Intelligent Waves Virtual Mobile Infrastructure Platform 4.1

Hypori Client (iOS) Security Target

Version 4.1
August 14, 2018

Prepared by:
Intelligent Waves, LLC
1801 Robert Fulton Drive, Suite 440
Reston, VA 20191

Copyright

LIST OF TABLES

1. Security Target Introduction

This section identifies the Target of Evaluation (TOE) along with identification of the Security Target (ST) itself. The section includes documentation organization, ST conformance claims, and ST conventions.

The TOE is the Hypori Client component of the Virtual Mobile Infrastructure Platform version 4.1 provided by Intelligent Waves, LLC

The Security Target contains the following additional sections:

- Security Target Introduction (Section 1)
- TOE Description (Section 2)
- Security Problem Definition (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8).
- Appendix: iOS APIs (Section 9)

1.1 Security Target, TOE and CC Identification

ST Title – Intelligent Waves Virtual Mobile Infrastructure Platform 4.1 Hypori Client (iOS) Security Target

ST Version – Version 4.1

ST Date – August 14, 2018

TOE Identification – Hypori Client (iOS) 4.1

TOE Developer – Intelligent Waves, LLC

Evaluation Sponsor – Intelligent Waves, LLC

CC Identification – *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, September 2012*

1.2 Conformance Claims

This TOE is conformant to the following CC specifications:

- This ST is conformant to the *Protection Profile for Application Software*, Version 1.2, 22 April 2016 (PP APP SW) including *DoD Annex for Protection Profile for Application Software v1.2,* Version 1, Release 1, 21 February 2018. The following NIAP Technical Decisions apply to evaluation assurance activities.
    - TD0238 – User-modifiable files FTP_AEX_EXT.1.4
    - TD0217 – Compliance to RFC5759 and RFC5280 for using CRLs
    - TD0192 – Update to FCS_STO_EXT.1 Application Note
    - TD0178 – Integrity for installation tests in AppSW PP
    - TD0174 – Optional Ciphersuites for TLS
    - TD0172 – Additional APIs added to FCS_RBG_EXT.1.1
    - TD0163 – Update to FCS_TLSC_EXT.1.1 Test 5.4 and FCS_TLSS_EXT.1.1 Test
    - TD0221 – FMT_SMF.1.1 Assignments moved to Selections
    - TD0119 – FCS_STO_EXT.1.1 in PP_APP_v1.2

    - TD0107 – FCS_CKM - ANSI X9.31-1998, Section 4.1 for Cryptographic Key Generation
    - TD0244: FCS_TLSC_EXT - TLS Client Curves Allowed
    - TD0268: FMT_MEC_EXT.1 Clarification
    - TD0283: Cipher Suites for TLS in SWApp v1.2
    - TD0300: Sensitive Data in FDP_DAR_EXT.1
    - TD0304: Update to FCS_TLSC_EXT.1.2
    - TD0305: Handling of TLS connections with and without mutual authentication
    - TD0327: Default file permissions for FMT_CFG_EXT.1.2

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 4, September 2012.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 4, September 2012.
    - Part 3 Extended

1.3 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a number in parentheses placed at the end of the component. For example, FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement, (1) and (2).
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …"). Note that 'cases' that are not applicable in a given SFR have simply been removed without any explicit identification.
- The PP APP SW uses an additional convention – the 'case' – which defines parts of an SFR that apply only when corresponding selections are made or some other identified conditions exist. Only the applicable cases are identified in this ST and they are identified using **bold** text.
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.3.1 Terminology

PP APP SW provides definitions for terms specific to the application software technology as well as general Common Criteria terms. The technology-specific terms are:

- Address Space Layout Randomization
- Application
- Application Programming Interface
- Credential
- Data Execution Prevention
- Developer
- Mobile Code
- Operating System
- Personally Identifiable Information
- Platform
- Sensitive Data
- Stack Cookie
- Vendor

Terms from the Common Criteria are:

- Common Criteria
- Common Evaluation Methodology
- Protection Profile
- Security Target
- Target of Evaluation
- TOE Security Functionality
- TOE Summary Specification
- Security Functional Requirement
- Security Assurance Requirement

This ST does not include additional technology-specific terminology.

1.3.2 Abbreviations

This section identifies abbreviations and acronyms used in this ST.

API	Application Programming Interface
App	Software application
ASLR	Address Space Layout Randomization
CC	Common Criteria
CEM	Common Evaluation Methodology
DEP	Data Execution Prevention
DoD	Department of Defense
OS	Operating System
PII	Personally Identifiable Information
PP	Protection Profile
PP APP SW	Protection Profile for Application Software
SAR	Security assurance requirement
SFR	Security functional requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality

TSS	TOE Summary Specification
VMI	Virtual Mobile Infrastructure

2. TOE Description

After a brief overview of the Hypori Virtual Mobile Infrastructure product, this section describes its iOS-based Hypori Client component, which is the Target of Evaluation (TOE). The description covers TOE architecture, logical boundaries, and physical boundaries.

2.1 Product Overview

In the Hypori Virtual Mobile Infrastructure (VMI) platform, end users running an iOS-based Hypori Client on their mobile device access a virtual Android device running on a server in the cloud. The virtual device on the server contains the operating system, the data, and the applications, using TLS 1.2 encryption to communicate securely with the Hypori Client. The Hypori iOS thin client application provides secure access to the remote Android virtual device and brokers access between the mobile device's sensors and the applications executing in the virtual device on a Hypori server. The client applications are agnostic to the version of Android executing in the virtual device.

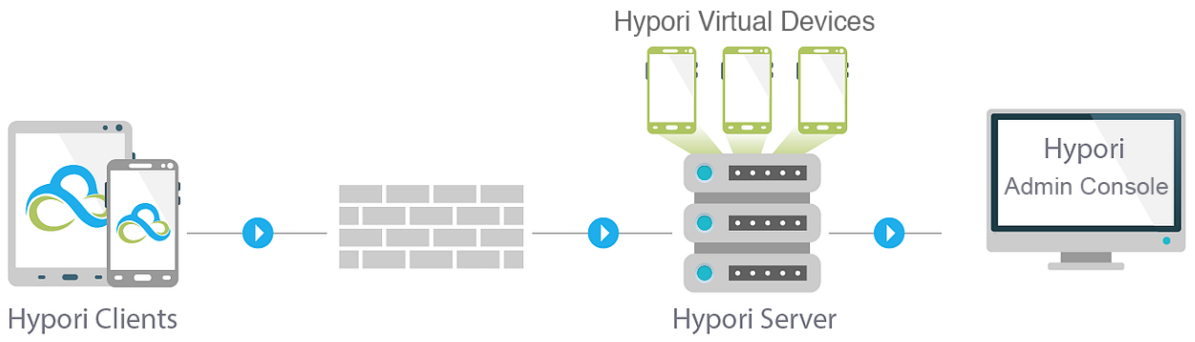

Figure 1 Hypori Virtual Mobile Infrastructure (VMI)

The Hypori VMI platform includes the following components:

- **Hypori Client:** This is an iOS-based thin client that installs on the end user's mobile device and communicates with the Hypori Virtual Device on the server through secure encrypted protocols.
- **Hypori Virtual Device:** This is an Android-based virtualized mobile device executing on a server in the cloud.
- **Hypori Servers:** This is the cloud server cluster that hosts the Hypori Virtual Devices.
- **Hypori Admin Console:** This is a browser-based administration user interface that is used to manage the Hypori system.

