generator requires continuous verification. The module runs all required module self-tests pertaining to the firmware. This self-test is invoked automatically when starting the module. In addition, during startup of the hardware, the hardware DRBG invokes its independent self-test.

The occurrence of a self-test error in either the firmware or the hardware DRBG triggers an immediate shutdown of the device preventing any operation.

All self-tests performed by the module are listed and described in this section.

Power-Up Tests

The following tests are performed each time the TOE OS starts and must be completed successfully for the module to operate in the approved mode. If any of the following tests fail, then the device powers itself off. To rerun the self-tests

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

on demand, the user must reboot the device. If the followings tests succeed, then the device continues with its normal power-up process.

Cryptographic Algorithm Tests

Algorithm	Modes	Test
AES Implementation selected by the module for the corresponding environment AES-128	ECB, CBC	KAT[3] Separate encryption/decryption operations are performed
AES SKG Hardware Accelerator Implementation AES-256	ECB	KAT Separate encryption/decryption operations are performed
AES SKG Hardware Accelerator Implementation AES-256	CBC	KAT Encryption operation is performed
Hardware DRBG (CTR_DRBG)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT[4]
ECDSA	SigGen, SigVer	KAT Separate signature generation/verification operations are performed
EC Diffie-Hellman "Z" computation	N/A	KAT
Entropy Source	N/A	RCT APT

Table 29: Cryptographic Algorithm Tests

Software/Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple corecrypto Module v18 [Apple silicon, Secure Key Store, Hardware, SL2]. The module's HMAC-SHA-256 is used as a FIPS-approved algorithm for the integrity test. If the test fails, then the device powers itself off. If the test succeeds, then the device continues with its normal power-up process.

Critical Function Tests

No other critical function test is performed on power up.

Conditional Tests

The following items describe the conditional tests supported by the Apple corecrypto Module v18 [Apple silicon, Secure Key Store, Hardware, SL2]:

- Pair-wise Consistency Test
  The module generates ECDSA key pairs and performs all required pair-wise consistency tests (signature generation and verification) with the newly generated key pairs.
- Critical Function Test

  No other critical function test is performed conditionally.

[3] Self-test is subject to the "selector" approach for the different implementations of AES.
[4] Self-test is subject to the "selector" approach for the different implementations of SHA.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.6.9.4. Application Integrity

Apple issues certificates to the TOE OS application developers that developers use to sign their applications. The TOE OS checks each application's signature to ensure that it was signed using a valid Apple-issued certificate by using a hardware-protected asymmetric key. This applies to both application installation and application execution.

7.1.7. TOE Access (FTA)

7.1.7.1. Session Locking

The TOE devices can be configured to transit to a locked state after a configurable time interval of inactivity. This time can be defined by an administrator using a configuration profile.

Displaying notifications when in the locked state can be prohibited via the allowLockScreenNotificationsView key in the Restrictions Payload of a configuration profile.

7.1.7.2. Restricting Access to Wireless Networks

Users and administrators can restrict the wireless networks to which a TOE device connects. Using the configuration profile, administrators can define the wireless networks to which the device is allowed to connect using SSIDs and the EAP types allowed for authentication. This also includes the following attributes:

- Specification of the CA(s) from which the TSF will accept WLAN authentication server certificates(s)
- Specification of the CA(s) from which the TSF will accept WLAN authentication server certificates(s)
- Security type
- Authentication protocol
- Client credentials to be used for authentication
- Client credentials to be used for authentication

For the list of radios supported by each device, see Appendix A.1 "Devices Covered by this Evaluation". The standards listed there define the frequency ranges.

7.1.7.3. Lock Screen/Access Banner Display

An advisory warning message regarding unauthorized use of the TOE can be defined using an image that is presented during the lock screen. Configuration for this is described in FMT_SMF.1 Function 36.

Because the banner is an image, there are no character limitations. Information is restricted to what can be included in an image appropriate to the device display.

7.1.8. Trusted Path/Channels (FTP)

The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of the protocols listed in Table 30.

Protocol	ST Requirements	Used for
802.11ax	In addition to the minimum trusted channel requirements and supported by FCS_COP.1	Wireless access points
802.11ac-2013		

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Protocol	ST Requirements	Used for
802.11-2012	FTP_ITC_EXT.1 FTP_ITC.1/WLAN	
802.1X EAP-TLS	FTP_ITC_EXT.1 FTP_ITC.1/WLAN	WLAN
TLS 1.1	FCS_TLSC_EXT.1/WLAN	Secure data transfer
TLS 1.2	FCS_TLSC_EXT.1 FCS_TLSC_EXT.1/WLAN FDP_UPC_EXT.1/APPS	
IPsec	FCS_IPSEC_EXT.1 FTP_ITC_EXT.1	VPN
Bluetooth (v4.2, v5.0, v5.2, v5.3) with BR/EDR and LE	FDP_UPC_EXT.1/BLUETOOTH	Trusted IT products
HTTPS	FCS_HTTPS_EXT.1 FDP_UPC_EXT.1/APPS FTP_ITC_EXT.1 FTP_ITC_EXT.1(2) FTP_TRP.1(2)	OTA updates; secure communication over a network

Table 30: Protocols used for Trusted Channels

IPsec supports authentication using shared keys or certificate-based authentication. The TOE's TLS client supports certificate-based mutual authentication.

7.1.8.1. EAP-TLS and TLS

For Wi-Fi, the TOE supports EAP-TLS using TLS v1.1 and v1.2 and supports the following cipher suites:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

EAP-TLS can be configured as one of the EAP types accepted using the AcceptEAPTypes key in the Wi-Fi payload of the configuration profile.

When configuring the TOE to utilize EAP-TLS as part of a Wi-Fi Protected Access 2 (WPA2) or Wi-Fi Protected Access 3 (WPA3) protected Wi-Fi-network, the CA certificate(s) to which the server's certificate must chain can be configured using the PayloadCertificateAnchorUUID key in the Wi-Fi payload of the configuration profile.

Using the PayloadCertificateAnchorUUID and TLSTrustedServerNames keys in the Wi-Fi payload of the configuration profile, the administrator can enforce that untrusted certificates are not accepted and the authentication fails if such an untrusted certificate is presented.

The TOE also provides mobile applications TLS version 1.2 (client) capabilities via an API service that includes support for the following cipher suites from FCS_TLSC_EXT.1:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
- TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246
- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

In addition to these cipher suites that have been tested as part of the evaluation, other cipher suites are supported by the TOE. These are listed at
https://developer.apple.com/documentation/security/1550981-ssl_cipher_suite_values?language=objc

Furthermore, the elliptic curve cipher suites above may utilize the following supported elliptic curve extensions by default:

- secp256r1 (P-256)
- secp384r1 (P-384)
- secp521r1 (P-521)

The TOE supports the following cryptographic key establishment schemes in TLS:

- For cipher suites with RSA key exchange: RSAES-PKCS1-v1_5 RSA-based key establishment with 2048- bit, 3072-bit, and 4096-bit keys as specified in Section 7.2 of [RFC8017].
- For cipher suites with ECDH key exchange: Elliptic curve-based key establishment with NIST curves P-256, P-384, and P-521 as specified in [SP800-56A-Rev3], "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography".

When an application uses the provided APIs to attempt to establish a trusted channel, the TOE will compare the Subject Alternative Name (SAN) contained within the peer certificate (specifically the SAN fields, IP Address, and Wildcard certificate if applicable) to the Fully Qualified Domain Name (FQDN) of the requested server. The Common Name (CN) is ignored. If the FQDN in the certificate does not match the expected SAN for the peer, then the application cannot establish the connection.

Applications can request from the TOE the list of cipher suites supported and then define which of the supported cipher suites they enable for the TLS-protected session they are going to set up. Once the connection has been set up, the application can retrieve the ciphersuite negotiated with the communication partner.

When setting up a TLS session, the library function for the handshake (SSLHandshake) will indicate, via a result code, any error that occurred during the certificate chain validation.

Communication between the MDM Agent and the MDM Server is protected by HTTPS (which employs TLS) using the above supported cipher specifications.

Certificate pinning is supported. The user of the TLS framework can use this certificate pinning support. However, existing TLS clients in the TOE, such as the Safari browser, do not support certificate pinning. WLAN also does not support certificate pinning.

7.1.8.1.1. TLS Mutual Authentication

For TLS mutual authentication, each application can have one or more client certificates. For X.509 certificates, the TOE allows applications to store client certificates in either a P12 file or in a keychain. A P12 file only holds one client certificate, but a keychain can hold multiple client certificates.

For applications that use a P12 file, there is only one client certificate choice when performing TLS mutual authentication. The application passes this choice to the TLS API.

For applications that use a keychain, there may be multiple client certificate choices. An application can select the appropriate client certificate from the keychain and pass the selected certificate to the TLS API or it can pass an array of client certificates to the TLS API. The TLS API uses the first certificate in the array and ignores the other certificates in the array when establishing a connection.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

There are no other factors beyond configuration necessary to engage in mutual authentication.

7.1.8.1.2. TLS Client Renegotiation

The TOE supports TLS client secure renegotiation through the use of the “renegotiation_info” TLS extension in accordance with [RFC5746].

