Hypori Halo Client (iOS) 4.3

Security Target

Version 1.0
February 15, 2024

Prepared for:
Hypori, Inc.
1801 Robert Fulton Drive, Suite 340
Reston, VA 20191

Prepared by:

Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive; Columbia, Maryland 21046

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 Halo Client (iOS) 4.3 component of the Hypori Platform provided by Hypori, 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 – Hypori Halo Client (iOS) 4.3 Security Target

ST Version – Version 1.0

ST Date – February 15, 2024

TOE Identification – Hypori Halo Client (iOS) 4.3

TOE Developer – Hypori, Inc.

Evaluation Sponsor – Hypori, Inc.

CC Identification – *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017*

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 1.4, 2021-10-07 [PP_APP_v1.4].

The following NIAP Technical Decisions apply to the security target or the evaluation assurance activities.

- TD0780: FIA_X509_EXT.1 Test 4 Clarification
- TD0756: Update for platform-provided full disk encryption
- TD0743: FTP_DIT_EXT.1.1 Selection exclusivity
- TD0719: ECD for PP APP V1.3 and 1.4
- TD0717: Format changes for PP_APP_V1.4
- TD0664: Testing activity for FPT_TUD_EXT.2.2

The following NIAP Technical Decisions are listed on the NIAP website, but are not applicable to this evaluation, for the reasons given:

- TD0798: Static Memory Mapping Exceptions
  - The Security Target does not include any list of explicit exceptions in FPT_AEX_EXT.1.1
- TD0747: Configuration Storage Option for Android

  - The TOE is not installed on an Android platform.
- TD0736: Number of elements for iterations of FCS_HTTPS_EXT.1
  - The Security Target does not include FCS_HTTPS_EXT.1/Server.
- TD0650: Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4
  - The Security Target does not claim conformance to the PP-Module for VPN Clients.
- TD0628: Addition of Container Image to Package Format
  - The TOE is not a container image.

Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.

- Part 2 Extended

Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, April 2017.

- 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.
- 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_v1.4] 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.

Table 1 Abbreviations

API	Application Programming Interface
APNs	Apple Push Notification service
App	Software application
ASLR	Address Space Layout Randomization
CC	Common Criteria
CEM	Common Evaluation Methodology
HTTP	Hypertext Transfer Protocol
IPA	iOS App Store Package file
MDM	Mobile Device Management
OS	Operating System
PII	Personally Identifiable Information
PLIST	Property List file
PP	Protection Profile
PP_APP_v1.4	Protection Profile for Application Software
SAR	Security assurance Requirement
SCM	Source Code Management
SFR	Security functional requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality

TSS	TOE Summary Specification

2. TOE Description

After a brief overview of the Hypori Halo Client (iOS) product, this section describes its Hypori Halo Client 4.3 (iOS) 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 Halo Client (iOS) platform, end users running a Hypori Halo Client (iOS) 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 Halo Client (iOS). The Hypori iOS application provides secure access to the remote Android virtual device and brokers access between the mobile device and the applications executing in the virtual device on a Hypori server. The client applications are indifferent to the version of iOS executing on the physical device.

The following diagram illustrates the user data connection for the TOE.

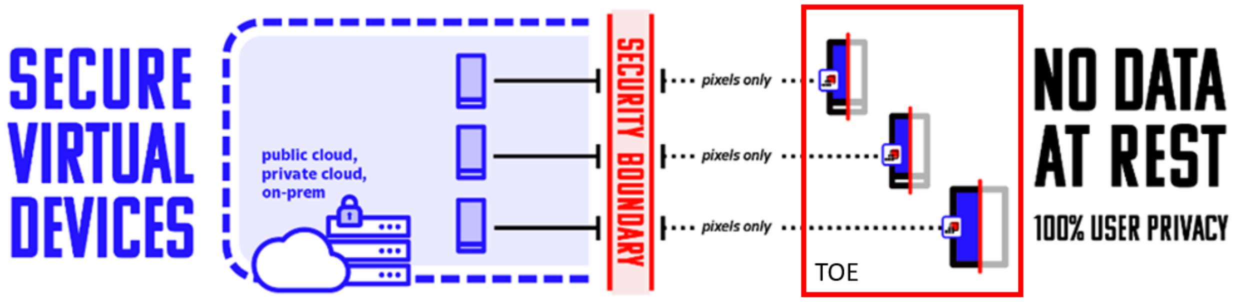

Figure 1 Hypori Halo Client (iOS) User Data Flow

The user's physical device is a "window" to their virtualized smartphone residing in the cloud or on-premises. The Hypori Halo app captures touch and sensor data from any iOS device. Encrypted pixels are transmitted to and from the physical device to access the enterprise applications in the cloud.

Hypori Halo delivers secure access to enterprise apps and data via a separate, secure virtual device from a smartphone or tablet. It uses cloud-based, zero-trust architecture, guarantees no data on the device, and 100% separation of personal and enterprise data.

The platform device which hosts the Hypori Halo application is not included in the TOE boundary.

The following diagram shows the Hypori system, including its components and networks. Unlike many software solutions, some of the Hypori servers are installed on virtual servers while others are installed on physical servers.