2.2 TOE Overview

The TOE is the Hypori Client. The following diagram shows how the TOE interacts with a Hypori Virtual Device running applications on a Hypori Server. The Hypori Client is a thin client that communicates only with a Hypori Device on a Hypori Server and not with other servers or applications.

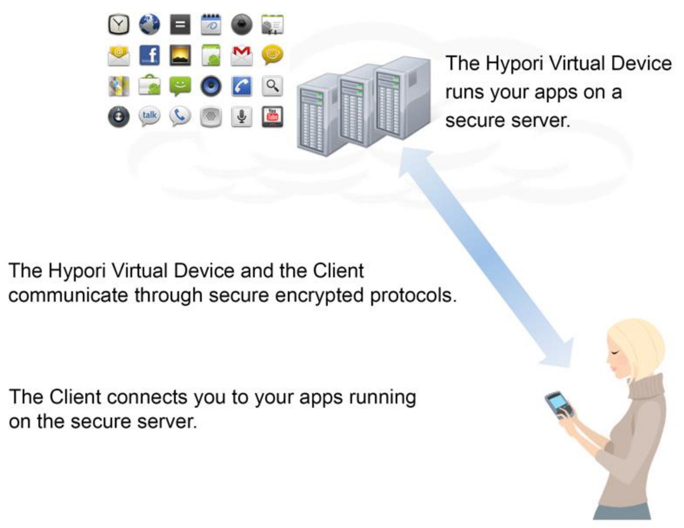
The Hypori Virtual Device runs your apps on a secure server.
The Hypori Virtual Device and the Client communicate through secure encrypted protocols.
The Client connects you to your apps running on the secure server.

Figure 2 iOS Hypori Client as Part of VMI Platform

2.3 TOE Architecture

The section describes the TOE architecture including physical and logical boundaries. Figure 3 shows the relationship of the TOE to its operational environment along with the TOE boundary. The security functional requirements identify the libraries included in the application package.

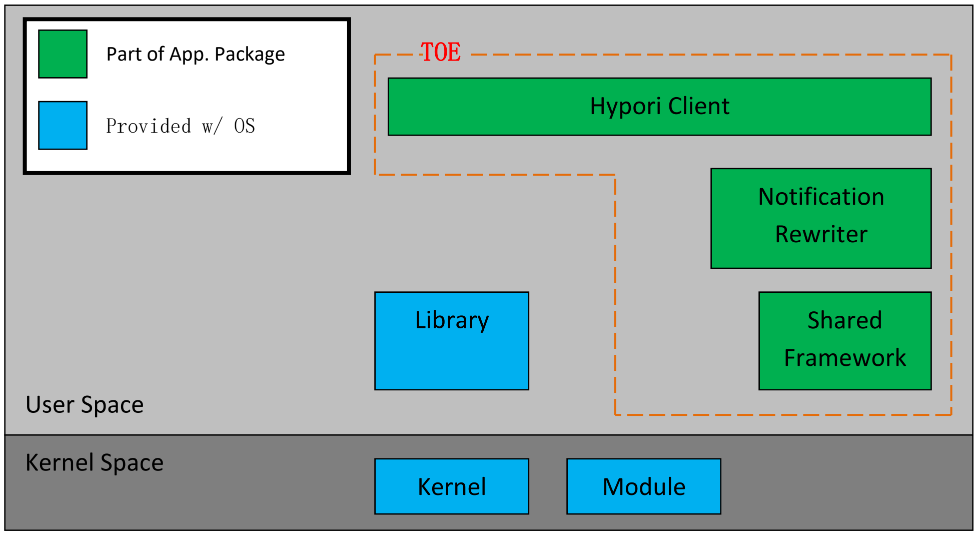

Figure 3 TOE Boundary for iOS Devices

2.3.1 Physical Boundaries

The TOE consists of an iOS-based Hypori Client application as defined in the Hypori Client installation package. The Hypori Client is a thin client that only communicates with the Hypori server. The Hypori server, applications running on the Hypori server, and any functions not specified in this security target are outside the scope of the TOE.

2.3.1.1 Software Requirements

The TOE runs on iOS versions 10.0, 10.1, 10.2, and 10.3.

2.3.1.2 Hardware Requirements

The TOE imposes no hardware requirements beyond iOS operating system requirements.

2.3.2 Logical Boundaries

This section summarizes the security functions provided by the TOE:

- Cryptographic support
- User data protection
- Identification and Authentication
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

2.3.2.1 Cryptographic support

The TOE establishes secure communication with the Hypori server using TLS. The client uses cryptographic services provided by the platform. TOE stores credentials and certificates for mutual authentication in the platform's keychain.

2.3.2.2 User data protection

The TOE informs a user of hardware and software resources the TOE accesses. It uses the platform's permission mechanism to get a user's approval for access as part of the installation process. The user initiates a secure network connection to the Hypori server using the TOE. In general, sensitive data resides on the Hypori server and not the Hypori Client, although the client does store credentials as per section 2.3.2.1.

2.3.2.3 Identification and Authentication

The TOE uses the platform's certification validation services to authenticate the X.509 certificate the Hypori server presents as part of establishing a TLS connection.

2.3.2.4 Security management

Security management consists of setting Hypori Client configuration options. The TOE uses the platform's mechanisms for storing the configuration settings.

2.3.2.5 Privacy

The TOE does not knowingly transmit PII over a network.

2.3.2.6 Protection of the TSF

The TOE uses security features and APIs that the platform provides. The TOE leverages package management for secure installation and updates.

2.3.2.7 Trusted path/channels

TOE uses TLS 1.2 for all communication with the Hypori server.

2.4 TOE Documentation

The TOE includes the following Hypori Client documentation.

- *Hypori User Guide,* Version 4.1.0

- *Hypori User Guide Common Criteria Configuration and Operation*, Version 4.1

3. Security Problem Definition

This security target includes by reference the Security Problem Definition from the PP APP SW. The Security Problem Definition consists of threats that a conformant TOE is expected to address and assumptions about the operational environment of the TOE.

In general, the PP APP SW has presented a Security Problem Definition appropriate for application software that runs on mobile devices, as well as on desktop and server platforms. The Hypori Client is an iOS application running on a mobile device. As such, the PP APP SW Security Problem Definition applies to the TOE.

4. Security Objectives

Like the Security Problem Definition, this security target includes by reference the Security Objectives from the PP APP SW. The PP APP SW security objectives for the operational environment are reproduced below, since these objectives characterize technical and procedural measures each consumer must implement in their operational environment.

In general, the PP APP SW has presented a Security Objectives statement appropriate for application software that runs on mobile devices, as well as on desktop and server platforms. Consequently, the PP APP SW security objectives are suitable for the Hypori Client TOE.