7.1.8.2. Bluetooth

The TOE supports Bluetooth (v4.2, v5.0, v5.2, v5.3) including Basic Rate/Enhanced Data Rate (BR/EDR) and Low Energy (LE) with the following Bluetooth profiles:

- Hands-Free Profile (HFP 1.8)
- Phone Book Access Profile (PBAP 1.2)
- Advanced Audio Distribution Profile (A2DP 1.4)
- Audio/Video Remote Control Profile (AVRCP 1.6)
- Personal Area Network (PAN) Profile
- Human Interface Device (HID) Profile
- Message Access Profile (MAP 1.4)
- Wireless iPhone Accessory Protocol (WiAP)
- Braille

By default, Bluetooth is enabled. The supported Bluetooth version depends on the device. For the mapping of devices to Bluetooth versions, see Appendix A.1 "Devices Covered by this Evaluation".

User permission is required to pair the TOE device with a remote Bluetooth device. Manual authorization is implicitly configured since pairing can only occur when explicitly authorized through the Settings » Bluetooth interface of the TOE device. During the pairing time, the TOE user needs to manually enter a PIN, confirm that a provided PIN matches on both devices, or affirm the remote device name depending on the remote Bluetooth

The TOE automatically authorizes the remote Bluetooth device during pairing for all Bluetooth profiles the remote device announces to support during the pairing operation. This approach avoids user confusion between a paired device to which the TOE is connected and authorized and thus can communicate with and a device to which the TOE is connected but not yet authorized with which the TOE cannot yet communicate. To de-authorize a device, the user would unpair the device. The TOE establishes a "trusted relationship" with an authorized device at the time of pairing. The only difference in behavior between a trusted device and an untrusted device is that the untrusted device must first be manually authorized as described in the previous paragraph.

The TOE requires that remote Bluetooth devices support an encrypted connection. Devices that want to pair with the TOE via Bluetooth are required by the TOE OS to use Secure Simple Pairing, which uses ECDH-based authentication and key exchange with curve P-256. See the Bluetooth Specifications [BT_SPEC] for details.

The TOE generates a new ephemeral ECDH key pair for every new connection attempt. The ECDH keys are generated in user space using the corecrypto user space module. No data can be transferred via Bluetooth until pairing has been completed. The TOE terminates the connection if the remote device stops encryption while connected to the TOE.

The only time the device is Bluetooth discoverable is when the Bluetooth configuration panel is active and in the foreground. There is no toggle switch for discoverable or not discoverable. Unless the configuration panel is the active panel, the device is not discoverable.

Connections via BR/EDR and LE are secured using 128-bit AES Counter with CBC-MAC (AES-CCM-128) mode. The Wi-Fi chip performs the bulk Bluetooth AES-CCM-128 cryptography. No other key sizes are supported; thus, smaller key sizes cannot be negotiated. A local database is kept of all Bluetooth device addresses for paired devices, which is checked prior to any automatic connection attempt.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Bluetooth devices may not establish more than one connection. Multiple connection attempts (i.e., pairing and session initialization attempts) from the same BD_ADDR for an established connection will be discarded. For details of the security of Bluetooth/LE, see the Bluetooth Specifications [BT_SPEC].

The TOE supports the Logical Link Control and Adaptation Layer Protocol (L2CAP) through an API in the IOBluetoothDevice class.

An RFCOMM channel object can be obtained by opening an RFCOMM channel in a device or by requesting a notification when a channel is created (this is commonly used to provide services). See the IOBluetooth RFCOMMChannel class.

A service ID is used to identify the local service.

7.1.8.3. Wireless LAN (WLAN)

The TOE implements the wireless LAN protocol as defined in IEEE 802.11 (2012). The TOE uses the random number generators of the corecrypto cryptographic modules for the generation of keys and other random values used as part of this protocol.

As required by IEEE 802.11 (2012), the TOE implements the CTR with CBC-MAC protocol (CCMP) with AES (128- bit key) as defined in section 11.4.3 of 802.11. This protocol is mandatory for IEEE 802.11 (2012) and is also the default protocol for providing confidentiality and integrity for wireless LANs that comply with IEEE 802.11. Newer device models support AES-CCMP-256 with 256-bit keys following IEEE 802.11ac-2013 as well as AES-GCMP-256 with 256-bit key sizes, respectively, following IEEE 802.11ac-2013. The implementation of these AES algorithms is performed by the bulk encryption operation of the Wi-Fi chip.

AES key wrapping as defined in [SP800-38F] is used to wrap the Group Temporal Key (GTK), which is sent in an Extensible Authentication Protocol (EAPOL) key frame in message three of the 4-way handshake defined in section 11.6.2 of IEEE 802.11 (2012).

AES key unwrapping used to unwrap the GTK is performed as described in [SP800-38F] section 6.1, Algorithm 2: W-1(C), and in section 6.2, Algorithm 4: KW-AD(C).

Additionally, PRF-384 is implemented as defined in IEEE 802.11-2012, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", section 11.6.1.2. It is implemented in the TOE as part of the WPA implementation and is used for the key generation of AES keys when the Counter Mode CBC-MAC Protocol

(CCMP) cipher (defined in section 11.4.3.1 of IEEE 802.11-2012) is used. PRF-704 is implemented as defined by IEEE 802.11ac-2013 for TOE device models supporting GCMP.

The random bit generator used is the one provided by the main device (i.e., FCS_RBG_EXT.1/SW).

The Wi-Fi Alliance certificates for the devices in this evaluation are in Appendix A.2 "Wi-Fi Alliance Certificates".

7.1.8.4. VPN

The TOE VPN configuration covers a range of aspects addressed throughout this section:

- AlwaysOn VPN – VPN configuration allowing for a full control of the organization over device traffic (See AlwaysOn VPN section)
  - cellular and WiFi - Additional configurations are supported by the TOE
  - Captive Networking - Exceptions are supported
- IPSec – Protocol allowing the TOE to natively support the VPN architecture with strong cryptography (see IPsec General section)
  - Protocol and VPN Configuration
  - Security Policy Database (SPD)
  - IKEv2, cryptography and authentication mechanisms

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

7.1.8.4.1. AlwaysOn VPN

For managed and supervised devices, the TOE must be configured with an 'AlwaysOn' VPN where the organization has full control over device traffic by tunneling all IP traffic back to the organization using an Internet Key Exchange (IKE) v2 based IPsec tunnel (See IKE section for more details on its configuration).

Configuration key values dedicated to the VPN type 'AlwaysOn':

- VPN interfaces (cellular and/or Wi-Fi) for which the VPN is 'AlwaysOn' (default = cellular and Wi-Fi)
- VPN service exceptions (AirPrint, cellular services, voicemail)
- Captive networking exceptions (if any)

7.1.8.4.2. IPsec VPN Client

IPsec VPN connection protects all traffic (“always on” configuration) entering and leaving the TOE platform interfaces, and is implemented in the TOE natively, as part of the TOE OS:

- There is no separate "client" application;
- The VPN tunnels are configured and controlled by Network Extension Framework, which is a part of the Core OS Layer described in Section 1.5.2 "TOE Architecture".
- The TOE implements the IPsec protocol as specified in [RFC4301] . Configuration of VPN connection settings is performed by the IPsec VPN client administrator.
- The packets are processed by the TOE in little-endian order.

IPsec Configuration:

- DISCARD – “not allow
- BYPASS – “send plaintext”
  - The TOE allows a limited number of services to be configured to DISCARD or BYPASS
  - Example: AirPrint, cellular services, voicemail, and applications that make use of Captive Networking Identifiers.
- PROTECT – “sent through the IPsec tunnel” under specific SPD rules

IPsec service rules are configured in the Security Policy Database (SPD):

- Select "Allow traffic via tunnel" to match a PROTECT rule.
- "Drop Traffic" will cause that traffic to match a DISCARD rule.
- "Allow traffic outside tunnel" will create a BYPASS rule for that service.

The SPD is implemented by the TOE, which, as a managed device, is configured using a configuration profile either manually, through the Apple Configurator 2, or via an MDM solution. See Section 7.1.5.2 "Configuration Profiles" for more information.

The VPN tunnels, according to the service configuration, do:

- Protect all data, other than that described in Section 7.1.8.4.1.
  - Configure the object VPN in the Profile Specific Payload so that packet / VPN payload is processed against SPD
  - Includes IPsec Dictionary Keys, IKEv2 Dictionary Keys, DNS Dictionary Keys, Proxies Dictionary Keys, and AlwaysOn Dictionary Keys.
- Ignore (discard) any other plaintext data that is received automatically (no need to configure an explicit discard.

There are no differences in the routing of IP traffic when using any of the supported baseband protocols.

Evaluation Notes:

- In the evaluated configuration, a catch-all value must be set.
- The TOE also provides an API for third-party VPN clients.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- The TOE supports enabling/disabling the VPN protection (see FMT_SMF.1 Function 3 enable/disable the VPN protection).

The TOE OS is responsible for each type of VPN auditable event.