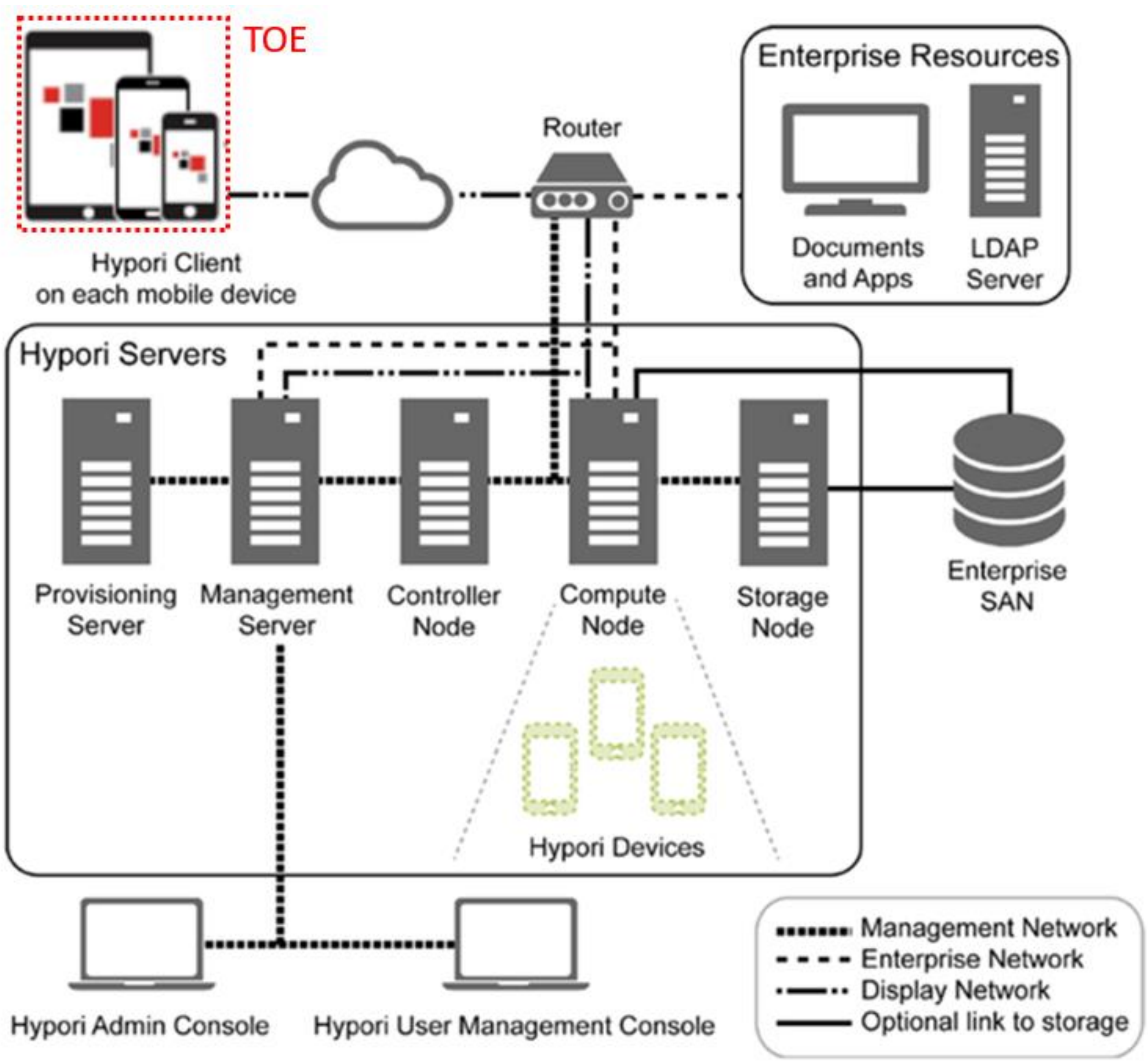

Figure 2 Hypori Solution

The Hypori solution includes the following components:

- **Hypori Halo Client:** This is an iOS-based application that installs on the end user's mobile device and communicates with the Hypori Virtual Device on the server through secure encrypted protocols. The platform device is not included in the TOE boundary.
- **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.
- **Hypori User Management Console:** A web application to manage users within a designated Hypori Halo environment.

The Hypori Virtual Device, Servers, User Management Console, and Admin Console are not included in the evaluation.

2.2 TOE Overview

The TOE is the iOS-based Hypori Halo Client. The following diagram shows how the TOE interacts with a Hypori Device running applications on a Hypori Server. The Hypori Halo Client is an application that communicates only with a Hypori Virtual Device on a Hypori Server and not with other servers or applications.

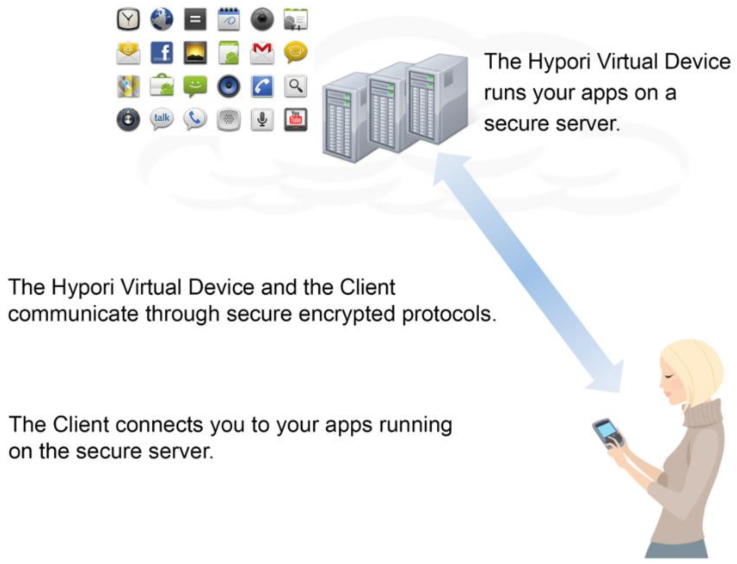
The Hypori Virtual Device runs your apps on a secure server.
The Hypori Virtual Device and the Client communicate through secure encrypted protocols.
The Client connects you to your apps running on the secure server.

Figure 3 Hypori Halo Client Communication with a Hypori Virtual Device on a Hypori Server

2.3 TOE Architecture

This section describes the TOE architecture including physical and logical boundaries. Figure 4 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.