4.1 Security Objectives for the Operational Environment

OE.PLATFORM	The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.
OE.PROPER_USER	The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.
OE.PROPER_ADMIN	The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

5. IT Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The security functional requirements have all been drawn from: *Protection Profile for Application Software*, Version 1.2, 22 April 2016 (PP APP SW). As a result, refinements and operations already performed in that PP are not identified (e.g., highlighted) here, rather the requirements have been copied from that PP and any residual operations have been completed herein. Of particular note, PP APP SW made a number of refinements and completed some of the SFR operations defined in the CC. PP APP SW should be consulted to identify those changes if necessary.

The security assurance requirements are the set of SARs specified in PP APP SW.

5.1 Extended Requirements

All of the extended requirements in this ST have been drawn from the PP APP SW. The PP APP SW defines the following extended SFRs. Since these SFRs are not redefined in this ST, readers should consult PP APP SW for more information in regard to these CC extensions.

- FCS_CKM_EXT.1 Cryptographic Key Generation Services
- FCS_RBG_EXT.1 Random Bit Generation Services
- FCS_STO_EXT.1 Storage of Credentials
- FCS_TLSC_EXT.1 TLS Client Protocol
- FCS_TLSC_EXT.2 TLS Client Protocol
- FCS_TLSC_EXT.4 TLS Client Protocol
- FDP_DAR_EXT.1 Encryption Of Sensitive Application Data
- FDP_NET_EXT.1 Network Communications
- FDP_DEC_EXT.1 Access to Platform Resources
- FIA_X509_EXT.1 X.509 Certificate Validation
- FIA_X509_EXT.2 X.509 Certificate Authentication
- FMT_MEC_EXT.1 Supported Configuration Mechanism
- FMT_CFG_EXT.1 Secure by Default Configuration
- FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information
- FPT_AEX_EXT.1 Anti-Exploitation Capabilities
- FPT_API_EXT.1 Use of Supported Services and APIs
- FPT_LIB_EXT.1 Use of Third Party Libraries
- FPT_TUD_EXT.1 Integrity for Installation and Update
- FTP_DIT_EXT.1 Protection of Data in Transit

5.2 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by the Hypori Client TOE.

Table 1 TOE Security Functional Components

Requirement Class	Requirement Component
FCS: Cryptographic support	FCS_CKM_EXT.1 Cryptographic Key Generation Services
	FCS_RBG_EXT.1 Random Bit Generation Services
	FCS_STO_EXT.1 Storage of Credentials
	FCS_TLSC_EXT.1 TLS Client Protocol
	FCS_TLSC_EXT.2 TLS Client Protocol
	FCS_TLSC_EXT.4 TLS Client Protocol
FDP: User data protection	FDP_DAR_EXT.1 Encryption of Sensitive Application Data
	FDP_DEC_EXT.1 Access to Platform Resources
	FDP_NET_EXT.1 Network Communications
FIA: Identification and authentication	FIA_X509_EXT.1 X.509 Certificate Validation
	FIA_X509_EXT.2 X.509 Certificate Authentication
FMT: Security management	FMT_CFG_EXT.1 Secure by Default Configuration
	FMT_MEC_EXT.1 Supported Configuration Mechanism
	FMT_SMF.1 Specification of Management Functions
FPR: Privacy	FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information
FPT: Protection of the TSF	FPT_AEX_EXT.1 Anti-Exploitation Capabilities
	FPT_API_EXT.1 Use of Supported Services and APIs
	FPT_LIB_EXT.1 Use of Third Party Libraries
	FPT_TUD_EXT.1 Integrity for Installation and Update
FTP: Trusted path/channels	FTP_DIT_EXT.1 Protection of Data in Transit

5.2.1 Cryptographic Support (FCS)

5.2.1.1 Cryptographic Key Generation Services (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1[1] The application shall [***generate no asymmetric cryptographic keys***].

5.2.1.2 Random Bit Generation Services (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1 The application shall [***use no DRBG functionality***] for its cryptographic operations.

5.2.1.3 Storage of Credentials (FCS_STO_EXT.1)

FCS_STO_EXT.1.1[2] The application shall **[*invoke the functionality provided by the platform to securely store [user TLS client key and server account password]*]** to non-volatile memory.

5.2.1.4 TLS Client Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1[3] The application shall [***invoke platform-provided TLS 1.2***] supporting the following cipher suites: [

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289

[1] This SFR was modified per NIAP TD0107

[2] This SFR was modified per NIAP TD0119.

[3] This SFR was modified per NIAP TD0283.

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
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]

FCS_TLSC_EXT.1.2 The application shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3 The application shall establish a trusted channel only if the peer certificate is valid.

5.2.1.5 TLS Client Protocol (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1 The application shall support mutual authentication using X.509v3 certificates.

5.2.1.6 TLS Client Protocol (FCS_TLSC_EXT.4)

FCS_TLSC_EXT.4.1[4] The application shall present the **S**upported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***].

5.2.2 User Data Protection (FDP)

5.2.2.1 Encryption of Sensitive Application Data (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1[5] The application shall [***protect sensitive data in accordance with FCS_STO_EXT.1***,] in nonvolatile memory.

5.2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)

FDP_DEC_EXT.1.1 The application shall restrict its access to [

- ***network connectivity,***
- ***camera,***
- ***microphone,***
- ***location services,***
- ***[Wi-Fi,***
- ***Phone]***

] .

FDP_DEC_EXT.1.2 The application shall restrict its access to [***no sensitive information repositories***].

5.2.2.3 Network Communications (FDP_NET_EXT.1)

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- **user-initiated communication for *[connecting to the Hypori server],***
- **respond to *[push notifications from Apple's APNs platform by polling the Hypori server for notifications],***
- ***[application-initiated communication for polling the Hypori server for notifications],***

[4] This SFR was modified per NIAP TD0244.

[5] This SFR was modified per NIAP TD0300.

] .

5.2.3 Security Management (FMT)

5.2.3.1 Secure by Default Configuration (FMT_CFG_EXT.1)

FMT_CFG_EXT.1.1 The application shall only provide enough functionality to set new credentials when configured with default credentials or no credentials.

FMT_CFG_EXT.1.2[6] The application shall be configured by default with file permissions which protect the application's binaries and data files from modification by normal unprivileged user.

5.2.3.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)

FMT_MEC_EXT.1.1 The application shall invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.

5.2.3.3 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1[7] The TSF shall be capable of performing the following management functions [
[setting configuration options
applying configuration policies from the Hypori server]
] .

5.2.4 Privacy

5.2.4.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)

FPR_ANO_EXT.1.1 The application shall [***not transmit PII over a network***].

Application Note: Note that as per the claimed PP, this requirement applies only to PII that is specifically requested by the application and does not apply if the user volunteers PII without prompting.

5.2.5 Protection of the TSF (FPT)

5.2.5.1 Use of Supported Services and APIs (FPT_API_EXT.1)

FPT_API_EXT.1.1 The application shall use only documented platform APIs.

5.2.5.2 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)

FPT_AEX_EXT.1.1 The application shall not request to map memory at an explicit address except for [**no exceptions**].

FPT_AEX_EXT.1.2 The application shall **[*not allocate any memory region with both write and execute permissions*].**

FPT_AEX_EXT.1.3 The application shall be compatible with security features provided by the platform vendor.

FPT_AEX_EXT.1.4 The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

FPT_AEX_EXT.1.5 The application shall be compiled with stack-based buffer overflow protection enabled.

---

[6] This SFR was modified per NIAP TD0327.