7.1.8.4.3. IKEv2, cryptography and authentication mechanisms

The TOE (and product) only support IKEv2 in tunnel mode (as defined in RFCs 7296 and 4307), configured as follows:

- The IP address or hostname of the VPN server
- The identifier of the IKEv2 client in one of the supported formats
- The authentication method (shared key or certificate)
- The server certificate (for certificate-based authentication)
- If extended authentication is enabled
- The encryption algorithm
- The integrity algorithm

Supporting cryptographic mechanisms are provided to the VPN services by one of the following cryptographic modules:

(1) User Space CM: Apple corecrypto Module v18 [Apple silicon, User, Software, SL1], or

(2) Kernel Space CM: Apple corecrypto Module v18 [Apple silicon, Kernel, Software, SL1]

IKEv2 is supported by the following cryptography, in the following algorithm configurations:

- Symmetric algorithms for IKE and ESP encryption (from either user or kernel space)
    - AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256.
    - Bulk encryption (from kernel space)
- Integrity mechanisms (from either user or kernel space)
    - HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC- SHA-512.
- Key Exchange (Diffie-Hellman Groups) (from either user or kernel space)
    - DH Group 14 (2048-bit MODP), DH Group 15 (3072-bit MODP), DH Group 16 (4096-bit MODP), DH Group 17 (6144-bit MODP), DH Group 18 (8192-bit MODP), DH Group 19 (256-bit Random ECP), DH Group 20 (384-bit Random ECP), and DH Group 21 (521-bit ECP)
    - Key generation and key establishment (from user space)
    - secret value 'x' and nonces are generated by the TOE's software DRBG (FCS_RBG_EXT.1/SW)
        - The possible lengths of 'x' and the nonces are 224, 256, or 384 bits.
- Authentication
    - RSA or ECDSA X.509v3 digital certificate and pre-shared keys (conformance to RFC 8784)
    - In RFC 8784, the pre-shared key here is referred to as a PPK or Post-quantum Preshared Key, and is shared between the initiator and responder and is mixed into the IKE authentication exchange
- Entropy
    - 384 bits of entropy seed the private copy of the CTR_DRBG of the VPN Client

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

    - corecrypto ECDH or DH APIs are used to generate keys using the CTR_DRBG entropy (follows the key establishment algorithms defined in [SP800-56A-Rev3], ECDSA key generation algorithm in [FIPS186-5], Diffie-Hellman Group (MODP) key generation [RFC 3526], and IKEv2 key derivation function in [RFC5996])

As part of the peer authentication process, a comparison is made of the Subject Alternative Name (SAN) contained within the peer certificate to the to the Fully Qualified Domain Name (FQDN) of the requested server. The Common Name (CN) is ignored. If the FQDN in the certificate does not match the expected SAN for the peer, then the session will not be established.

- SAN mismatch or absence result in the authentication process failure.
- If the SAN matches the peer's identifier, the authentication process is successful

Additional IKE configurations:

- The TOE supports configurable time-based lifetimes for both IKEv2 Phase 1 and Phase 2 SAs.
    - Phase 1 SAs are configurable to 24 hours
    - Phase 2 SAs are configurable to 8 hours.
    - Settings are applied to the TOE via .xml profiles generated via an MDM, an Apple-specific tool such as "Apple Configurator 2," or by manually editing the .xml file directly.
- The TOE supports configuration of the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv2/ IKE_SA connection and the strength of the symmetric algorithm negotiated to protect the IKEv2 CHILD_SA connection
    - Strength is configured using .xml configuration files: the administrator must choose the cryptographic algorithms (including key strength) used for the IKE_SA.
    - Key strength must be one of 128 or 256 bits as specified in the IKEv2 Dictionary Keys, EncryptionAlgorithm Key.

In the evaluated configuration, during negotiation, the TOE will only negotiate the configured algorithms, which must be identical to IKEv2/IKE_SA and IKEv2 CHILD_SA. This configuration is specified in [CCGUIDE].

7.1.8.4.4. Residual Information Protection and Packet Processing

When a network packet is received, the TCP/IP stack allocates a new buffer in memory of the same size as the incoming packet then copies the packet into the new buffer, thereby overwriting the entire allocated buffer. The packet is only referred to by reference/address, not copied. The VPN encrypts and decrypts the packets in place because the size of the data does not change when using symmetric ciphers.

7.1.9. Security Audit (FAU)

7.1.9.1. Audit Records

The TOE logging capabilities can collect a wide array of information concerning TOE usage and configuration. The available commands and responses constitute audit records and must be configured by TOE administrators using profiles that are further explained in Section 7.1.5.2 "Configuration Profiles". These profiles must also be used to determine the audit storage capacity as well as default action when capacity is reached.

Although the specific audit record format is determined via configuration profile, the following attributes form the baseline:

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- Date and time the audit record was generated
- Process ID or information about the cause of the event
- Information about the intended operation
- Success or failure (where appropriate)

Audit record information is inaccessible on TOE devices and is only accessible externally on trusted workstations via the Apple Configurator 2 or to an MDM Server on enrolled devices.

Depending on the underlying OS of the trusted workstation or MDM Server, the audit records are transferred to the following locations:

- macOS
  - ~/Library/Logs/CrashReporter/MobileDevice/[Your_Device_Name]/
- Windows
  - C:\Users\[Your_User_Name]\AppData\Roaming\AppleComputer\Logs\Crash\MobileDevice\[Your_Device_Name]\

The following table lists the auditable events for the required SFRs and provide audit records format:

(MDF) SFR specified in the ST	Auditable events	Additional audit record contents	Example of audit records
FAU_GEN.1	Start-up and shutdown of the audit functions	-	**mdmd(libdyld.dylib)[6307]** <Notice> **mdmd(libdyld.dylib)[6314]** <Notice>
	All administrative actions	-	**dmd[3038]** <Notice>
	Start-up and shutdown of the OS	-	**SpringBoard(SpringBoard)[57]** <Notice> **SpringBoard(FrontBoard)[57]** <Notice> **SpringBoard(RunningBoardServices)[57]** <Notice> **CommCenter(IMFoundation)[80]** <Notice> **rapportd(IMFoundation)[76]** <Notice> **contextstored(CoreDuet)[56]** <Notice>
	Insertion or removal of removable media	-	**default 02:40:02.471883-0700** livefileproviderd ReallyMountVolume: **default 02:40:53.494457-0700** livefileproviderd Unmounting
FCS_STG_EXT.1	Import or key destruction	Identity of key. Role and identity of requestor.	**49506633**: AKS unwrap_media_key_from_class succeeded for tag = 7 **49506633**: aks_migrate_SEPUUID2b_to_classM_key() succeeded for AES, container = /dev/disk0s1 **49506633**: AKS unwrap_media_key_from_class succeeded for tag = 7 **apfs_meta_crypto_state_unwrap:980**: got key for volume 2F29025D-A75E-40CE-9EFE-61A6B8848880 **apfs_device_locked:3837**: apfs Data is now UN-locked! (flags 0x40)
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.	**apfs_unwrap_key:1263**: AKS unwrap_key failed, error = e00002e2
FDP_DAR_EXT.1	Failure to encrypt/decrypt data.	-	**securityd[96]** <Notice>
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	-	**securityd[96]** <Notice>
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database	Subject name of certificate	**mc_mobile_tunnel(MDM)[4225]** <Notice> **iPad profiled[97]** <Notice> **profiled[97]** <Notice>
FIA_X509_EXT.1	Failure to validate x.509v3 certificate	Reason for failure of validation	**default 14:49:33.115596+0200** nsurlsessiond boringssl_session_handshake_incomplete(191)
FPT_TST_EXT.1	Initiation of self-test	-	**SEP: SEP: FIPS POST** begin **SEP: FIPSPOST_L4** fipspost_post:109: PASSED: (2 ms) - fipspost_post_integrity **SEP: sks: FIPS POST** Succeeded

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

	Failure of self-test	-	**fipspost_post** **fipspost_post_integrity** -POST_FAILURE: 0xFFFFFFFF
FPT_TST_EXT.2/ PREKERNEL	Start-up of TOE	-	Darwin Kernel Version 19.0.0: Wed Oct 9 22:37:47 PDT 2019; root:xnu_development-6153.42.1~1/DEVELOP-MENT_ARM64_T8010 iBoot version: iBoot-5540.40.51
(WLAN)	Auditable events	Additional audit record contents	Example of audit records
FCS_TLSC_EXT.1 /WLAN	Failure to establish an EAP-TLS session.	Reason for failure. Non-TOE device endpoint connec-tion.	**wifid(WiFiPolicy)[45]** <Notice> **wifid(WiFiPolicy)[45]** <Error>
	Establishment/ termination of an EAP-TLS session.	Reason for failure. Non-TOE device endpoint connec-tion.	**wifid(WiFiPolicy)[45]** <Notice> **wifid(WiFiPolicy)[45]** <Error>
FIA_X509_EXT.1/ WLAN	Failure to validate x.509v3 cer-tificate	Reason for failure of validation	**default 14:49:33.115596+0200** nsurlsessiond bor-ingssl_session_handshake_incomplete(191)
FPT_TST_EXT.3/ WLAN	Execution of this set of TSF self-test.	-	**corecrypto_kext_start called: tracing enabled** FIPSPOST_KEXT fipspost_post:<XXX: PASSED: (0 ms) - fipspost_post_XXX ... FIPSPOST_KEXT fipspost_post:136: all tests PASSED (129 ms)
FTA_WSE_EXT.1	All attempts to connect to ac-cess points.	For each access point record the Complete SSID and MAC of the MAC Address. Success and failures (in-cluding reason for failure).	SSID and MAC address records: **default 17:47:01.890053-0800 wifid** [corewifi] **wifid(WiFiPolicy)[45]** <Notice> **wifid(WiFiPolicy)[45]** <Error>
FTP_ITC.1/WLAN	All attempts to establish a trusted channel.	Identification of the non-TOE de-vice endpoint of the channel.	**wifid(WiFiPolicy)[45]** <Notice>
(AGENT)	Auditable events	Additional audit record contents	Example of audit records
FAU_ALT_EXT.2	Success/failure of sending alert.	-	Success: **<code> default profiled** Beginning profile installation **<code> default profiled** Profile "XXX" installed **<code> default mdmd** Accepted new connection. MDM mdmd **<code> default profiled** Sending MDM settings changed notification. Failure: **<code> error profiled** Cannot Check Out. Error: NSError: Desc : The Internet connection appears to be offline. Domain : NSURLErrorDomain Code : -1009 Type : DMCFatalError MDMClientLibrary com.apple.de-vicemanagementclient
FAU_GEN.1(2)	Start-up and shutdown of the MDM Agent	-	**mdmd(libdyld.dylib)[6307]** <Notice> **mdmd(libdyld.dylib)[6314]** <Notice>
	MDM policy updated.	-	**profiled[97]** <Notice>
	Any modification commanded by the MDM Server	-	**default 15:05:35.071156+0200 mdmd** Received push noti-fication.
FAU_SEL.1(2)	All modifications to the audit configuration that occur while the audit collection functions are operating.	-	**profiled[97]** <Notice>
			mc_mobile_tunnel(MDM)[4225] <Notice>
			mc_mobile_tunnel(MDM)[4225] <Notice> **profiled[97]** <Notice>
FCS_TLSC_EXT.1	Failure to establish a TLS ses-sion	Reason for failure.	**wifid(WiFiPolicy)[45]** <Notice> **wifid(WiFiPolicy)[45]** <Error>