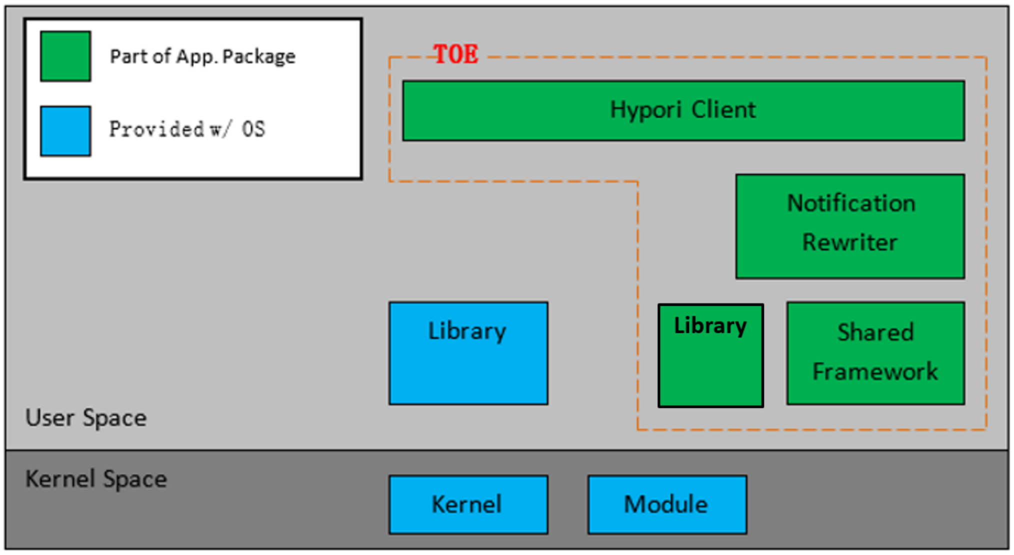

Figure 4 TOE Boundary for iOS Devices

2.3.1 Physical Boundaries

The TOE consists of a Hypori Halo Client software application available in the Hypori Halo Client installation package. The Hypori Halo Client is an iOS-based application that only communicates with the Hypori Virtual Device on the Hypori server. The Hypori server, the hardware mobile device, and 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 is supported on iOS releases iOS 15 and iOS 16.

2.3.1.2 Hardware Requirements

The TOE imposes no hardware requirements beyond the 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 Virtual Device on the Hypori server using TLS. The TOE uses cryptographic services provided by the platform. The TOE stores credentials and certificates for mutual authentication in the platform's key chain.

2.3.2.2 User data protection

The TOE informs a user of hardware and software resources the TOE accesses.

The user initiates a secure network connection to the Hypori Virtual Device on the Hypori server using the TOE. In general, sensitive data resides on the Hypori server and not the Hypori Halo Client, although the client does store credentials as per section 2.3.2.1.

2.3.2.3 Identification and Authentication

The TOE supports X.509 certificate validation as part of establishing TLS connections. The TOE relies on platform-provided functionality to support certificate validity checking, including the checking of certificate revocation status using OCSP. If the validity status of a certificate cannot be determined, the certificate will not be accepted.

2.3.2.4 Security management

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

2.3.2.5 Privacy

The TOE does not 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. The TOE package includes only those third-party libraries necessary for its intended operation.

2.3.2.7 Trusted path/channels

The TOE invokes the platform-provided functionality to encrypt all transmitted data using TLS 1.2 for all communication with the Hypori Virtual Device on the Hypori server.

2.4 TOE Documentation

Hypori provides the following product documentation in support of the installation and secure use of the TOE:

- Hypori User Guide Common Criteria Configuration and Operation, Version 4.3
- Hypori Halo Administrator Guide, Version 1.18

3. Security Problem Definition

This security target includes by reference the Security Problem Definition from the [PP_APP_v1.4]. 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_v1.4] 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 Halo Client is an iOS application running on a mobile device. As such, the [PP_APP_v1.4] 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_v1.4]. The [PP_APP_v1.4] 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_v1.4] 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_v1.4] security objectives are suitable for the Hypori Halo Client (iOS) 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 1.4, 7 October 2021 [PP_APP_v1.4]. As a result, any selection, assignment, or refinement operations already performed by that PP on the claimed SFRs are not identified here (i.e., they are not formatted in accordance with the conventions specified in section 1.3 of this ST). Formatting conventions are only applied on SFR text that was chosen at the ST author's discretion.

The security assurance requirements are the set of SARs specified in [PP_APP_v1.4].

5.1 Extended Requirements

All of the extended requirements in this ST have been drawn from the [PP_APP_v1.4]. The [PP_APP_v1.4] defines the following extended SFRs. Since these SFRs are not redefined in this ST, readers should consult [PP_APP_v1.4] 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
- 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_IDV_EXT.1 Software Identification and Versions
- FPT_LIB_EXT.1 Use of Third Party Libraries
- FPT_TUD_EXT.1 Integrity for Installation and Update
- FPT_TUD_EXT.2 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 Halo Client TOE.

Table 2 TOE Security Functional Components

Requirement Class	Requirement Component
FCS: Cryptographic support	FCS_CKM_EXT.1 Cryptographic Key Generation Services
	FCS_CKM.1/AK Cryptographic Asymmetric Key Generation
	FCS_CKM.2 Cryptographic Key Establishment

Requirement Class	Requirement Component
	FCS_RBG_EXT.1 Random Bit Generation Services
	FCS_STO_EXT.1 Storage of Credentials
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_IDV_EXT.1 Software Identification and Versions
	FPT_LIB_EXT.1 Use of Third Party Libraries
	FPT_TUD_EXT.1 Integrity for Installation and Update
	FPT_TUD_EXT.2 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 [***invoke platform-provided functionality for asymmetric key generation***].

5.2.1.2 Cryptographic Asymmetric Key Generation (FCS_CKM.1/AK)

FCS_CKM.1.1/AK[2] The application shall [

- ***invoke platform-provided functionality***

] ***to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm***

- ***[RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3"],***
- ***[ECC schemes] using ["NIST curves" P-384 and [P-256]]that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4],***

5.2.1.3 Cryptographic Key Establishment (FCS_CKM.2)

FCS_CKM.2.1 The application shall [***invoke platform-provided functionality***] to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:
[

[1] Modified per TD0717

[2] Modified per TD0717

- ***[RSA-based key establishment schemes] that meets the following: [NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"] ,***
- ***[Elliptic curve-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"] ,***

].

5.2.1.4 Random Bit Generation Services (FCS_RBG_EXT.1)

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

5.2.1.5 Storage of Credentials (FCS_STO_EXT.1)

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

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 The application shall [***protect sensitive data in accordance with FCS_STO_EXT.1***] in non-volatile 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,***
- ***[Fingerprint scanner]***].

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 Virtual Device on the Hypori server, connecting for help request Webpages hosted in AWS],***
- ***[application-initiated communication for polling the Hypori server for notifications]***

].

