Hypori Virtual Mobile Infrastructure Platform 3.1.0

Android Cloud Environment Client Security Target

Version 1.0

February 17, 2016

Prepared for:

Hypori, Inc.

9211 Waterford Centre Blvd, Suite 100

Austin, TX 78758

Prepared by:

Leidos Inc. (formerly Science Applications International Corporation)

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 Android Cloud Environment Client component of the Virtual Mobile Infrastructure Platform version 3.1.0 provided by Hypori, Inc..

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).

1.1 Security Target, TOE and CC Identification

ST Title – Hypori Virtual Mobile Infrastructure Platform 3.1.0 Android Cloud Environment Client Security Target

ST Version – Version 1.0

ST Date – February 17, 2016

TOE Identification – Android Cloud Environment Client 3.1.0

TOE Developer – Hypori, Inc.

Evaluation Sponsor – Hypori, Inc.

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 1.1, 5 November 2014 (PP APP SW) including *DoD Annex for Protection Profile for Application Software v1.0,* Version 1, Release 1 (DoD Annex), 22 October 2014. NIAP Technical Decision TD0051 Android Implementation of TLS in App PP v1.1 applies and is addressed in this ST. The following NIAP Technical Decisions apply to evaluation assurance activities.
    - TD0054: Clarification of FPT_API_EXT.1.1 Requirement in APP PP v1.1
    - TD0050: FMT_CFG_EXT.1.2 Change in APP SW PPv1.1
- 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.

ACE	Android Cloud Environment
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 Android Cloud Environment component, which is the Target of Evaluation (TOE). The description covers TOE architecture, logical boundaries, and physical boundaries.

2.1 Product Overview

Hypori Android Cloud Environment (ACE) is a Virtual Mobile Infrastructure (VMI) platform. End users running a Hypori ACE 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 ACE Client.

The Hypori VMI platform includes the following components:

- **ACE Client:** This is a thin client that installs on the end user's mobile device and communicates with the ACE Device on the server through secure encrypted protocols.
- **ACE Device:** This is an Android-based virtualized version of the end user's mobile device.
- **ACE Servers:** This is the cloud server cluster that hosts the ACE Devices.
- **ACE Administrator Portal:** This is a browser-based administration user interface that is used to manage the ACE system.

2.2 TOE Overview

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

Figure 1 ACE Client as Part of VMI Platform

2.3 TOE Architecture

The section describes the TOE architecture including physical and logical boundaries. Figure 2 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.