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

	Failure to verify presented identifier	Presented identifier and reference identifier.	**wifid(WiFiPolicy)[45]** <Notice> **wifid(WiFiPolicy)[45]** <Error>
	Establishment/termination of a TLS session.	Non-TOE device endpoint of connection.	**nsurlsessiond(CFNetwork)[162]** <Notice>
FIA_ENR_EXT.2	Enrollment in management	Reference identifier of MDM Server.	**default 14:09:39.308624+0200 profiled** Checking for MDM installation... **default 14:09:39.312514+0200 profiled** ...finished checking for MDM installation. **default 14:09:39.318516+0200 profiled** Beginning profile installation... **default 14:09:39.318710+0200 profiled** Beginning profile installation for com.apple.config.osxserver.atsec.com.mdm **default 14:09:39.321386+0200 profiled** Profile "com.apple.config.osxserver.atsec.com.mdm" is replacing an existing profile having the same identifier. **default 14:09:39.346118+0200 profiled** Refreshing MDM details. **Default 14:09:39.346309+0200 profiled** No MDM installation found.
FMT_POL_EXT.2	Failure of policy validation.	Reason for failure of validation.	**error 17:13:20.765096+0200 wifid** {ASSOC-} Failed to join(-369033199 - 0xEA010011): test **default 15:19:57.113029+0200 wifid** {AUTOJOIN, ASSOC*} Failed to associate with test, reason -369033199
FMT_SMF_EXT.4	Outcome (Success/failure) of function.	-	**default 15:05:35.071156+0200 mdmd** Received push notification.
FMT_UNR_EXT.1	Attempt to unenroll	-	**Default 19:43:43.048171-0800 Preferences** _didHideAlertController: <UIAlertController: 0x10a025600> **Default 19:43:43.096961-0800 profiled** Removing profile
FTP_ITC_EXT.1(2)	Initiation and termination of trusted channel.	Trusted channel protocol. Non-TOE device endpoint of connection.	**wifid(WiFiPolicy)[45]** <Notice>
(BT)	Auditable events	Additional audit record contents	Example of audit records
FAU_GEN.1.1/BT	Start-up and shutdown of the audit functions	-	**mdmd(libdyld.dylib)[6307]** <Notice> **mdmd(libdyld.dylib)[6314]** <Notice>
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user-rejected connection, incorrect pin entry).	**default 08:55:22.471374-0600 bluetoothd** Session <...> Decision: **default 08:56:35.373472-0600 bluetoothd** Rejecting SSP request for device 744DD52B

	Failed user authorization for local Bluetooth Service.	Bluetooth address and name of device. Bluetooth profile. Identity of local service with service ID.	**default 08:53:41.365190-0600 bluetoothd** Device found: CBDevice 9D6E9A89-6C27-273D-1295-B73EC2FDF868, BDA 7C:7A:91:E3:B1:00, Nm 'Test-Lab-System1', PID 0x0246 (?), VID 0x1D6B, VS 2, DsF 0x800000 < Pairing >, DvF 0x54000 < ClsP HIDG UsrC >, DvT LaptopComputer, RSSI -57, Color 0, FV '5.3.12', MicM Auto, Plcm M Enabled, srMd Disabled, CF 0x80000000000 < Attr >
			Local device ID: default 08:50:11.358071-0600 cloudpaird Successfully sending message {
			MessageType = CloudPairing;
			"Version 1" = {
			DeviceName = "iPhone-A14_updated"; EncryptionType = Basic; MessageType = PairingRequest; PublicAddress = "8C:EC:7B:06:16:D5"; RequestedKeyLength = 16; RequestedKeyType = (EncryptionKeys, IdentityKeys);};
			"Version 2" = {
			DeviceName = "iPhone-A14_updated"; EncryptionType = ECDH; MessageType = InitiatorPairingKeys; PublicAddress = "8C:EC:7B:06:16:D5"; RequestedKeyLength = 16; RequestedKeyType = (PublicKeys, IdentityKeys); RequestedKeys = { CloudNonce = {length = 16, bytes = 0xfd0dab7cea305af21fb7d8b7f86fe1a2}; CloudPublicKey

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

			= {length = 64, bytes = 0x27b6adf0 8b2b90ab cb344a1b 07b43042 ... f879146b 2e6b2adf }; IRK = {length 16, bytes = 0xcd318d659b7febadf7f50c3623e86ac4};}; TimeStamp = 751850567329;}; "Version 3" = { DeviceName = "iPhone-A14_updated"; EncryptionType = ECDH; MessageType = InitiatorPairingKeys; PublicAddress = "8C:EC:7B:06:16:D5"; RequestedKeyLength = 16; RequestedKeyType = (PublicKeys, IdentityKeys); RequestedKeys = { CloudNonce = {length = 16, bytes = 0xfd0dab7cea305af21fb7d8b7f86fe1a2}; CloudPublicKey = {length = 64, bytes = 0x27b6adf0 8b2b90ab cb344a1b 07b43042 ... f879146b 2e6b2adf }; IRK = {length = 16, bytes = 0xcd318d659b7febadf7f50c3623e86ac4};}; TimeStamp = 751850567329;};}
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Bluetooth address and name of device.	**default 08:54:05.557719-0600 bluetoothd** Session "com.apple.Preferences-MBF-419-83-unique-idcom.apple.Preferences-419" is asking to connect device "Test-Lab-System1" **default 08:53:41.365190-0600 bluetoothd** Device found: CBDevice 9D6E9A89-6C27-273D-1295-B73EC2FDF868, BDA 7C:7A:91:E3:B1:00, Nm 'Test-Lab-System1', PID 0x0246 (?), VID 0x1D6B, VS 2, DsF 0x800000 < Pairing >, DvF 0x54000 < ClsP HIDG UsrC >, DvT LaptopComputer, RSSI -57, Color 0, FV '5.3.12', MicM Auto, Plcm M Enabled, srMd Disabled, CF 0x80000000000 < Attr >
	Failure of Bluetooth connection.	Reason for failure.	**error 08:55:22.467318-0600 bluetoothd** Connection to device 744DD52B failed - result was 705
(VPN)	Auditable events	Additional audit record contents	Example of audit records
FAU_GEN.1.1/VPN	Start-up and shutdown of the audit functions	-	**mdmd(libdyld.dylib)[6307]** <Notice>: mdmd starting... **mdmd(libdyld.dylib)[6314]** <Notice>: mdmd preparing to stop.
	All administrative actions	-	**dmd[3038]** <Notice>
FCS_IPSEC_EXT.1	Decision to DISCARD or BYPASS network packets processed by the TOE.	Presumed identity of source subject. The entry in the SPD that applied to the decision.	In the evaluated configuration of Always On, there are no DISCARD or BYPASS audit records generated.
FCS_IPSEC_EXT.1	Failure to establish an IPsec SA.	Identity of destination subject. Reason for failure.	**<code> Default** 0x0 237 0 nesessionmanager: [com.apple.networkextension:] NESMAlwaysOnSession[TEST:5D3D3176-21A7-4C16-AC83-E4AE997C4189]: status changed to connecting **<code> Debug** 0x0 26107 0 NEIKEv2Provider: (NetworkExtension) [com.apple.networkextension:] Creating UDP NAT-T transport 10.15.15.184:4500(4500) to 10.181.181.17:4500 on "pdp_ip0" **<code>Info** 0x0 26107 0 NEIKEv2Provider: (NetworkExtension) [com.apple.networkextension:] Removing client [NEIKEv2TransportClient DAA761525C2FEA4F IKEv2Session[1, DAA761525C2FEA4F-4415253061787F4B]] for <NEIKEv2Transport> UDP NAT-T 10.15.15.184:4500 -> 10.181.181.17:4500
FCS_IPSEC_EXT.1	Establishment/Termination of an IPsec SA.	Identity of destination subject. Transport layer protocol, if applicable. Source subject service identifier, if applicable. Non-TOE endpoint of connection (IP address) for both successes and failures.	Establishment: **<code> Default** 0x0 237 0 nesessionmanager: [com.apple.networkextension:] NESMAlwaysOnSession[TEST:28CF9D41-9EE0-4D6F-B820-388C6D88C6C9]: status changed to connecting **<code> Debug** 0x0 20358 0 NEIKEv2Provider: (NetworkExtension) [com.apple.networkextension:] Creating UDP NAT-T transport 10.15.15.184:4500(4500) to 10.181.181.17:4500 on "pdp_ip0" Termination: **<code> Default** 0x0 237 0 nesessionmanager: [com.apple.networkextension:] NESMIKEv2VPNSession[Child:Primary Tunnel:TEST:28CF9D41-9EE0-4D6F-