5.2.3 Identification and authentication (FIA)

5.2.3.1 X.509 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1 The application shall [***invoke 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, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.

- The application shall validate that any CA certificate includes caSigning purpose in the key usage field
- The application shall validate the revocation status of the certificate using [***OCSP as specified in RFC 6960***].
- The application 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.
    - 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.[3]
    - 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.[4]

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.

5.2.3.2 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***].

5.2.4 Security Management (FMT)

5.2.4.1 Secure by Default Configuration (FMT_CFG_EXT.1)

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

FMT_CFG_EXT.1.2 The application shall be configured by default with file permissions which protect the application binaries and data files from modification by normal unprivileged users.

5.2.4.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)

FMT_MEC_EXT.1.1 The application shall [

[3] 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.

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

- ***invoke the mechanisms recommended by the platform vendor for storing and setting configuration options***].

5.2.4.3 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions [***[***

- ***setting configuration options***
- ***applying configuration policies from the Hypori server***

]] .

5.2.5 Privacy

5.2.5.1 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***].

5.2.6 Protection of the TSF (FPT)

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

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

5.2.6.2 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 built with stack-based buffer overflow protection enabled.

5.2.6.3 Software Identification and Versions (FPT_IDV_EXT.1)

FPT_IDV_EXT.1.1 The application shall be versioned with [***[iOS version information and Hypori internal versioning scheme]***].

5.2.6.4 Use of Third Party Libraries (FPT_LIB_EXT.1)

FPT_LIB_EXT.1.1 The application shall be packaged with only [

- **PLCrashReporter-1.2/CrashReporter.framework v1.2**
- **ios-openssl v1.1e**
- **OCMock-3.4**
- **Opus Audio Codec v1.1**
- **protobuf v2.5**
- **spice-qtk v2.2**
- **grpc-swift v1.8.1**
- **sodk.xcframework 1.0**
- **swift-atomics v1.0.2**

- **swift-collections v1.0.3**
- **swift-log v1.4.4**
- **swift-nio v2.43.1**
- **swift-nio-extras v1.14.0**
- **swift-nio-http2 v1.23.0**
- **swift-nio-ssl v2.23.0**
- **swift-nio-transport-services v1.15.0**
- **SwiftProtobuf v1.20.2**
- **YubiKit v2.0.0 RC1**

].

5.2.6.5 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 [***provide the ability***] to query the current version of the application software.

FPT_TUD_EXT.1.3 The application shall not download, modify, replace or update its own binary code.

FPT_TUD_EXT.1.4 Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.

FPT_TUD_EXT.1.5 The application is distributed [***as an additional software package to the platform OS***].

5.2.6.6 Integrity for Installation and Update (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1 The application shall be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.2.2 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.2.3 The application installation package shall be digitally signed such that its platform can cryptographically verify them prior to installation.

5.2.7 Trusted path/channels (FTP)

5.2.7.1 Protection of Data in Transit (FTP_DIT_EXT.1)

FTP_DIT_EXT.1.1[5] The application shall [

- i***nvoke platform-provided functionality to encrypt all transmitted data with [TLS] for [communication with the virtual Hypori Device running applications on a Hypori Server]***] between itself and another trusted IT product.

5.3 TOE Security Assurance Requirements

The security assurance requirements in Table 3 are included in this ST by reference from the [PP_APP_v1.4].

Table 3 Assurance Components

Requirement Class	Requirement Component

[5] Modified per TD0743.

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

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_v1.4] apply to the TOE evaluation.

6. TOE Summary Specification

This chapter describes the security functions:

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

6.1 Cryptographic support

The Hypori Halo Client makes use of the platform for cryptographic services. The Hypori Halo Client uses platform TLS services for secure communication with the Hypori Virtual Device on the Hypori server, including mutual authentication. The client uses TLS client certificates and the RSA or Elliptic Curve key pairs 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.

The TOE relies on the platform to provide all of its cryptographic functionality. The following iOS evaluations are conformant to the Common Criteria for IT Security Evaluation (ISO Standard 15408) and are listed at the National Information Assurance Partnership (NIAP) Product Compliant List.

The TOE was tested on the following iOS v15.7.7 and iOS 16.5.1(C) platforms:

iOS15

Apple iOS 15: iPhones, Update from v15.1.0 to v15.7.1

Validation Report Number: CCEVS-VR-VID11237-2022

https://www.niap-ccevs.org/Product/Maint.cfm?AMID=1542&PID=11237

Table 4 TOE Version iOS 15.7.7

Device Name	Model	Processor	Architecture	iOS
iPhone SE	A1723	A9	ARMv8.0-A	15.7.7

CAVP Certificates

Below are the module (Mod) names used in the Table below.

KRN - Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1] (Kernel Space)

SEP - SEP Hardware v2.0

SKS - Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

USR - Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] (User Space)

Note that the Apple corecrypto Module is v12.0, but the SEP Hardware is v2.0.

The TOE OS supports RSA [SP800-56B] key establishment schemes as both a sender (e.g., when starting a TLS session) and a recipient (e.g., during a rekey triggered by the remote endpoint). 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.

Table 5 iOS 15 CAVP Certificates

SFR	Algorithm	NIST Standard	CAVP Certificate
FCS_CKM.1/AK FTP_DIT_EXT.1	ECDSA KeyGen and KeyVer P-256, P-384	FIPS 186-4	USR - A2788 KRN – A2797 SKS – A2848
FCS_CKM.2 FTP_DIT_EXT.1	ECC Key Establishment (KAS-ECC-SSC) P-256, P-384	SP800-56Ar3	USR - A2786 SKS – A2845
FCS_CKM.1/AK FTP_DIT_EXT.1	RSA KeyGen 2048, 3072, 4096	FIPS186-4	USR - A2786[6]
FCS_CKM.2 FTP_DIT_EXT.1	RSA Key Establishment 2048, 3072, 4096	SP800-56B	Vender Affirmation as per PL#5 Add2 v2.0
FTP_DIT_EXT.1	AES CBC encrypt, decrypt 128-bit, 256-bit	FIPS 197 SP800-38A	USR – A2783 KRN – A2793 SKS – A2842
FTP_DIT_EXT.1	AES GCM Encrypt, decrypt 128-bit, 256-bit	FIPS 197 SP800-38D	USR – A2787 KRN – A2796 SKS – A2847
FTP_DIT_EXT.1	SHS Byte-oriented mode SHA2-256	FIPS 180-4	USR – A2789 KRN – A2798 SKS – A2849
FTP_DIT_EXT.1	SHS Byte-oriented mode SHA-1, SHA2-384, SHA2-512	FIPS 180-4	USR – A2788 KRN – A2797 SKS – A2848
FTP_DIT_EXT.1	RSA SigGen PKCS 1.5 and PKCSPSS Modulo: 2048, 3072, 4096 Using SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR – A2788 KRN – A2797
FTP_DIT_EXT.1	RSA SigVer PKCS 1.5 and PKCSPSS	FIPS 186-4	USR – 2788 KRN – A2797