[7] This SFR was modified per NIAP TD0221.

5.2.5.3 Integrity for Installation and Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The application shall [***leverage the platform***] to check for updates and patches to the application software.

FPT_TUD_EXT.1.2 The application shall be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.1.3 The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.

FPT_TUD_EXT.1.4 The application shall not download, modify, replace or update its own binary code.

FPT_TUD_EXT.1.5 The application shall [***provide the ability***] to query the current version of the application software.

FPT_TUD_EXT.1.6 The application installation package and its updates shall be digitally signed such that its platform can cryptographically verify them prior to installation.

5.2.5.4 Use of Third Party Libraries (FPT_LIB_EXT.1)

FPT_LIB_EXT.1.1 The application shall be packaged with only [

- **opus (audio compression)**
- **protocol buffers (from Google)**
- **spice-common (from RedHat for spice message encoding/decoding)**
- **PLCrashReporter (capture crash information for admins)**

].

5.2.6 Trusted path/channels (FTP)

5.2.6.1 Protection of Data in Transit (FTP_DIT_EXT.1)

FTP_DIT_EXT.1.1 The application shall [***encrypt all transmitted data with [TLS]***] between itself and another trusted IT product.

5.2.7 Identification and authentication (FIA)

5.2.7.1 X.509 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1[8] The application shall [***invoked platform-provided functionality***] to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The application 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 application shall validate the revocation status of the certificate using [***the Online Certificate Status Protocol (OCSP) as specified in RFC 2560***].
- The application shall validate the extendedKeyUsage field according to the following rules:

[8] This SFR was modified per NIAP TD0217.

- 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.[9]
- 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.
- S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the extendedKeyUsage field.[10]
- 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.
- 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.[11]

FIA_X509_EXT.1.2 The application shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

5.2.7.2 X.509 Certificate Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1 The application shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***TLS***].

FIA_X509_EXT.2.2 When the application cannot establish a connection to determine the validity of a certificate, the application shall [***not accept the certificate***].[12]

5.3 TOE Security Assurance Requirements

The security assurance requirements in Table 2 are included in this ST by reference from the PP APP SW.

Table 2 Assurance Components

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1 Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE
	ALC_CMS.1 TOE CM coverage
	ALC_TSU_EXT.1 Timely Security Updates
ATE: Tests	ATE_IND.1 Independent testing - conformance
AVA: Vulnerability assessment	AVA_VAN.1 Vulnerability survey

[9] The Hypori Client does not check extended key usage for Code Signing. The Hypori Client relies on the platform update mechanism. While Hypori signs each installation package with a Code Signing certificate, the platform verifies the certificate and package.

[10] The Hypori Client does not check extended key usage for Email Protection, since the Hypori Client does not perform email encryption or email signature verification.

[11] The Hypori Client does not check extended key usage for CMC Registration Authority, since the Hypori Client does not perform Enrollment over Secure Transport.

[12] The Hypori Client relies on the iOS platform for certificate validation services using OCSP.

These assurance requirements imply the following requirements from CC class 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
- ASE_REQ.1 Stated security requirements
- ASE_TSS.1 TOE summary specification

Consequently, the assurance activities specified in PP APP SW apply to the TOE evaluation.

6. TOE Summary Specification

This chapter describes the security functions:

- Cryptographic support
- User data protection
- Certificate validation
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

6.1 Cryptographic support

The Hypori Client makes use of the platform for cryptographic services. The Hypori Client uses platform TLS services for secure communication with the Hypori server, including mutual authentication. The client uses TLS client certificates and keys along with a CA certificate for the server. The user stores these certificates in the platform's key store during installation. The user need not install a CA certificate when the CA is a platform trusted CA.

6.1.1 FCS_CKM_EXT.1

The Hypori Client does not generate cryptographic keys. As part of installation, a user adds a Hypori server TLS client certificate and key to the platform's key store. The Hypori Client relies on the platform for TLS support. The platform generates all ephemeral TLS keys without direct Hypori Client action.

6.1.2 FCS_RBG_EXT.1

The Hypori Client relies on the platform for cryptographic services. Consequently, the Hypori Client itself uses no DRBG functions.

6.1.3 FCS_STO_EXT.1

Table 3 lists each Hypori Client persistent credential along with how the client uses and stores each credential.

Table 3: Persistent Credential Use and Storage

Credential	Purpose	Storage
User TLS client key	Authenticates Hypori Client when establishing TLS connection to Hypori server	iOS Keychain
Server account password	Authenticates user to Hypori server	iOS Keychain

6.1.4 FCS_TLSC_EXT.1

The Hypori Client relies on the platform for TLS protection of communication with the Hypori server. This includes providing the TLS client certificate to authenticate the client to the server. The Hypori server authenticates a Hypori Client. Different versions of the iOS operating systems support distinct sets of cipher suites, which are shown in Table 4. The column labeled Server shows the default Hypori server configuration, which may be changed to accommodate the other identified cipher suites.

Table 4 Cipher suite Support by iOS Version

	Cipher suite	**10.0**	**10.1**	**10.2**	**10.3**	**Server**
1.	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	Yes	Yes	Yes	Yes	Default
2.	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Yes	Yes	Yes	Yes	Default
3.	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	Yes	Yes	Yes	Yes	Default
4.	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Yes	Yes	Yes	Yes	Default
5.	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Yes	Yes	Yes	Yes	Default
6.	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Yes	Yes	Yes	Yes	Default
7.	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	Yes	Yes	Yes	Yes	Default
8.	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Yes	Yes	Yes	Yes	Default

For elliptic curve cipher suites, the Hypori Client relies on the platform for elliptic curves. The iOS platforms support NIST curves secp256r1, secp384r1, and secp521r1 and Supported Elliptic Curves Extension for TLS. No configuration is required by a Hypori Client user.

The Hypori Client establishes the reference identifier using the configured server host name. The Hypori Client validates the first CN and the subject alternative names against the configured reference identifier. It supports wildcards and IP addresses. Pinning is not supported in the client.

6.1.5 FCS_TLSC_EXT.2

The Hypori Client establishes the reference identifier using the configured server host name. The Hypori Client validates the first CN and the subject alternative names against the configured reference identifier. It supports wildcards and IP addresses. Pinning is not supported in the client.

6.1.6 FCS_TLSC_EXT.4

For elliptic curve cipher suites, the Hypori Client relies on the platform for elliptic curves. The iOS platforms support NIST curves secp256r1, secp384r1, and secp521r1 and Supported Elliptic Curves Extension for TLS. No configuration is required by a Hypori Client user.

6.2 User data protection

The Hypori Client uses the platform's permission mechanisms to inform the user of hardware and software resources the client accesses. The client presents the required permissions to the user for approval at first launch. A user initiates network connections to the Hypori server. In general, sensitive data resides on the Hypori server and is not stored on the Hypori Client. Sensitive data on the Hypori Client is limited to credentials, which the client stores as described in section 6.1. The client does not maintain Personally Identifiable Information (PII).

6.2.1 FDP_DAR_EXT.1

Hypori Client sensitive data consist of user TLS client key and server account password credentials. FCS_STO_EXT.1 Storage of Secrets specifies the platform's iOS keychain for protecting keys and credentials. In accordance with FCS_STO_EXT.1, the Hypori Client stores these credentials in the platform's iOS keychain as described in section 6.1.2.