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

			B820-388C6D88C6C9:pdp_ip0]: status changed to disconnecting **<code> Debug** 0x0 20358 0 NEIKEv2Provider: (NetworkExtension) [com.apple.networkextension:] IKEv2Session[1, 580F2FB62A42B04D-1D490D1FA899BAAD] Sending request of length 72 with ID 2 on <NEIKEv2Transport> UDP NAT-T 10.15.15.184:4500 -> 10.181.181.17:4500 **<code> Info** 0x0 20358 0 NEIKEv2Provider: (NetworkExtension) [com.apple.networkextension:] Removing client [NEIKEv2TransportClient 580F2FB62A42B04D IKEv2Session[1, 580F2FB62A42B04D-1D490D1FA899BAAD]] for <NEIKEv2Transport> UDP NAT-T 10.15.15.184:4500 -> 10.181.181.17:4500
FMT_SMF.1/VPN	Success or failure of management function. Specify VPN gateways to use for connections.	-	The gateways are defined in a Configuration Profile. Success: **profiled(NetworkExtension)[486]** <Notice> **nesessionmanager[214]** <Notice> Failure: **NEIKEv2Provider(NetworkExtension)[14270]** <Notice>

7.1.9.2. MDM Agent Alerts

The MDM Agent generates and sends an alert in response to an MDM Server request (i.e., applying a policy, receiving a reachability event). The Status key field in Table 31 is used as the alert message to satisfy the FAU_ALT_EXT.2 requirements. The MDM Agent's response is being used as the alert transfer mechanism.

When a configuration profile is sent to an MDM Agent, the MDM Agent responds using an "Alert", the MDM Result Payload, a plist-encoded dictionary containing the following keys, as well as other keys returned by each command.

Key	Type	Content
Status	String	Status. Legal values are described as: **Acknowledged** Everything went well. **Error** An error has occurred. See the ErrorChain for details. **CommandFormatError** A protocol error has occurred. The command may be malformed. **Idle** The device is idle (there is no status). **NotNow** The device received the command but cannot perform it at this time. It will poll the server again in the future.
UDID	String	Unique Device ID (UDID) of the device
CommandUDID	String	Universally Unique ID (UUID) of the command that this response is for (if any)
ErrorChain	Array	Optional—Array of dictionaries representing the chain of errors that occurred

Table 31: MDM Agent Status Commands

During installation:

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- The user or administrator tells the device to install an MDM payload (specifying the object MDM in Configuration Profiles » Profile-Specific Payload Keys for Managed Devices)
- The device connects to the check-in server. The device presents its identity certificate for authentication, along with its UDID and push notification topic.

Note: Although UDIDs are used by MDM, the use of UDIDs is deprecated for TOE OS apps.

If the server accepts the device, the device provides its push notification device token (Device Push Token) to the server. The server should use this token to send push messages to the device. This check-in message also contains a PushMagic string. The server must remember this string and include it in any push messages it sends to the device.

During normal operation:

- The server (at some point in the future) sends out a push notification to the device.
- The device polls the server for a command in response to the push notification.
- The device performs the command.
- The MDM Agent contacts the server to report the result of the last command and to request the next command.

From time to time, the Device Push Token may change. When a change is detected, the device automatically checks in with the MDM Server to report its new push notification token (Device Push Token).

Note: The device polls only in response to a push notification; it does not poll the server immediately after installation. The server must send a push notification to the device to begin a transaction.

The MDM Agent initiates communication with the MDM Server in response to a push notification by establishing an HTTPS connection to the MDM Server URL. The MDM Agent validates the server's certificate and then uses the identity specified in its MDM payload as the client authentication certificate for the connection.

When an MDM Server wants to communicate with a TOE device, a silent notification is sent to the TOE's MDM Agent via the Apple Push Notification (APN) service, prompting it to check in with the server. The process of notifying the MDM Agent does not send any proprietary information to or from the APNS. The only task performed by the push notification is to wake the device so it checks in with the MDM Server.

7.1.9.2.1. Queuing of Alerts

In cases where the HTTPS channel is unavailable, for example because the device is out of range of a suitable network, an alert regarding the successful installation of policies is queued until the device is able to communicate with the server again. The queue cannot become long because if the device is out of communication with the MDM Server, no additional requests can be received. If the MDM Server does not receive the alert, the MDM Server should re-initiate the transfer until a response is received from the device.

There are certain times when the device is not able to do what the MDM Server requests. For example, databases cannot be modified while the device is locked with Data Protection. When a device cannot perform a command due to these types of situations, it will send the NotNow status without performing the command. The server may send another command immediately after receiving this status but chances are the following command will also be refused.

After sending a NotNow status, the device will poll the server at some future time. The device will continue to poll the server until a successful transaction is completed.

The device does not cache the command that was refused. If the server wants the device to retry the command, it must send the same command again later, when the device polls the server.

The server does not need to send another push notification in response to this status. However, the server may send another push notification to the device to have it poll the server immediately.

The following commands are guaranteed to execute on the TOE OS and never return NotNow:

- DeviceInformation

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

- ProfileList
- DeviceLock
- EraseDevice
- ClearPasscode
- CertificateList
- ProvisioningProfileList
- InstalledApplicationList
- Restrictions

7.1.9.2.2. Alerts on Successful Application of Policies

Candidate policies are generated by the administrator and disseminated as a configuration profile using one of the methods already described in Section 7.1.5.2 "Configuration Profiles".

The protocol for managing configuration profiles between the MDM Server and the MDM Agent is specified in object MDM in the Profile-Specific Payload Keys » Managed Devices.

When the application of policies to a mobile device is successful, the MDM Agent replies with an MDM Result Payload with Status value "Acknowledged".

If a policy update is not successfully installed, then the MDM Agent replies with an MDM Result Payload with Status value "Error" or CommandFormatError, or "NotNow".

7.1.9.2.3. Alerts on Receiving Periodic Reachability Events

Periodic reachability events are initiated by the MDM Server using Push Notifications. When a periodic reachability event is received, the MDM Agent contacts the server in the manner described in Section 7.1.9.2 "MDM Agent Alerts," above.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

8. Abbreviations, Terminology, References

8.1. Abbreviations

Abbreviation	Meaning
A2DP	Advanced Audio Distribution Profile
ABM	Apple Business Manager
ACL	Access Control List
AES	Advanced Encryption Standard
APFS	Apple File System
API	Application Programming Interface
APN	Apple Push Notification
APNS	Apple Push Notification Service
ARC	Advanced Reference Counting
ARM	Advanced RISC Machine
ASLR	Address Space Layout Randomization
AVRCP	Audio/Video Remote Control Profile
BAF	Biometric Authentication Factor
BMD	Biometric Management Design
BR/EDR	Basic Rate/Enhanced Data Rate
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-MAC
CCMP	Counter with CBC-MAC Protocol
CDMA	Code Division Multiple Access
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CN	Common Name
CTR	Counter
CVE	Common Vulnerabilities and Exposures
DAR	Data at Rest
DC-HSDPA	Dual-Carrier High Speed Downlink Packet Access
DEK	Data Encryption Key
DES	Data Encryption Standard
DFU	Device Firmware Upgrade
DH	Diffie-Hellman
DMA	Direct Memory Access
DN	Distinguished Name
DNS	Domain Name Server
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DNS	Domain Name System
DTLS	Datagram Transport Layer Security
EAL	Evaluation Assurance Level
EAP	Extensible Authentication Protocol
EAPOL	Extensible Authentication Protocol Over LAN
EAP-TLS	Extensible Authentication Protocol Transport Layer Security

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Abbreviation	Meaning
EAR	Entropy Assessment Report
EC	Elliptic Curve
ECB	Electronic Codebook
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECID	Exclusive Chip Identification
EDGE	Enhanced Data rates for GSM Evolution
EKU	Extended Key Usage
EST	Enrollment over Secure Transport
EV-DO	Evolution-Data Optimized
FAR	False Acceptance Rate
FIPS	Federal Information Processing Standard
FPN	Fixed Pattern Noise
FQDN	Fully Qualified Domain Name
FRR	False Rejection Rate
GCM	Galois/Counter Mode
GID	Group Key
GPS	Global Positioning Satellites
GSM	Global System for Mobile
GTK	Group Temporal Key
HCI	Host Controller Interface
HFP	Hands-Free Profile
HID	Human Interface Device Profile
HMAC	Keyed-hash Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
HSDPA	High Speed Downlink Packet Access
HSPA+	High Speed Packet Access Plus
ID	Identity
IKE	Internet Key Exchange
IOMMU	Input–Output Memory Management Unit
IPsec	Internet Protocol Security
ISA	Instruction Set Architecture
IV	Initialization Vector
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
KAT	Known Answer Test
KDF	Key Derivation Function
KEK	Key Encryption Key
KEXT	Kernel Extension
KW	Key Wrap
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy
ltc	LibTomCrypt
LTE	Long Term Evolution
MAC	Message Authentication Code
MAP	Message Access Profile
MD	Mobile Device
MDFPP	Mobile Device Fundamentals Protection Profile
MDM	Mobile Device Management
MMI	Man-Machine Interface