[6] The corresponding RSA keygen claim was not made in the platform ST, but the algorithm certificate includes the function being referenced.

SFR	Algorithm	NIST Standard	CAVP Certificate
	Modulo: 2048, 3072, 4096 Using SHA-1, SHA2-256, SHA2-384, SHA2-512		
FTP_DIT_EXT.1	ECDSA SigGen P-256, P-384, P-521 using SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR – A2788 KRN – A2797 SKS – A2848
FTP_DIT_EXT.1	ECDSA SigVer P-256, P-384, P-521 using SHA-1, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR – A2788 KRN – A2797 SKS – A2848
FTP_DIT_EXT.1	HMAC Byte-oriented mode HMAC-SHA2-256	FIPS 198-1	USR – A2789 KRN – A2798 SKS – A2849
FTP_DIT_EXT.1	HMAC Byte-oriented mode HMAC-SHA-1, HMAC-SHA2-384, HMAC-SHA2-512	FIPS 198-1	USR – A2788 KRN – A2797 SKS – A2848
FTP_DIT_EXT.1	CTR_DRBG(AES) AES-256	SP800-90A	USR – A2787 KRN – A2796
FTP_DIT_EXT.1	CTR_DRBG(AES) AES-256	SP800-90A	SEP – DRBG-2025

iOS16

Apple iOS 16: iPhones

Validation Report Number: CCEVS-VR-VID11349-2023

https://www.niap-ccevs.org/Product/Compliant.cfm?PID=11349

The NIAP evaluation was completed using Apple iOS Version 16.3.

Table 6 TOE Version iOS 16.5.1(C))

Device Name	Model	Processor	Architecture	iOS
iPhone X	A1865	A11 Bionic	ARMv8.2-A	16.5.1(C)

CAVP Certificates

USR - The user space software module implementation name for v13.0.

KRN - The kernel space software module name for v13.0

SKS-FW – The secure key store (SKS) firmware module implementation name for v13.0

SKS-HW - The secure key store (SKS) hardware module implementation name for v2.0

BC - The Broadcom core hardware module implementation name.

The TOE OS supports RSA (SP800-56B-Rev1) key establishment schemes as both a sender (e.g., when starting a TLS session) and a recipient (e.g., during a rekey triggered by the remote endpoint). 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.

Table 7 iOS 16 CAVP Certificates

SFR	Algorithm	NIST Standard	CAVP Certificate
FCS_CKM.1/AK FTP_DIT_EXT.1	ECDSA KeyGen and KeyVer P-256, P-384	FIPS 186-4	USR - A3428 KRN - A3686 SKS-FW - A4109
FCS_CKM.2 FTP_DIT_EXT.1	ECC Key Establishment (KAS-ECC-SSC) P-256, P-384	SP800-56Ar3	USR - A3426 SKS-FW - A4106
FCS_CKM.1/AK FTP_DIT_EXT.1	RSA KeyGen 2048, 3072, 4096	FIPS186-4	USR – A3426[7]
FCS_CKM.2 FTP_DIT_EXT.1	RSA Key Establishment 2048, 3072, 4096	SP800-56B	Vendor Affirmation as per PL#5 Add2 v2.0
FTP_DIT_EXT.1	AES CBC encrypt, decrypt 128-bit, 256-bit	FIPS 197 SP800-38A	USR - A3423 KRN - A3682 SKS-FW - A4103 SKS-HW – C319
FTP_DIT_EXT.1	AES GCM Encrypt, decrypt 128-bit, 256-bit	FIPS 197 SP800-38D	USR - A3427 KRN - A3685 SKS-FW - A4108
FTP_DIT_EXT.1	SHS Byte-oriented mode SHA2-256	FIPS 180-4	USR - A3429 KRN - A3687 SKS-FW - A4110
FTP_DIT_EXT.1	SHS Byte-oriented mode SHA-1, SHA2-384, SHA2-512	FIPS 180-4	USR - A3428 KRN - A3686 SKS-FW - A4109
FTP_DIT_EXT.1	RSA SigGen PKCS 1.5 and	FIPS 186-4	USR - A3428 KRN - A3686

[7] The corresponding RSA keygen claim was not made in the platform ST, but the algorithm certificate includes the function being referenced.

SFR	Algorithm	NIST Standard	CAVP Certificate
	PKCSPSS Modulo: 2048, 3072, 4096 Using SHA2-256, SHA2-384, SHA2-512		
FTP_DIT_EXT.1	RSA SigVer PKCS 1.5 and PKCSPSS Modulo: 2048, 3072, 4096 Using SHA-1, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR - A3428 KRN - A3686
FTP_DIT_EXT.1	ECDSA SigGen P-256, P-384, P-521 using SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR - A3428 KRN - A3686 SKS-FW - A4109
FTP_DIT_EXT.1	ECDSA SigVer P-256, P-384, P-521 using SHA-1, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4	USR - A3428 KRN - A3686 SKS-FW - A4109
FTP_DIT_EXT.1	HMAC Byte-oriented mode HMAC-SHA2-256	FIPS 198-1	USR - A3429 KRN - A3687 SKS-FW - A4110
FTP_DIT_EXT.1	HMAC Byte-oriented mode HMAC-SHA-1, HMAC-SHA2-384, HMAC-SHA2-512	FIPS 198-1	USR – A3428 KRN – A3686 SKS-FW – A4109
FTP_DIT_EXT.1	CTR_DRBG(AES) AES-256	SP800-90A	SKS-HW - DRBG 2014 USR - A3427 KRN - A3685

6.1.1 FCS_CKM_EXT.1

The Hypori Halo Client requires asymmetric key generation services for secure communication to the Hypori Virtual Device on the Hypori Server. The Hypori Halo Client invokes platform-provided functionality for asymmetric key generation. As part of installation, a user adds a TLS client certificate and the RSA or Elliptic Curve key pairs to the platform's key store. The platform generates all ephemeral TLS keys without direct Hypori Client action.