The Hypori Client stores application account options and any cached configuration settings (such as the server's hostname, port, notification properties, and settings to control the client application's behavior for disconnecting, keyboard, access to phone features, and jailbreak checking) provided by the Hypori Server using the iOS Protected Until First User Authentication protection class.

6.2.2 FDP_NET_EXT.1

The Hypori Client relies on user-initiated network communication to connect to the Hypori Virtual Device. The Hypori Client uses remote-initiated network communication to check for notifications and display them to the user when the system is configured for push notifications. The Hypori Client uses application-initiated network communication to periodically check for notifications and display them to the user when the system is configured for notification polling.

6.2.3 FDP_DEC_EXT.1

At first launch, the Hypori Client presents to the user some of the permissions requested by the application that are needed to operate. A user can accept (or reject) the permissions, but rejecting permissions may cause apps on the Hypori Virtual Device to not function properly. Some permissions are requested by the application only as the feature is required on first use (such as microphone input). Table 5 shows the permissions required by the Hypori Client. Those marked with an '*' are prompted for when the Hypori Client is started for the first time.

Table 5: iOS Permissions

Permission	Description
Background Operation	Support background fetch and remote notification features (never prompted for – a user can change this using the iOS settings application).
Camera	Access the mobile device's camera. *
Location	Provide location for authentication and virtual device apps. *
Microphone	Provide audio input for virtual device apps.
Photo Library	Support camera app features.
Notifications	Support notification display features. *

Updates to the Hypori Client may automatically add additional capabilities within each group. A user can accept (or reject) new permissions to complete any update that includes permissions not in the list above.

A user initiates a network connection to the Hypori server by starting the Hypori Client and entering account information. After the Hypori Client connects to the Hypori server, the applications the user accesses run on the Hypori Device in the Hypori server, not on the mobile device. The Hypori Client does not listen on any ports for inbound connection requests. The Hypori Client interacts only with the Hypori server. When a Hypori Device application needs information from a server (such as a map server), the Hypori Device – not the Hypori Client – communicates with the server (which may be an internal, enterprise server).

The Hypori Client does not maintain PII. Hence, it does not transmit PII over any network.[13]

6.3 Identification and authentication

The Hypori Client uses iOS certificate validation services to authenticate the X.509 certificate the Hypori server presents as part of the establishing a TLS connection.

6.3.1 FIA_X509_EXT.1

The iOS platform performs certification path validation as part of the TLS service. The Hypori Client relies on the platform for TLS services and package updates. Hence, the platform checks extended key usage for Server Authentication, Client Authentication, and Code Signing purposes. The Hypori Client performs validation of the revocation status of certificates using OSCP capabilities provided by the platform. The Hypori Client does not perform email encryption, email signature verification, and Enrollment over Secure Transport. Consequently, no check is made for extended key usage Email Protection and CMC Registration Authority purposed.

[13] The Hypori Client does maintain user credentials. In particular, the Hypori Client transmits a user's account name and TLS client certificate when connecting to the Hypori Server. However, PP APP SW distinguishes credentials from PII.

The iOS platform performs certification path validation as part of the TLS service. The platform certificate path methodology to manage X.509 certificate trust evaluation is described in the following document:

> https://developer.apple.com/library/content/technotes/tn2232/_index.html#//apple_ref/doc/uid/DTS40012884-CH1-SECTRUSTEVALUATIONFUNDAMENTALS

6.3.2 FIA_X509_EXT.2

The Hypori Client presents the TLS client certificate and key to the Hypori server to authenticate a TLS connection. During account setup, the user identifies which certificate to present for each account. The user selects a certificate from the certificate store. The user can change the selection from Client Certificate under Connection on the Settings page. The TLS client certificate is an X.509 certificate.

The user stores a CA certificate for the server certificates in the platform's keychain during installation. (The user need not install a CA certificate when the CA is a platform trusted CA.) On iOS devices, the Hypori Client uses the iOS platform certificate path validation services with the CA certificate to validate the certificate presented by the Hypori server. The Hypori Client enables the iOS services to use OCSP to determine the revocation status and is configured to fail the connection if the certificate is revoked or the connection to the OCSP responder fails.

6.4 Security management

The Hypori Client maintains account configuration and account state information in an encrypted file in the application's sandbox and uses the iOS platform's cryptographic key store for managing the user's keys.

6.4.1 FMT_CFG_EXT.1

Hypori Client credentials consist of user TLS client key and server account password. The Hypori Client installer does not include a default client key or server account password. A user installs a TLS client certificate and private key from a certificate file using the platform's certificate services. A user's IT group provides the user with a server account password.

6.4.2 FMT_MEC_EXT.1

The Hypori Client uses the iOS Key Value Store internally to hold account credentials. In addition, configuration options for the Hypori Client are managed and stored remotely on the Hypori Server. The Hypori client cache's these settings policies associated with the account in the iOS Key Value Store and updates them on the device when they are changed by the server. The users of the Hypori Client do not have direct access to the account configuration file's contents which is protected and stored according to FDP_DAR_EXT.1.

6.4.3 FMT_SMF.1

For each account, the Hypori Client provides the capability to set the Hypori server IP address, Hypori server port, account name, and TLS client certificate (key). The Hypori Client can enable the Remember Password setting for each account. The operational guidance recommends that the user disable this functionality. The Hypori Client Remember Password setting can also be disabled by policies received from the Hypori server.

The Hypori Client does not require any configuration to use ports and protocols in a manner consistent with DoD Ports and Protocols guidance, including the DoD Ports Protocols Services Management (PPSM) Category Assurance List (CAL). The Hypori Client does not listen on any ports for inbound connection requests. The Hypori Client interacts only with the Hypori server. When a Hypori Device application needs information from a server (such as a map server), the Hypori Device – not the Hypori Client – communicates with the server (which may be an internal, enterprise server).

6.5 Privacy

The Hypori Client does not knowingly transmit PII over a network.

6.5.1 FPR_ANO_EXT.1

The Hypori Client does not knowingly transmit PII over a network.

6.6 Protection of the TSF

The Hypori Client uses security features and APIs that the platform provides. The client leverages iOS package management for secure installation and updates.

6.6.1 FPT_AEX_EXT.1

To enable ASLR and stack protection on the iOS Hypori Client, Intelligent Waves builds with the `-fPIE -pie` and the `-fstack-protector-strong` flags. Hypori Client does not invoke `mmap` or `mprotect` from the iOS SDK.

6.6.2 FPT_API_EXT.1

The Hypori Client uses the iOS APIs listed in section 9 Appendix: iOS APIs.

6.6.3 FPT_LIB_EXT.1

The Hypori Client package includes only the third-party libraries listed in the security functional requirement.

6.6.4 FPT_TUD_EXT.1

Intelligent Waves distributes the Hypori Client as an IPA file for iOS devices. A user may obtain the installation package through Apple App Store or the enterprise IT group of the user. A user obtains Hypori Client updates using the platform's update mechanism or from the user's IT group. Intelligent Waves signs the installation package and Apple re-signs it. On iOS devices, iOS will only install a package from the Apple App Store if it has a valid signature from both Apple and the app developer.