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Abbreviation	Meaning
MMU	Memory Management Unit
NDRNG	Non-deterministic Random Number Generator
neon	ARM NEON instructions
NITZ	Network, Identity and Time Zone
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
OSP	Organizational Security Policy
OTA	Over-the-Air
OTP-ROM	One Time Programmable Read-Only Memory
PAA	Processor Algorithm Accelerator
PAD	Presentation Attack Detection
PAE	Port Access Entity
PAN	Personal Area Network Profile
PBAP	Phone Book Access Profile
PBKDF	Password-Based Key Derivation Function
PCIe	Peripheral Component Interconnect Express
PHY	Physical Layer
PKCS	Public Key Cryptography Standards
PMK	Pairwise Master Key
POST	Pre-Operational Self-Tests
PP	Protection Profile
PRF	Pseudorandom Function
RAM	Random Access Controller
RBG	Random Bit Generator
RFCOMM	Radio Frequency Communication
REK	Root Encryption Key
RFC	Request for Comments
RISC	Reduced Instruction Set Computing
RSA	Rivest-Shamir-Adleman
S/MIME	Secure/Multipurpose Internet Mail Extensions
SA	Security Association
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SAT	Satellite radio
SCDMA	Synchronous Code Division Multiple Access
SCEP	Simple Certificate Enrollment Protocol
SDIO	Secure Digital Input Output
SEE	Separate Execution Environment
SEP	Secure Enclave Processor
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SigGen	Signature Generation
SigVer	Signature Verification
skg	Secure Key Generation
SKS	Secure Key Store
SiP	System-in-Package
SoC	System on Chip
SP	Security Policy
SP	Special Publication
SPD	Security Policy Database

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Abbreviation	Meaning
SSID	Service Set Identifier
SSL	Secure Sockets Layer
ST	Security Target
TD	Technical Decision
TD-LTE	Time Division Long-Term Evolution
TD-SCDMA	Time Division Synchronous Code Division Multiple Access
TLS	Transport Layer Security
TOE	Target of Evaluation
TRNG	True Random Number Generator
TSF	TOE Security Functionality
TSS	TOE Summary Specification
UDID	Unique Device Identifier
UI	User Interface
UMTS	Universal Mobile Telecommunications System
USSD	Unstructured Supplementary Service Data
UID	Unique Identifier
UUID	Universally Unique Identifier
UWB	Ultra-Wideband
vng	Vector Next Generation
VPN	Virtual Private Network
VPP	Volume Purchase Program
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3
XN	Execute Never
XEX	Xor-Encrypt-Xor
XTS	XEX-based tweaked-codebook mode with ciphertext stealing

8.2. References

ABM_Guide Apple Business Manager User Guide
Date 2025
Location https://support.apple.com/guide/apple-business-manager/welcome/web

AConfig Apple Configurator 2 User Guide for Mac
Date 2025
Location https://support.apple.com/guide/apple-configurator-2/welcome/mac

Agent PP-Module for MDM Agents
Version 1.0
Date 2019-04-25
Location https://www.niap-ccevs.org/protectionprofiles/441

AP_SEC Apple Platform Security
Date 2024-12
Location https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf

APFS_DEV_DOC About Apple File System
Date 2025

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

	Location https://developer.apple.com/documentation/foundation/file_system/about_apple_file_system
APFS_DOC	File System Formats Available in Disk Utility on Mac Date 2025 Location https://support.apple.com/en-euro/guide/disk-utility/dsku19ed921c/mac
BIO	PP-Module: collaborative PP-Module for Biometric enrolment and verification – for unlocking the device - [BIOPP-Module] Version 1.1 Date 2022-09-12 Location https://www.niap-ccevs.org/protectionprofiles/476
BLUETOOTH_HELP	Pair a third-party Bluetooth accessory with your iPhone or iPad Date 2025-02-19 Location https://support.apple.com/en-us/105108
BT	PP-Module for Bluetooth Version 1.0 Date 2021-04-15 Location https://www.niap-ccevs.org/protectionprofiles/425
BT_SPEC	Bluetooth Specifications Date 2021-07-13 Location https://www.bluetooth.com/specifications/
CC	Common Criteria for Information Technology Security Evaluation Version 3.1R5 Date April 2017 Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf
CCGUIDE	Apple iOS 18: iPhone and Apple iPadOS 18: iPad Common Criteria Configuration Guide Date 2025 Location This document
CertPinning	Identity Pinning: How to configure server certificates for your app Date 2021-01-14 Location https://developer.apple.com/news/?id=g9ejcf8y
CKTSREF	Certificate, Key, and Trust Services Date 2025 Location https://developer.apple.com/documentation/security/certificate-key-and-trust-services
DeployRef	Apple Platform Deployment Date 2025-01 Location https://support.apple.com/guide/deployment/welcome/web
DEV_MAN	Device Management Date 2025 Location https://developer.apple.com/documentation/devicemanagement
FIPS140-3	Security Requirements for Cryptographic Modules

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

	Date	2019-03-22
	Location	https://csrc.nist.gov/pubs/fips/140-3/final
FIPS180-4	Secure Hash Standard (SHS)	
	Date	2015-08
	Location	https://csrc.nist.gov/pubs/fips/180-4/upd1/final
FIPS186-5	Digital Signature Standard (DSS)	
	Date	2023-02-03
	Location	https://csrc.nist.gov/pubs/fips/186-5/final
FIPS197	Advanced Encryption Standard (AES)	
	Date	2023-05-09
	Location	https://csrc.nist.gov/pubs/fips/197/final
FIPS19801	The Keyed-Hash Message Authentication Code (HMAC)	
	Date	2008-07
	Location	https://csrc.nist.gov/pubs/fips/198-1/final
iPad_UG	iPad User Guide iPadOS 18 (Latest)	
	Date	2025
	Location	https://support.apple.com/guide/ipad/welcome/18.0/ipados/18.0
KEYCHAINPG	Keychain Services (Programming Guide)	
	Date	2025
	Location	https://developer.apple.com/documentation/security/keychain-services
LOGGING	Logging	
	Date	2025
	Location	https://developer.apple.com/documentation/os/logging
MANAGE_CARDS	Change or Remove the Payment Card that You Use with Apple Pay	
	Date	2025-02-18
	Location	https://support.apple.com/en-us/118219
MDF	Base-PP: Protection Profile for Mobile Device Fundamentals	
	Version	3.3
	Date	2022-09-12
	Location	https://www.niap-ccevs.org/protectionprofiles/468
PASSCODE_Help	Use a passcode with your iPhone, iPad or iPod touch	
	Date	2024-11-27
	Location	https://support.apple.com/en-us/119586
PAY_SETUP	Set Up Apple Pay	
	Date	2024-12-17
	Location	https://support.apple.com/en-us/108398
PP-Config	PP-Configuration for Mobile Device Fundamentals, Biometric enrollment and verification – for unlocking the device, Bluetooth, MDM Agents, Virtual Private Network (VPN) Clients, and WLAN Clients	
	Version	1.1
	Date	2025-01-03
	Location	https://www.niap-ccevs.org/protectionprofiles/487

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

PROFS_LOGS	Profiles and Logs Date 2025 Location https://developer.apple.com/bug-reporting/profiles-and-logs/?platform=ios
RFC3394	Advanced Encryption Standard (AES) Key Wrap Algorithm Date 2002-09 Location https://www.ietf.org/rfc/rfc3394.txt
RFC4301	Security Architecture for the Internet Protocol Date 2005-12 Location https://www.ietf.org/rfc/rfc4301.txt
RFC5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile Date 2008-05 Location https://www.ietf.org/rfc/rfc5280.txt
RFC5746	Transport Layer Security (TLS) Renegotiation Indication Extension Date 2010-02 Location https://www.ietf.org/rfc/rfc5746.txt
RFC5996	Internet Key Exchange Protocol Version 2 (IKEv2) Date 2010-09 Location https://www.ietf.org/rfc/rfc5996.txt
RFC7748	Elliptic Curves for Security Date 2016-01 Location https://www.ietf.org/rfc/rfc7748.txt
RFC8017	PKCS #1: RSA Cryptography Specifications Version 2.2 Date 2016-11 Location https://www.ietf.org/rfc/rfc8017.txt
RFC88994	Simple Certificate Enrolment Protocol Date 2020-09 Location https://www.ietf.org/rfc/rfc8894.txt
SP800-132	Recommendation for Password-Based Key Derivation: Part 1: Storage Applications Date 2010-12 Location https://csrc.nist.gov/pubs/sp/800/132/final
SP800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques Date 2001-12 Location https://csrc.nist.gov/pubs/sp/800/38/a/final
SP800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode forAuthentication and Confidentiality Date 2007-07-20 Location https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final
SP800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode(GCM) and GMAC Date 2007-11 Location https://csrc.nist.gov/pubs/sp/800/38/d/final