6.1.2 FCS_CKM.1/AK

The TOE relies on the platform generation of asymmetric cryptographic keys for the secure communication to the Hypori Virtual Device on the Hypori Server. For elliptic curve cipher suites, the Hypori Halo Client relies on the platform for elliptic curves.

The iOS 15 and 16 platforms call the Apple corecrypto Module libraries to create the ECC and RSA keys. The iOS platform supports NIST curves secp256r1, secp384r1 and RSA key sizes of 2048-bit, 3072-bit, 4096-bit and Supported Elliptic Curves Extension for TLS. No configuration is required by a Hypori Halo Client user.

The TOE calls the iOS API to create a dictionary from which the RSA and ECC cryptographic key pairs are defined. The same platform API is used for both RSA and ECC key generation.

6.1.3 FCS_CKM.2

The TOE relies on the platform generation of asymmetric cryptographic keys for the secure communication to the Hypori Virtual Device on the Hypori Server.

The application invokes platform-provided functionality to generate asymmetric cryptographic keys in accordance with the following cryptographic key generation algorithm:

- RSA schemes using cryptographic key sizes of 2048-bit, 3072-bit, and 4096-bit that meets the following standard: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3",
- ECC schemes using "NIST curves" P-256 and P-384 that meets the following standard: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4.

6.1.4 FCS_RBG_EXT.1

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

6.1.5 FCS_STO_EXT.1

Table 8 8 lists each Hypori Halo Client persistent credential along with how the client uses and stores each credential.

Table 8 Persistent Credential Use and Storage

Credential	Purpose	Storage
User TLS client private key	The RSA/Elliptic Curve key pairs and the X509 certificate are used for TLS mutual authentication (client side of TLS exchange) that is implemented by the platform.	iOS Keychain

6.2 User data protection

The Hypori Halo 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 during installation. A user initiates network connections to the Hypori Virtual Device on 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 Halo 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 Halo Client sensitive data consists of the user TLS client private key credential. FCS_STO_EXT.1 Storage of Secrets specifies the platform's iOS keychain for protecting the credential. In accordance with FCS_STO_EXT.1, the Hypori Halo Client stores this credential in the platform's iOS keychain as described in section 6.1.3.

6.2.2 FDP_DEC_EXT.1

At first launch, the Hypori Halo 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 Halo Client. Those marked with an '*' are prompted for when the Hypori Halo Client is started for the first time.

Table 9 Hypori Halo Client 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. *
FaceID/TouchID	Enables the use of biometric identification
Cellular Data	Enables the Hypori App to use Cellular Data when Wifi is not available.

Updates to the Hypori Halo 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 Virtual Device on the Hypori server by starting the Hypori Halo Client and entering account information. After the Hypori Halo Client connects to the Hypori Virtual Device on the Hypori server, the applications the user accesses run on the Hypori Device in the Hypori server, not on the mobile device. The Hypori Halo Client does not listen on any ports for inbound connection requests. The Hypori Halo 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 Halo Client – communicates with the server (which may be an internal, enterprise server).

The TOE does not access any sensitive information repositories as defined by the [PP_APP_v1.4].

The Hypori Client does not maintain PII. Hence, it does not transmit PII over any network.[8]

6.2.3 FDP_NET_EXT.1

The Hypori Client relies on user-initiated network communication to connect to the Hypori Virtual Device. The Hypori Halo 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. The Hypori Halo Client relies on user-initiated requests at https://hypori-client-help.s3.amazonaws.com/ios/master/index.html to access the Help webpage and https://hypori.com/support/ to request product help and support.

6.3 Identification and authentication

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

[8] The Hypori Halo Client does maintain user credentials. In particular, the Hypori Halo Client transmits a user's account name and TLS client certificate when connecting to the Hypori Server. However, **Error! Reference source not found.** distinguishes credentials from PII.

6.3.1 FIA_X509_EXT.1

The iOS platform performs certificate path validation in accordance with RFC 5280 as part of the TLS service. It recursively builds certificate chains until a valid chain is found or all possible paths are exhausted. The chain begins at the leaf certificate and ends in the final trusted root certificate[9] .

The iOS platform performs certificate path validation as part of the TLS service. The Hypori Halo 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 Halo Client relies on the platform to validate the revocation status of certificates using OSCP capabilities provided by the platform. The Hypori Halo Client does not perform email encryption, email signature verification, or Enrollment over Secure Transport. Consequently, no check is made for extended key usage Email Protection or CMC Registration Authority purposes.

The application uses the platform to validate the revocation status of the certificate using OCSP as specified in RFC 6960.

6.3.2 FIA_X509_EXT.2

The Hypori Halo Client can be used to contact the Hypori provisioning portal and download the user's credentials and store them into the iOS KeyChain using the following methods:

- Configure the Hypori Halo Client Account using a QR Code
    - The user will receive an enrollment email from the Hypori Halo administrator titled "Your Hypori account is ready". This email contains the QR code required to add the account as well as the account information. After the Hypori Halo Client is installed and launched, the application will ask for permission to allow the Hypori Halo Client access to your physical device's camera. The camera will automatically scan the QR code, proceed with the installation process, and save the credential information in the key store and connect the user to virtual workspace.
- Configure the Hypori Halo Client using the One-Time Password (OTP) Method
    - The OTP method of account provisioning is mostly used by users who may have the camera disabled on their physical device. The user will receive an enrollment email from the Hypori Halo administrator titled "Your Hypori account is ready". After the Hypori Halo Client is installed and launched, the application will ask for the user to populate fields using the information provided in the "Your Hypori account is ready" email: Email Address or Login Name, Server URL/Port Number, One-Time Password (OTP). The user can optionally change the Account Name. The installation process, will save the credential information in the key store and connect the user to virtual workspace.