The Hypori Client is signed with a unique certificate. It can be delivered via the Apple App Store, MDM, or other enterprise app stores. The certificate information:

```
X.509, CN=Joe Smith, OU=Development, O=Intelligent Waves LLC, C=US
      [certificate is valid from 11/06/17 14:53 PM to 10/31/42 15:53 PM]
      [CertPath not validated: Path does not chain with any of the trust
anchors]
```

A user can see the current version of the Hypori Client by checking the footer information on all screens.

Intelligent Waves provides customers with timely updates. A customer chooses their preferred communication. The Intelligent Waves Support Department will notify customers of updates using each customer's preferred communication mechanism. Application changes may be pushed to end users via the Apple App Store like any other application or via an enterprise application store internal to a customer. Typical delivery times for security updates are 5 to 10 business days.

Intelligent Waves maintains a Security Portal online. Every customer is registered with the Support Portal. Intelligent Waves notifies each customer of a new security report on the Support portal using the customers preferred communication mechanism. Intelligent Waves secures the Support Portal via SSL and user authentication. Each customer contact must log in with their specific credentials in order to see the security reports.

6.7 Trusted path/channels

The Hypori Client uses TLS 1.2 for all communication with Hypori server.

6.7.1 FTP_DIT_EXT.1

The Hypori server is the only trusted IT product the Hypori Client communicates with. For all communication with the Hypori server, the Hypori Client connects to the server using TLS 1.2 provided by the platform. In addition, the server may be configured to request additional authentication parameters such as:

- passwords associated with the client certificate and an LDAP account.
- an RSA SecurID token’s generated code.

Transmission of these values to the server are protected by the TLS encrypted communications.

7. Protection Profile Claims

This ST conforms to the *Protection Profile for Application Software,* Version 1.2, 22 April 2016 (PP APP SW)

As explained in Section 3, Security Problem Definition, the Security Problem Definition of the PP APP SW has been included by reference into this ST.

As explained in Section 4, Security Objectives, the Security Objectives of the PP APP SW have been included by reference into this ST.

The following table identifies all the security functional requirements in this ST. Each SFR is reproduced from the PP APP SW and operations completed as appropriate.

Table 6 SFR Protection Profile Sources

Requirement Class	Requirement Component	Source
FCS: Cryptographic support	FCS_CKM_EXT.1 Cryptographic Key Generation Services	PP APP SW
	FCS_RBG_EXT.1 Random Bit Generation Services	PP APP SW
	FCS_STO_EXT.1 Storage of Credentials	PP APP SW
	FCS_TLSC_EXT.1 TLS Client	PP APP SW
	FCS_TLSC_EXT.2 TLS Client	PP APP SW
	FCS_TLSC_EXT.4 TLS Client	PP APP SW
FDP: User data protection	FDP_DAR_EXT.1 Encryption of Sensitive Application Data	PP APP SW
	FDP_DEC_EXT.1 Access to Platform Resources	PP APP SW
	FDP_NET_EXT.1 Network Communications	PP APP SW
FIA: Identification and authentication	FIA_X509_EXT.1 X.509 Certificate Validation	PP APP SW
	FIA_X509_EXT.2 X.509 Certificate Authentication	PP APP SW
FMT: Security management	FMT_CFG_EXT.1 Secure by Default Configuration	PP APP SW
	FMT_MEC_EXT.1 Supported Configuration Mechanism	PP APP SW
	FMT_SMF.1 Specification of Management Functions	PP APP SW
FPR: Privacy	FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information	PP APP SW
FPT: Protection of the TSF	FPT_AEX_EXT.1 AntiExploitation Capabilities	PP APP SW
	FPT_API_EXT.1.1 Use of Supported Services and APIs	PP APP SW
	FPT_LIB_EXT.1 Use of Third Party Libraries	PP APP SW
	FPT_TUD_EXT.1 Integrity for Installation and Update	PP APP SW
FTP: Trusted path/channels	FTP_DIT_EXT.1 Protection of Data in Transit	PP APP SW

8. Rationale

This security target includes by reference the PP APP SW Security Problem Definition, Security Objectives, and Security Assurance Requirements. The security target makes no additions to the PP APP SW assumptions. PP APP SW security functional requirements have been reproduced with the PP APP SW operations completed. Operations on the security requirements follow PP APP SW application notes and assurance activities. Consequently, PP APP SW rationale applies but is incomplete. The TOE Summary Specification rationale below serves to complete the rationale required for the security target.

8.1 Dependency Rationale

PP APP SW includes mandatory, selection-based, optional, selection based, and objective requirements. The ST includes all mandatory requirements as well as all selection-based requirements that apply to selections made in the ST. The ST includes one optional elements (FCS_TLSC_EXT.2.1 and FCS_TLSC_EXT.4.1) and no objective requirements.

Table 7 presents rationale demonstrating that this ST includes all applicable selection-base requirements. The requirements column lists all PP APP SW security functional requirements. SFRs listed in italics are not included in this ST. The type of requirement is identified by M, Op, SB, and Ob, which stand for mandatory, optional, selection-based, and objective, respectively. The table lists dependencies of each requirement together with the condition that establishes the dependency. Finally, the table includes whether each dependency is satisfied or not applicable along with a brief rationale.

Table 7 Rationale for Selection-Based Requirements

Requirement	Type	Dependency	Condition	Satisfied	Rationale
FCS_RBG_EXT.1	M	FCS_RBG_EXT.2	Select implement DRBG	Not applicable	Not selected
FCS_RBG_EXT.2	*SB*	*None*	*None*	*Not applicable*	
FCS_STO_EXT.1	M	FCS_COP.1(1)	Select implement secure store	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_CKM_EXT.1	Unconditional	Satisfied	Included in ST
FCS_TLSC_EXT.4	SB	FCS_TLSC_EXT.4.1	Select elliptic curve cipher suites	Satisfied	Included in ST
FCS_TLSC_EXT.1	SB	FCS_CKM.1	Select implement TLS	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_CKM.2	Select implement TLS	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_COP.1(1)	Select implement TLS	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_COP.1(2)	Select implement TLS	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_COP.1(3)	Select implement TLS	Not applicable	Not selected
FCS_TLSC_EXT.1	SB	FCS_COP.1(4)	Select implement	Not applicable	Not selected