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SP800-38E	Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode forConfidentiality on Storage Devices Date 2010-01 Location https://csrc.nist.gov/pubs/sp/800/38/e/final
SP800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping Date 2012-12 Location https://csrc.nist.gov/pubs/sp/800/38/f/final
SP800-56A-Rev3	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography Date 2018-04 Location https://csrc.nist.gov/pubs/sp/800/56/a/r3/final
SP800-56B-Rev2	Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography Date 2019-03 Location https://csrc.nist.gov/pubs/sp/800/56/b/r2/final
SP800-56C-Rev2	Recommendation for Key-Derivation Methods in Key-Establishment Schemes Date 2020-08 Location https://csrc.nist.gov/pubs/sp/800/56/c/r2/final
SP800-90A-Rev1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators Date 2015-06 Location https://csrc.nist.gov/pubs/sp/800/90/a/r1/final
TLSPKG	Functional Package for Transport Layer Security (TLS) Version 1.1 Date 2019-03-01 Location https://www.niap-ccevs.org/protectionprofiles/439
TRUST_STORE	List of available trusted root certificates in iOS 18, iPadOS 18, macOS 15, tvOS 18, visionOS 2, and watchOS 11 Date 2024-11-15 Location https://support.apple.com/en-us/121672
VPNC	PP-Module for Virtual Private Network (VPN) Clients Version 2.5 Date 2024-06-24 Location https://www.niap-ccevs.org/protectionprofiles/487
WLANC	PP-Module for WLAN Clients Version 1.0 Date 2022-03-31 Location https://www.niap-ccevs.org/protectionprofiles/463

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

A. Appendixes

A.1 Devices Covered by this Evaluation

This appendix lists the devices that are covered by this evaluation and provides the technical characteristics of each device including the instruction set architecture a.k.a. microarchitecture.

The following shorthand expressions are used in this section.

BAF	Biometric Authentication Factor Type
BT	Bluetooth (version) (2400 MHz to 2483.5 MHz)
Core	Wi-Fi/Bluetooth chip core
ISA	Instruction Set Architecture
SAT	Satellite (Emergency SOS via satellite) (Globalstar's L and S bands)
SoC	System on Chip
UWB	Ultra-Wideband

SoC	ISA
A13 Bionic	ARMv8.4-A
A14 Bionic	ARMv8.5-A
A15 Bionic	ARMv8.6-A
A17 Pro	
M1	ARMv8.5-A
M2	ARMv8.6-A
M4	ARMv9.2-A

Table 32: SoC to ISA Mappings

Device Info	Model No.	Cellular
iPad 10.2-inch (9th Gen) **BAF: Touch ID (Gen.1)** **BT: 4.2** **SoC : A13 Bionic** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4355**	A2602	Wi-Fi only
	A2603	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) Gigabit-class LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 29, 30, 34, 38, 39, 40, 41, 66, and 71)
	A2604	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) Gigabit-class LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 34, 38, 39, 40, 41, and 66)
	A2605	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) Gigabit-class LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 34, 38, 39, 40, 41, and 66)

Table 33: iPad: A13 Bionic (ARMv8.4-A) Models

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Device Info	Model No.	Cellular
iPad Air (4th Gen) **BAF: Touch ID (Gen.4)** **BT: 5.0** **SoC : A14 Bionic** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2072	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) Gigabit-class LTE, (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 34, 38, 39, 40, 41, 66)
	A2316	Wi-Fi only
	A2324	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) Gigabit-class LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 29, 30, 34, 38, 39, 40, 41, 46, 48, 66, 71)
	A2325	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) 4G LTE Advanced (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 34, 38, 39, 40, 41, 42, 46, 48, 66)
iPad 10.9-inch (10th Gen) **BAF: Touch ID (Gen.4)** **BT: 5.2** **SoC : A14 Bionic** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2696	Wi-Fi only
	A2757	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
	A2777	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Table 34: iPad: A14 Bionic (ARMv8.5-A) Models

Device Info	Model No.	Cellular
iPad mini (6th Gen) **BAF: Touch ID (Gen.4)** **BT: 5.0** **SoC : A15 Bionic** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2567	Wi-Fi only
	A2568	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	A2569	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)

Table 35: iPad: A15 Bionic (ARMv8.6-A) Model

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Device Info	Model No.	Cellular
iPad mini (7th Gen) **BAF: Touch ID (Gen.4)** **BT: 5.3** **SoC : A17 Pro** **Wi-Fi: Wi-Fi 6E (802.11ax) with 2x2 MIMO** **Core: 4388**	A2995	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
	A2993	Wi-Fi only
	A2996	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Table 36: iPad: A17 Pro (ARMv8.6-A) Model

Device Info	Model No.	Cellular
iPad Pro 11-inch (3rd Gen) **BAF: Face ID (Gen.1)** **BT: 5.0** **SoC : M1** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2301	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) 5G NR mmWave (Bands n260, n261) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	A2377	Wi-Fi only
	A2459	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) 5G NR mmWave (Bands n260, n261) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	A2460	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
iPad Pro 12.9-inch (5th Gen) **BAF: Face ID (Gen.1)** **BT: 5.0** **SoC : M1** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2378	Wi-Fi only
	A2379	Wi-Fi only
	A2461	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) 5G NR mmWave (Bands n260, n261) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Device Info	Model No.	Cellular
	A2462	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
iPad Air (5th Gen) **BAF: Touch ID (Gen.4)** **BT: 5.0** **SoC : M1** **Wi-Fi: Wi-Fi 6 (802.11ax)** **Core: 4387**	A2588	Wi-Fi only
	A2589	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n29, n30, n38, n40, n41, n48, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
	A2591	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n29, n30, n38, n40, n41, n48, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Table 37: iPad: M1 (ARMv8.5-A) Models

Device Info	Model No.	Cellular
iPad Pro 11-inch (4th Gen) **BAF: Face ID (Gen.1)** **BT: 5.3** **SoC : M2** **Wi-Fi: Wi-Fi 6E (802.11ax)** **Core: 4388**	A2435	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n28, n29, n30, n38, n40, n41, n48, n66, n71, n77, n78, n79) 5G NR mmWave (Bands n258, n260, n261)
	A2759	Wi-Fi only
	A2761	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1.700/2.100, 1.900, 2.100 MHz)
	A2762	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n29, n30, n38, n40, n41, n48, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
iPad Pro 12.9-inch (6th Gen) **BAF: Face ID (Gen.1)** **BT: 5.3** **SoC : M2** **Wi-Fi: Wi-Fi 6E (802.11ax)** **Core: 4388**	A2436	Wi-Fi only
	A2437	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1.700/2.100, 1.900, 2.100 MHz)
	A2764	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n29, n30, n38, n40, n41, n48, n66, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

Device Info	Model No.	Cellular
	A2766	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 46, 48) UMTS/HSPA/HSPA+/DC HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Table 38: iPad: M2 (ARMv8.6-A) Models

Device Info	Model No.	Cellular
iPad Pro 13-inch (M4) **BAF: Face ID (Gen.3)** **BT: 5.3** **SoC : M4** **Wi-Fi: Wi-Fi 6E (802.11ax) with 2x2 MIMO** **Core: 4388**	A2926	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n75, n76, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
	A2925	Wi-Fi only
	A3007	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n75, n76, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
iPad Pro 11-inch (M4) **BAF: Face ID (Gen.3)** **BT: 5.3** **SoC : M4** **Wi-Fi: Wi-Fi 6E (802.11ax) with 2x2 MIMO** **Core: 4388**	A2837	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n75, n76, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)
	A2836	Wi-Fi only
	A3006	5G NR (Bands n1, n2, n3, n5, n7, n8, n12, n14, n20, n25, n26, n28, n29, n30, n38, n40, n41, n48, n66, n70, n71, n75, n76, n77, n78, n79) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 11, 12, 13, 14, 17, 18, 19, 20, 21, 25, 26, 28, 29, 30, 32, 66, 71) TD-LTE (Bands 34, 38, 39, 40, 41, 42, 48) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz)

Table 39: iPad: M4 (ARMv9.2-A) Models

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

A.2 Wi-Fi Alliance Certificates

The following table lists the Wi-Fi Alliance certificates for the devices covered by this evaluation.