The Hypori Client presents the TLS client certificate and public key to the Hypori server to authenticate a TLS connection. The TLS client certificate is an X.509 certificate.

The user stores a CA certificate for the server 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). On iOS devices, the Hypori Client uses iOS platform certificate path validation services with the CA certificate to validate the certificate presented by the Hypori server. The Hypori Client extracts the OCSP information from the certificate and performs the revocation checking to ensure that the certificate has not been revoked.

If the OCSP server fails to respond or there is an error, the Hypori Client will not accept the certificate (invalid) and not establish the connection.

[9] The iOS platform performs certificate 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.

6.4 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 Halo Client credentials consist of the user TLS client private key. The Hypori Halo Client installer does not include a default client key. The TOE obtains and stores the certificate and private key from the server during initial configuration. The user is not able to access any TOE functionality prior to the installation of the TLS client certificate and private key.

The application is configured by default with file permissions which protect the application's binaries and data files from modification by normal unprivileged users. The iOS protection classes use the NSFileProtectionComplete and NSFileProtectionCompleteUntilFirstUserAuthentication to protect the application's binaries and data files. See the following link for additional details.

https://developer.apple.com/documentation/foundation/nsfileprotectioncomplete?changes=_5&language=objc

6.4.2 FMT_MEC_EXT.1

The Hypori Halo Client invokes the recommended iOS mechanisms for storing account settings files. NSUserDefaults is the platformed provided mechanism for saving configuration data for the application.

The account options stored by NSUserDefaults consists of the Hypori Server hostname (IP address/URL), port number of the Hypori Server, Account Name, and the email address. The NSUserDefaults is invoked without any user intervention.

The Hypori Halo Client policies downloaded from the Hypori server are also stored using NSUserDefaults.

6.4.3 FMT_SMF.1

For each account, the Hypori Halo Client provides the capability to initially set the Hypori server URL, Hypori server port, account name, email address, and TLS client certificate (private key). Except for the private key, these values are provided to the user and either manually entered during initial configuration or obtained by the TOE when the user scans the QR Code. The TOE acquires the TLS client certificate (private key) from the Hypori backend server during the account setup process. After the account has been set-up, the user only has the capability to change the account name.

Client policies are automatically downloaded from the Hypori Server and are applied during initial configuration. After initial configuration, the Hypori Halo Client policies are downloaded and refreshed every time the Hypori Halo Client authenticates to the Hypori Server, whether or not any changes have been made on the Hypori Server. The Hypori Halo Client does not listen on any ports for inbound connection requests. The Hypori Halo Client interacts only with the Hypori server. When a virtual Hypori Device application needs information from a server (such as a map server), the virtual Hypori Device – not the Hypori Halo Client – communicates with the server (which may be an internal, enterprise server).

6.5 Privacy

6.5.1 FPR_ANO_EXT.1

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

6.6 Protection of the TSF

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

6.6.3 FPT_IDV_EXT.1

The TOE is the Hypori Halo Client (iOS) v 4.3. The TOE is identified and versioned by the Apple App Store version identifiers as well as the internal Hypori build information.

As an example, the Apple App Store will show two variations of the Hypori App version:

- 4.3.0 – this form is just a version/release number
- 4.3.0 (403000001) – this form is what Apple calls a build number. These values can be seen in the Apple App Store.

The Hypori Halo Client application also provides version information in the application itself (e.g. in the settings UI and the account list UI). An example of the format inside the app is:

- 4.3.0 (403000001-fecd124e) - The Hypori Halo Client 4.3.0 version is interpreted as a major.minor.maintenance-release format. The additional information in this format string is Hypori internals specific and relates to a particular software tag used for tracking released builds in Hypori's SCM repository.

6.6.4 FPT_LIB_EXT.1

The Hypori Halo Client package includes only the third-party libraries listed below:

- PLCrashReporter-1.2/CrashReporter.framework v1.2
- ios-openssl v1.1e
- OCMock-3.4
- Opus Audio Codec v1.1
- protobuf v2.5
- spice-qtk v2.2
- grpc-swift v1.8.1
- swift-atomics v1.0.2
- swift-collections v1.0.3
- swift-log v1.4.4
- swift-nio v2.4.3.1
- swift-nio-extras v1.14.0
- swift-nio-http2 v1.23.0
- swift-nio-ssl v2.23.0
- swift-nio-transport-services v1.15.0
- SwiftProtobuf v1.20.2

6.6.5 FPT_TUD_EXT.1, FPT_TUD_EXT.2

Hypori distributes the Hypori Halo 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 Halo Client updates using the platform's update mechanism or from the user's IT group. Hypori 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.

To verify the version of the Hypori Halo Client, open the Hypori Halo Client, but do not connect to the Virtual Device. On the Hypori Halo Client Accounts screen, select the ellipses menu and click on 'About'. The About screen will display the version number, build information and copyright.

6.7 Trusted path/channels

The Hypori Halo 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 Halo Client communicates with. For all communication with the Hypori server, the Hypori Halo Client connects to the server using TLS 1.2 provided by the platform.

The TOE calls to the platform to leverage this functionality in three places:

- PLIST/"No arbitrary loads" is a mechanism to tell the iOS platform that the Hypori Halo Client will not make any HTTP or non-TLS protected communications. It is set in a PLIST file that is bundled inside the Apple IPA file. The PLIST is not a classic API call, but a mechanism for the Hypori Halo Client to constrain how it can invoke remote services.
- The calls outgoing on NSURLSession will invoke App Transport Security (ATS).
- The calls outgoing on SWIFTNIO use Network.Framework and DO NOT invoke ATS.

Apple's documentation of App Transport Security (ATS) can be found here:

- https://developer.apple.com/documentation/bundleresources/information_property_list/nsapptransportsecurity?language=objc.

Apple's documentation of SWIFTNIO Network.Framework can be found here:

- https://opensource.apple.com/projects/swiftnio/.
- https://nitinagam.medium.com/network-framework-in-ios-6eebacb99894.