Requirement	Type	Dependency	Condition	Satisfied	Rationale
			TLS		
FCS_TLSC_EXT.1	SB	FIA_X509_EXT.1	Unconditional	Satisfied	Included in ST
FCS_TLSC_EXT.2.1	Op	FIA_X509_EXT.2	Unconditional	Satisfied	Included in ST
FCS_CKM_EXT.1	SB	FCS_CKM.1	Select invoke or implement key generation	Not applicable	Not selected
FCS_CKM.1	*Op*	*None*	*None*	*Not applicable*	
FCS_CKM.2	*Op*	*None*	*None*	*Not applicable*	
FCS_COP.1(1)	*Op*	*None*	*None*	*Not applicable*	
FCS_COP.1(2)	*Op*	*None*	*None*	*Not applicable*	
FCS_COP.1(3)	*Op*	*None*	*None*	*Not applicable*	
FCS_COP.1(4)	*Op*	*None*	*None*	*Not applicable*	
FCS_DTLS_EXT.1	*Op*	*FCS_TLSC_EXT.1*	*Unconditional*	*Not applicable*	*SFR not in ST*
FCS_HTTPS_EXT.1	*Op*	*FCS_TLSC_EXT.1*	*Unconditional*	*Not applicable*	*SFR not in ST*
FDP_DEC_EXT.1	M	None	None	Not applicable	
FDP_DAR_EXT.1	M	File Encryption EP	Select implement encrypt sensitive data	Not applicable	Not selected
FDP_NET_EXT.1	M	None	None	Not applicable	
FIA_X509_EXT.1	SB	None	None	Not applicable	
FIA_X509_EXT.2	SB	None	None	Not applicable	
FMT_MEC_EXT.1.1	M	None	None	Not applicable	
FMT_CFG_EXT.1	M	None	None	Not applicable	
FMT_SMF.1	M	None	None	Not applicable	
FPR.ANO.EXT.1	M	None	None	Not applicable	
FPT_API_EXT.1.1	M	None	None	Not applicable	
FPT_API_EXT.1.2	*Ob*	*None*	*None*	*Not applicable*	
FPT_AEX_EXT.1	M	None	None	Not applicable	
FPT_TUD_EXT.	M	None	None	Not applicable	
FPT_LIB_EXT.1	M	None	None	Not applicable	
FPT_IDV_EXT.1	*Ob*	*None*	*None*	*Not applicable*	
FTP_DIT_EXT.1	M	FCS_HTTPS_EXT.1	Select HTTPS	Not applicable	Not selected
FTP_DIT_EXT.1	M	FCS_TLSC_EXT.1	Select TLS	Satisfied	Included in ST
FTP_DIT_EXT.1	M	FCS_DTLS_EXT.1	Select DTLS	Not applicable	Not selected

8.2 TOE Summary Specification Rationale

Each subsection in Section 6, the TOE Summary Specification, describes a security function of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding security function. The security functions work together to satisfy all of the security functional requirements. Furthermore, all of the security functions are necessary in order for the TSF to provide the required security functionality.

This section in conjunction with section 6 TOE Summary Specification provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions works together to provide all of the security requirements. The security functions described in the TOE summary specification are all necessary for the required security functionality in the TSF. Table 8 demonstrates the relationship between security requirements and security functions.

Table 8 Security Functions vs. Requirements Mapping

	Cryptographic support	**User data protection**	**Identification and authentication**	**Security management**	**Privacy**	**Protection of the TSF**	**Trusted path/channels**
FCS_CKM_EXT.1	X						
FCS_RBG_EXT.1	X						
FCS_STO_EXT.1	X						
FCS_TLSC_EXT.1	X						
FCS_TLSC_EXT.2	X						
FCS_TLSC_EXT.4	X						
FDP_DAR_EXT.1		X					
FDP_NET_EXT.1		X					
FDP_DEC_EXT.1		X					
FIA_X509_EXT.1			X				
FIA_X509_EXT.2			X				
FMT_CFG_EXT.1				X			
FMT_MEC_EXT.1				X			
FMT_SMF.1				X			
FPR_ANO_EXT.1					X		
FPT_AEX_EXT.1						X	
FPT_API_EXT.1						X	
FPT_LIB_EXT.1						X	
FPT_TUD_EXT.1						X	
FTP_DIT_EXT.1							X

9. Appendix: iOS APIs

The Hypori Client uses the following iOS APIs:

1. AVAudioSession
2. AVCaptureDevice
3. AVCaptureDeviceInput
4. AVCaptureMetadataOutput
5. AVCaptureSession
6. AVCaptureStillImageOutput
7. AVCaptureVideoDataOutput
8. AVCaptureVideoPreviewLayer
9. AudioComponentFindNext
10. AudioComponentInstanceDispose
11. AudioComponentInstanceNew
12. AudioOutputUnitStart
13. AudioOutputUnitStop
14. AudioUnitInitialize
15. AudioUnitRender
16. AudioUnitSetProperty
17. AudioUnitUninitialize
18. CACurrentMediaTime
19. CBCentralManager
20. CBUUID
21. CC_SHA1
22. CC_SHA256
23. CFArrayGetCount
24. CFArrayGetValueAtIndex
25. CFAutorelease
26. CFCopyDescription
27. CFDataGetTypeID
28. CFDictionaryGetValue
29. CFGetTypeID
30. CFPropertyListCreateDeepCopy
31. CFRelease
32. CFRetain
33. CFRunLoopGetCurrent
34. CFUUIDCreate
35. CFUUIDCreateFromUUIDBytes
36. CFUUIDCreateString
37. CFUUIDGetUUIDBytes
38. CGAffineTransformMakeRotation
39. CGAffineTransformMakeTranslation
40. CGAffineTransformRotate

41. CGAffineTransformScale
42. CGAffineTransformTranslate
43. CGBitmapContextCreate
44. CGBitmapContextCreateImage
45. CGColorSpaceCreateDeviceRGB
46. CGColorSpaceRelease
47. CGContextAddEllipseInRect
48. CGContextConcatCTM
49. CGContextDrawImage
50. CGContextDrawPath
51. CGContextFillPath
52. CGContextRelease
53. CGContextRotateCTM
54. CGContextSetAllowsAntialiasing
55. CGContextSetFillColorWithColor
56. CGContextSetInterpolationQuality
57. CGContextSetLineWidth
58. CGContextSetStrokeColorWithColor
59. CGContextTranslateCTM
60. CGDataProviderCreateWithData
61. CGDataProviderRelease
62. CGImageCreate
63. CGImageDestinationAddImage
64. CGImageDestinationCreateWithData
65. CGImageDestinationFinalize
66. CGImageGetAlphaInfo
67. CGImageGetBitmapInfo
68. CGImageGetBitsPerComponent
69. CGImageGetColorSpace
70. CGImageGetHeight
71. CGImageGetWidth
72. CGImageRelease
73. CGRectApplyAffineTransform
74. CGRectContainsPoint
75. CGRectGetMinY
76. CGRectInset
77. CGRectIntegral
78. CLLocation
79. CLLocationManager
80. CMBlockBufferAppendBufferReference
81. CMBlockBufferCopyDataBytes
82. CMBlockBufferCreateWithBufferReference
83. CMBlockBufferCreateWithMemoryBlock
84. CMBlockBufferGetDataLength

85. CMBlockBufferReplaceDataBytes
86. CMCopyDictionaryOfAttachments
87. CMMotionManager
88. CMSampleBufferCreate
89. CMSampleBufferGetDataBuffer
90. CMSampleBufferGetFormatDescription
91. CMSampleBufferGetImageBuffer
92. CMSampleBufferGetSampleAttachmentsArray
93. CMTimeMake
94. CMVideoFormatDescriptionCreateFromH264ParameterSets
95. CMVideoFormatDescriptionGetDimensions
96. CMVideoFormatDescriptionGetH264ParameterSetAtIndex
97. CTCallCenter
98. CVOpenGLESTextureCacheCreate
99. CVOpenGLESTextureCacheCreateTextureFromImage
100. CVOpenGLESTextureCacheFlush
101. CVOpenGLESTextureGetName
102. CVOpenGLESTextureGetTarget
103. CVPixelBufferGetBaseAddress
104. CVPixelBufferGetBytesPerRow
105. CVPixelBufferGetHeight
106. CVPixelBufferGetWidth
107. CVPixelBufferLockBaseAddress
108. CVPixelBufferUnlockBaseAddress
109. EAGLContext
110. GLKMatrix3Invert
111. GLKView
112. LAContext
113. MFMailComposeViewController
114. NSArray
115. NSAssertionHandler
116. NSAttributedString
117. NSBundle
118. NSCache
119. NSCharacterSet
120. NSClassFromString
121. NSCondition
122. NSData
123. NSDate
124. NSDateFormatter
125. NSDictionary
126. NSError
127. NSException
128. NSFileCoordinator