SoC	Device Name	Model No.	Wi-Fi Alliance
A13 Bionic	iPad 10.2-inch (9th Gen)	A2602	WFA113796
		A2603	WFA113795
		A2604	WFA113795
		A2605	WFA113796
A14 Bionic	iPad Air (4th Gen)	A2316	WFA99939 / WFA99940
		A2324	
		A2072	
		A2325	
	iPad 10.9-inch (10th Gen)	A2696	WFA120871
		A2757	WFA120040
		A2777	WFA120040
A15 Bionic	iPad mini (6th Gen)	A2567	WFA113722
		A2568	WFA112068
		A2569	WFA112068
		A3096	
A17 Pro	iPad mini (7th Gen)	A2995	WFA132087
		A2993	
		A2996	WFA132087
M1	iPad Pro 11-inch (3rd Gen)	A2301	WFA110479
		A2377	
		A2459	
		A2460	
	iPad Pro 12.9-inch (5th Gen)	A2378	WFA110951
		A2379	
		A2461	
		A2462	
	iPad Air (5th Gen)	A2588	WFA110951
		A2589	
		A2591	
M2	iPad Pro 11-inch (4th Gen)	A2436	WFA120037
		A2759	WFA120872
		A2761	WFA120037
		A2762	WFA120872
	iPad Pro 12.9-inch (6th Gen)	A2436	WFA120875
		A2437	WFA120874

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SoC	Device Name	Model No.	Wi-Fi Alliance
		A2764	WFA120874
		A2766	WFA120875
M4	iPad Pro 13-inch (M4)	A2926	WFA130063
		A2925	WFA130065
		A3007	WFA130063
	iPad Pro 11-inch (M4)	A2837	WFA130062
		A2836	WFA130064
		A3006	WFA130062

Table 40: iPad: Wi-Fi Alliance Certificates

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

A.3 SFR to CAVP Certificate Mappings

This appendix maps the SFRs to the CAVP certificates.

CAVP Mapping Table

SFR	Algorithm	Modes/Other	Standard	CAVP
Mapping of SFRs to CAVP certificates (USR cryptographic library)				
FCS_CKM.1	ECC	Curves: P-256, P-384, P-521	[FIPS186-5]	A6512
	FFC	Groups: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	[SP800-56Ar3]	A6510
FCS_CKM.1/VPN	ECDSA	Curves: P-256, P-384, P-521	[FIPS186-5]	A6512
FCS_CKM.2/UN-LOCKED	KAS-ECC-SSC	Curves: P-256, P-384, P-521	[SP800-56Ar3]	A6510
	KAS-FFC-SSC	Groups: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	[SP800-56Ar3]	A6510
	RSA Key Establish-ment	Modulus: 2048, 3072, 4096	[RFC8017]	CCTL Tested
FCS_CKM_EXT.3	HKDF	HMAC-SHA2-256	[SP800-56Cr2]	A6512
FCS_COP.1/ENCRYPT	AES-CBC	128-bits, 256-bits encrypt, decrypt	[SP800-38A]	A6507
	AES-CCM	128 bits, 256 bits encrypt, decrypt	[SP800-38C]	A6511
	AES-GCM	128 bits, 256 bits encrypt, decrypt	[SP800-38D]	A6511
	AES-KW	128 bits, 256 bits encrypt, decrypt	[SP800-38F]	A6508
	AES-XTS	128 bits, 256 bits encrypt, decrypt	[SP800-38E]	A6507
FCS_COP.1/HASH	SHA-1	Byte-oriented mode	[FIPS180-4]	A6512
	SHA2-256, SHA2-384, SHA2-512			A6513
FCS_COP.1/SIGN	RSA SigGen	Modulus: 2048, 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A6512
	RSA SigVer	Modulus: 2048, 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A6512
	ECDSA SigGen	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6512

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	Algorithm	Modes/Other	Standard	CAVP
	ECDSA SigVer	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6512
FCS_COP.1/KEYHMAC	HMAC-SHA-1	Byte-oriented mode	[FIPS198-1]	A6512
	HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512			A6513
FCS_RBG_EXT.1/SW	CTR_DRBG	AES-256	[SP800-90Ar1]	A6511
Mapping of SFRs to CAVP certificates (KRN cryptographic library)				
FCS_CKM.1	ECC	Curves: P-256, P-384, P-521	[FIPS186-5]	A6407
FCS_COP.1/ENCRYPT	AES-CBC	128 bits, 256 bits encrypt, decrypt	[SP800-38A]	A6403
	AES-CCM	128 bits, 256 bits encrypt, decrypt	[SP800-38C]	A6406
	AES-GCM	128 bits, 256 bits encrypt, decrypt	[SP800-38D]	A6406
	AES-KW	128 bits, 256 bits encrypt, decrypt	[SP800-38F]	A6404
	AES-XTS	128 bits, 256 bits encrypt, decrypt	[SP800-38E]	A6403
FCS_COP.1/HASH	SHA-1	Byte-oriented mode	[FIPS180-4]	A6407
	SHA2-256, SHA2-384, SHA2-512			A6408
FCS_COP.1/SIGN	RSA SigGen	Modulus: 2048, 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A6407
	RSA SigVer	Modulus: 2048, 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A6407
	ECDSA SigGen	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6407
	ECDSA SigVer	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6407
FCS_COP.1/KEYHMAC	HMAC-SHA-1	Byte-oriented mode	[FIPS198-1]	A6407
	HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512			A6408
FCS_RBG_EXT.1/SW	CTR_DRBG	AES-256	[SP800-90Ar1]	A6406
Mapping of SFRs to CAVP certificates (SKS-FW cryptographic library)				

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

SFR	Algorithm	Modes/Other	Standard	CAVP
FCS_CKM.1	ECC	Curves: P-256, P-384, P-521	[FIPS186-5]	A6407
	ECC	Curve25519	[RFC7748]	CCTL Tested
FCS_CKM.2/LOCKED	ECC	Curve25519	[RFC7748]	CCTL Tested
FCS_CKM.2/UN-LOCKED	KAS-ECC-SSC	Curves: P-256, P-384, P-521	[SP800-56Ar3]	A6557
FCS_COP.1/ENCRYPT	AES-CBC	128 bits, 256 bits encrypt, decrypt	[SP800-38A]	A6554
	AES-CCM	128 bits, 256 bits encrypt, decrypt	[SP800-38C]	A6559
	AES-GCM	128 bits, 256 bits encrypt, decrypt	[SP800-38D]	A6559
	AES-KW	128 bits, 256 bits encrypt, decrypt	[SP800-38F]	A6555
	AES-XTS	128 bits, 256 bits encrypt, decrypt	[SP800-38E]	A6554
FCS_COP.1/HASH	SHA-1	Byte-oriented mode	[FIPS180-4]	A6560
	SHA2-256, SHA2-384, SHA2-512			A6561
FCS_COP.1/SIGN	ECDSA SigGen	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6560
	ECDSA SigVer	Curves: P-256, P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A6560
FCS_COP.1/KEYHMAC	HMAC-SHA-1	Byte-oriented mode	[FIPS198-1]	A6560
	HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512			A6561
Mapping of SFRs to CAVP certificates (SKS-HW cryptographic library)				
FCS_COP.1/ENCRYPT	AES-CBC	128 bits, 256 bits encrypt, decrypt	[SP800-38A]	A510, A1469, A2863, A3496, A6547
FCS_RBG_EXT.1/HW	CTR_DRBG	AES-256	[SP800-90Ar1]	A501, A1362, A2864, A3490, A6548
Mapping of SFRs to CAVP certificates (BC cryptographic library)				
FCS_COP.1/ENCRYPT	AES-CCM	128 bits encrypt, decrypt	[SP800-38C]	AES 3678, AES 5926, AES 5927, A1932
		256 bits encrypt, decrypt	[SP800-38C]	AES 5952, AES 5953, A1932
	AES-GCM	128 bits, 256 bits encrypt, decrypt	[SP800-38D]	AES 5926, AES 5927, A1932

Table 41: SFRs to CAVP Certificates for iPad

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

CAVP Mapping of Broadcom Cores

Table 42 maps iPad to Broadcom cores and Broadcom CAVP certificates.

iPad	Broadcom Core # (Version)	Supported Algorithms
iPad 10.2-inch (9th gen) (A13 Bionic)	4355 (aes_core.vhd rev 10)	AES-CCM-128
iPad Air (4th gen) (A14 Bionic) iPad (10th gen) (A14 Bionic) iPad mini (6th gen) (A15 Bionic) iPad Pro 11-inch (3rd gen) (M1) iPad Pro 12.9-inch (5th gen) (M1) iPad Air (5th gen) (M1)	4387 Aux (2.4 GHz) (aes_core_gcm.vhd rev 6)	AES-CCM-128 AES-GCM-128 AES-GCM-256
		AES-CCM-256
	4387 Main (5 GHz) (aes_core_gcm_simult_enc_mic.vhd rev 6)	AES-CCM-128 AES-GCM-128 AES-GCM-256
		AES-CCM-256
iPad mini (7th gen) (A17 Pro) iPad Pro 11-inch (4th gen) (M2) iPad Pro 12.9-inch (6th gen) (M2) iPad Pro 11-inch (M4) iPad Pro 13-inch (M4)	4388 Aux (2.4 GHz) (aes_core_gcm.vhd rev 6)	AES-CCM-128 AES-GCM-128 AES-GCM-256
		AES-CCM-256
	4388 Main (5 GHz) (aes_core_gcm_simult_5_cycle.vhd rev 2)	AES-CCM-128 AES-CCM-256 AES-GCM-128 AES-GCM-256

Table 42: Broadcom Core Supported Algorithms and CAVP Certificates

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision

A.4 Inventory of TSF Binaries and Libraries

The list in this appendix is **proprietary** and only appears in the proprietary version of the Supplemental Annex. In addition, this annex contains the inventory of TSF binaries and libraries required by the Assurance Activity for FPT_AEX_EXT.3.

Note that the list is considered **proprietary** because entries in this list are specifically for internal development and do not reflect the production environment, however, by design one cannot capture the file listing on a production build of the TOE OS.

Copyright © 2025 Apple Inc. All rights reserved.
This document may be reproduced and distributed only in its original entirety without revision