6.8 Timely Security Updates

6.8.1 ALC_TSU_EXT.1

Hypori provides customers with timely updates. A customer chooses their preferred communication. The Hypori 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.

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

7. Protection Profile Claims

This ST conforms to the *Protection Profile for Application Software,* Version 1.4, 2021-10-07 [PP_APP_v1.4].

As explained in Section 3, Security Problem Definition, the Security Problem Definition of the [PP_APP_v1.4] has been included by reference into this ST.

As explained in Section 4, Security Objectives, the Security Objectives of the [PP_APP_v1.4] 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_v1.4] and operations completed as appropriate.

Table 10 SFR Protection Profile Sources

Requirement Class	Requirement Component	Source
FCS: Cryptographic support	FCS_CKM_EXT.1 Cryptographic Key Generation Services	[PP_APP_v1.4]
	FCS_CKM.1/AK Cryptographic Asymmetric Key Generation	[PP_APP_v1.4]
	FCS_CKM.2 Cryptographic Key Establishment	[PP_APP_v1.4]
	FCS_RBG_EXT.1 Random Bit Generation Services	[PP_APP_v1.4]
	FCS_STO_EXT.1 Storage of Credentials	[PP_APP_v1.4]
FDP: User data protection	FDP_DAR_EXT.1 Encryption of Sensitive Application Data	[PP_APP_v1.4]
	FDP_DEC_EXT.1 Access to Platform Resources	[PP_APP_v1.4]
	FDP_NET_EXT.1 Network Communications	[PP_APP_v1.4]
FIA: Identification and authentication	FIA_X509_EXT.1 X.509 Certificate Validation	[PP_APP_v1.4]
	FIA_X509_EXT.2 X.509 Certificate Authentication	[PP_APP_v1.4]
FMT: Security management	FMT_CFG_EXT.1 Secure by Default Configuration	[PP_APP_v1.4]
	FMT_MEC_EXT.1 Supported Configuration Mechanism	[PP_APP_v1.4]
	FMT_SMF.1 Specification of Management Functions	[PP_APP_v1.4]
FPR: Privacy	FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information	[PP_APP_v1.4]
FPT: Protection of the TSF	FPT_AEX_EXT.1 AntiExploitation Capabilities	[PP_APP_v1.4]
	FPT_API_EXT.1.1 Use of Supported Services and APIs	[PP_APP_v1.4]
	FPT_IDV_EXT.1 Software Identification and Versions	[PP_APP_v1.4]
	FPT_LIB_EXT.1 Use of Third Party Libraries	[PP_APP_v1.4]
	FPT_TUD_EXT.1 Integrity for Installation and Update	[PP_APP_v1.4]
	FPT_TUD_EXT.2 Integrity for Installation and Update	[PP_APP_v1.4]
FTP: Trusted path/channels	FTP_DIT_EXT.1 Protection of Data in Transit	[PP_APP_v1.4]

8. Rationale

This security target includes by reference the [PP_APP_v1.4] Security Problem Definition, Security Objectives, and Security Assurance Requirements. The security target makes no additions to the [PP_APP_v1.4] assumptions. [PP_APP_v1.4] security functional requirements have been reproduced with the [PP_APP_v1.4] operations completed. Operations on the security requirements follow [PP_APP_v1.4] application notes and assurance activities. Consequently, [PP_APP_v1.4] 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

The Protection Profile for Application Software [PP_APP_v1.4] contains all the requirements claimed in this Security Target. As such, the dependencies are not applicable since the PP has been approved.

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 11 demonstrates the relationship between security requirements and security functions.

Table 11 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_CKM.1/AK	X						
FCS_CKM.2	X						
FCS_RBG_EXT.1	X						
FCS_STO_EXT.1	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_IDV_EXT.1						X	

	Cryptographic support	User data protection	Identification and authentication	Security management	Privacy	Protection of the TSF	Trusted path/channels
FPT_LIB_EXT.1						X	
FPT_TUD_EXT.1						X	
FPT_TUD_EXT.2						X	
FTP_DIT_EXT.1							X

9. Appendix: iOS APIs

The Hypori Halo 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_SHA256
22. CFArrayGetCount
23. CFArrayGetValueAtIndex
24. CFAutorelease
25. CFCopyDescription
26. CFDataGetTypeID
27. CFDictionaryGetValue
28. CFGetTypeID
29. CFPropertyListCreateDeepCopy
30. CFRelease
31. CFRetain
32. CFRunLoopGetCurrent
33. CFUUIDCreate
34. CFUUIDCreateFromUUIDBytes
35. CFUUIDCreateString
36. CFUUIDGetUUIDBytes
37. CGAffineTransformMakeRotation
38. CGAffineTransformMakeTranslation
39. CGAffineTransformRotate
40. CGAffineTransformScale
41. CGAffineTransformTranslate
42. CGBitmapContextCreate

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

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

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

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

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

263. UNNotificationServiceExtension
264. UNNotificationSound
265. UNUserNotificationCenter
266. VTCompressionSessionCreate
267. VTCompressionSessionEncodeFrame
268. VTCompressionSessionInvalidate
269. VTDecompressionSessionCanAcceptFormatDescription
270. VTDecompressionSessionCreate
271. VTDecompressionSessionDecodeFrame
272. VTDecompressionSessionInvalidate
273. VTSessionSetProperty
274. glActiveTexture
275. glAttachShader
276. glBindTexture
277. glBlendFunc
278. glClear
279. glClearColor
280. glCompileShader
281. glCreateProgram
282. glCreateShader
283. glDeleteProgram
284. glDeleteShader
285. glDeleteTextures
286. glDisable
287. glDisableVertexAttribArray
288. glDrawElements
289. glEnable
290. glEnableVertexAttribArray
291. glGenTextures
292. glGetAttribLocation
293. glGetProgramiv
294. glGetUniformLocation
295. glLinkProgram
296. glShaderSource
297. glTexImage2D
298. glTexParameterf
299. glTexParameteri
300. glTexSubImage2D
301. glUniform1f
302. glUniform1i
303. glUniformMatrix4fv
304. glUseProgram
305. glVertexAttribPointer
306. glViewport

307.WKWebView