129. NSFileManager
130. NSHTTPURLResponse
131. NSIndexPath
132. NSIndexSet
133. NSInvalidArgumentException
134. NSJSONSerialization
135. NSKeyedArchiver
136. NSKeyedUnarchiver
137. NSLayoutConstraint
138. NSLocale
139. NSLock
140. NSLog
141. NSMapTable
142. NSMutableArray
143. NSMutableData
144. NSMutableDictionary
145. NSMutableIndexSet
146. NSMutableParagraphStyle
147. NSMutableSet
148. NSMutableString
149. NSMutableURLRequest
150. NSNotificationCenter
151. NSNull
152. NSNumber
153. NSObject
154. NSOperationQueue
155. NSPredicate
156. NSPropertyListSerialization
157. NSRecursiveLock
158. NSScanner
159. NSSelectorFromString
160. NSSet
161. NSString
162. NSStringFromCGRect
163. NSStringFromClass
164. NSThread
165. NSTimeZone
166. NSTimer
167. NSURL
168. NSURLCredential
169. NSURLRequest
170. NSURLSession
171. NSURLSessionConfiguration
172. NSUUID

173. NSUserDefaults
174. OSAtomicAdd32Barrier
175. SCNetworkReachabilityCreateWithAddress
176. SCNetworkReachabilityCreateWithName
177. SCNetworkReachabilityGetFlags
178. SCNetworkReachabilityScheduleWithRunLoop
179. SCNetworkReachabilitySetCallback
180. SCNetworkReachabilityUnscheduleFromRunLoop
181. SSLClose
182. SSLCopyPeerTrust
183. SSLCreateContext
184. SSLGetNumberSupportedCiphers
185. SSLGetSupportedCiphers
186. SSLHandshake
187. SSLRead
188. SSLSetCertificate
189. SSLSetConnection
190. SSLSetEnabledCiphers
191. SSLSetIOFuncs
192. SSLSetPeerDomainName
193. SSLSetProtocolVersionMax
194. SSLSetProtocolVersionMin
195. SSLSetSessionOption
196. SSLWrite
197. SecCertificateCopyData
198. SecCertificateCopySubjectSummary
199. SecCertificateCreateWithData
200. SecIdentityCopyCertificate
201. SecIdentityCopyPrivateKey
202. SecItemAdd
203. SecItemCopyMatching
204. SecItemDelete
205. SecItemUpdate
206. SecKeyRawSign
207. SecPKCS12Import
208. SecTrustCopyProperties
209. SecTrustCopyResult
210. SecTrustEvaluate
211. SecTrustSetAnchorCertificates
212. SecTrustSetAnchorCertificatesOnly
213. UIAlertAction
214. UIAlertController
215. UIAlertView
216. UIApplication

217. UIApplicationMain
218. UIBarButtonItem
219. UIButton
220. UICollectionViewCell
221. UICollectionViewController
222. UICollectionViewFlowLayout
223. UIColor
224. UIDevice
225. UIDocumentMenuViewController
226. UIFont
227. UIGraphicsBeginImageContextWithOptions
228. UIGraphicsEndImageContext
229. UIGraphicsGetCurrentContext
230. UIGraphicsGetImageFromCurrentImageContext
231. UIImage
232. UIImageJPEGRepresentation
233. UIImagePNGRepresentation
234. UIImagePickerController
235. UIImageView
236. UILabel
237. UILocalNotification
238. UILongPressGestureRecognizer
239. UIMenuController
240. UIMenuItem
241. UINavigationController
242. UINib
243. UIPanGestureRecognizer
244. UIPasteboard
245. UIRectFillUsingBlendMode
246. UIResponder
247. UIScreen
248. UIScrollView
249. UISlider
250. UIStoryboard
251. UISwitch
252. UITableView
253. UITableViewCell
254. UITableViewController
255. UITapGestureRecognizer
256. UITextField
257. UITextView
258. UIUserNotificationSettings
259. UIView
260. UIViewController

261. UIWebView
262. UIWindow
263. UNMutableNotificationContent
264. UNNotificationRequest
265. UNNotificationServiceExtension
266. UNNotificationSound
267. UNUserNotificationCenter
268. VTCompressionSessionCreate
269. VTCompressionSessionEncodeFrame
270. VTCompressionSessionInvalidate
271. VTDecompressionSessionCanAcceptFormatDescription
272. VTDecompressionSessionCreate
273. VTDecompressionSessionDecodeFrame
274. VTDecompressionSessionInvalidate
275. VTSessionSetProperty
276. abort
277. arc4random
278. atan2
279. atoi
280. bzero
281. calloc
282. class_getInstanceMethod
283. close
284. connect
285. dispatch_after
286. dispatch_async
287. dispatch_get_global_queue
288. dispatch_once
289. dispatch_queue_create
290. dispatch_sync
291. dispatch_time
292. exit
293. fclose
294. fcntl
295. fflush
296. fopen
297. fprintf
298. fputs
299. free
300. fwrite
301. gai_strerror
302. getaddrinfo
303. getenv
304. getpid

305. getsockopt
306. glActiveTexture
307. glAttachShader
308. glBindTexture
309. glBlendFunc
310. glClear
311. glClearColor
312. glCompileShader
313. glCreateProgram
314. glCreateShader
315. glDeleteProgram
316. glDeleteShader
317. glDeleteTextures
318. glDisable
319. glDisableVertexAttribArray
320. glDrawElements
321. glEnable
322. glEnableVertexAttribArray
323. glGenTextures
324. glGetAttribLocation
325. glGetProgramiv
326. glGetUniformLocation
327. glLinkProgram
328. glShaderSource
329. glTexImage2D
330. glTexParameterf
331. glTexParameteri
332. glTexSubImage2D
333. glUniform1f
334. glUniform1i
335. glUniformMatrix4fv
336. glUseProgram
337. glVertexAttribPointer
338. glViewport
339. log10
340. mach_error_string
341. malloc
342. memcpy
343. memmove
344. memset
345. method_getImplementation
346. open
347. os_activity_initiate
348. pipe

349. poll
350. pow
351. printf
352. pthread_mutex_destroy
353. pthread_mutex_init
354. pthread_mutex_lock
355. pthread_mutex_unlock
356. puts
357. realloc
358. recv
359. sched_yield
360. select
361. send
362. setsockopt
363. shutdown
364. snprintf
365. socket
366. strcmp
367. strdup
368. strerror
369. strlen
370. strtoul
371. sys_errlist
372. system
373. uname
374. usleep
375. vfprintf
376. vm_allocate
377. vm_deallocate
378. vm_remap
379. write