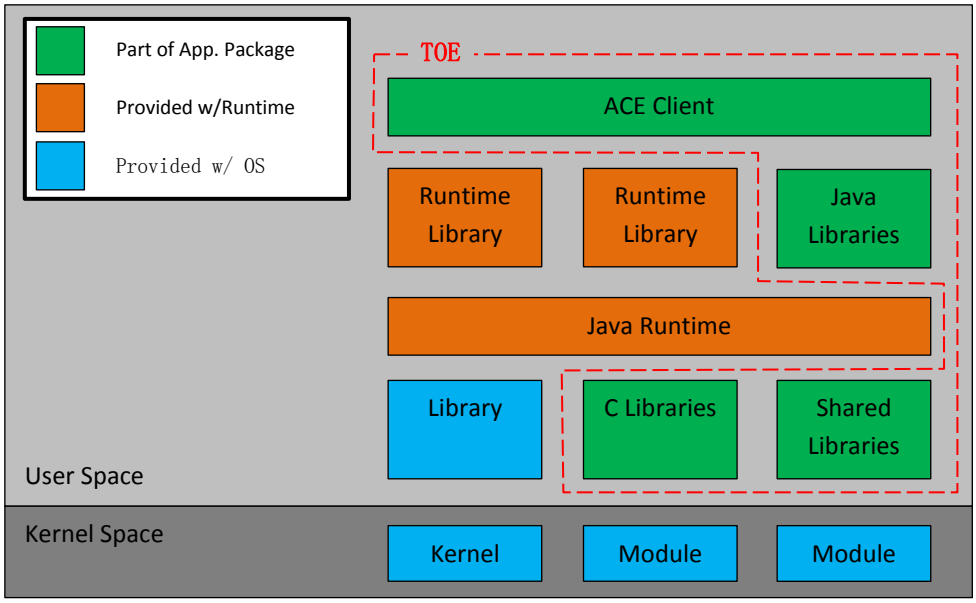

Figure 2 TOE Boundary

2.3.1 Physical Boundaries

The TOE consists of ACE Client application and configuration settings as defined in the ACE Client installation package for Android. ACE Client is a thin client that only communicates with the ACE server. The ACE Server, applications running on the ACE 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 Android versions 4.3, 4.4, 5.0 and 5.1.

2.3.1.2 Hardware Requirements

The TOE imposes no hardware requirements beyond Android 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
- Protection of the TSF
- Trusted path/channels

2.3.2.1 Cryptographic support

The TOE establishes secure communication with the ACE Server using TLS. The client uses cryptographic services provided by the Android platform. TOE stores credentials and certificates for mutual authentication in the Android key store.

2.3.2.2 User data protection

The TOE informs a user of hardware and software resources the TOE accesses. It uses the Android 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 ACE Server using the TOE. In general, sensitive data resides on the ACE Device and not the ACE 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 Android certification validation services to authenticate the X.509 certificate the ACE Server presents as part of the establishing a TLS connection.

2.3.2.4 Security management

Security management consists of setting ACE Client configuration options. The TOE uses Android mechanisms for storing the configuration settings.

2.3.2.5 Protection of the TSF

The TOE uses security features and APIs that the Android platform provides. The TOE leverages Android package management for secure installation and updates. The TOE package includes only those third-party libraries necessary for its intended operation.

2.3.2.6 Trusted path/channels

TOE uses TLS 1.2 for all communication with ACE Server.

2.4 TOE Documentation

The TOE includes the following ACE Client documentation.

- *Hypori ACE User Guide Common Criteria Configuration and Operation,* Version 3.1.0
- *Hypori ACE Client Install Guide for Android Devices,* Version 3.1 - Enterprise Distribution
- *Hypori ACE User Guide for Android Devices,* Version 3.1 - ACE Device

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 which runs on mobile devices, as well as on desktop and server platforms. The ACE Client is an Android 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 appropriate for application software which runs on mobile devices, as well as on desktop and server platforms. Consequently, the PP APP SW security objectives are suitable for the ACE 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 1.1, 5 November 2014 (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 Secrets
- FCS_TLSC_EXT.1 TLS Client Protocol (including optional element 1.4)
- FDP_DAR_EXT.1 Encryption Of Sensitive Application Data
- 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
- 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 ACE 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 Secrets
	FCS_TLSC_EXT.1 TLS Client Protocol (including element 1.4)

Requirement Class	Requirement Component
FDP: User data protection	FDP_DAR_EXT.1 Encryption of Sensitive Application Data
	FDP_DEC_EXT.1 Access to Platform Resources
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
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 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 Secrets (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 key and server account password]*]** to nonvolatile memory.

5.2.1.4 TLS Client Protocol on Android 4.3 (FCS_TLSC_EXT.1(1))

FCS_TLSC_EXT.1.1(1) The application shall [***invoke platform-provided TLS 1.2***] supporting the following ciphersuites:

Mandatory Ciphersuites:

TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Ciphersuites: [:

TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,

TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246

] .

FCS_TLSC_EXT.1.2(1) The application shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3(1) The application shall only establish a trusted channel if the peer certificate is valid.

FCS_TLSC_EXT.1.4(1) The application shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.1.5(1) The application shall present the **S**upported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1,***] and no other curves.

5.2.1.5 TLS Client Protocol on Android 4.4 (FCS_TLSC_EXT.1(2))

FCS_TLSC_EXT.1.1(2) The application shall [***invoke platform-provided TLS 1.2***] supporting the following ciphersuites:

Mandatory Ciphersuites:

TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Ciphersuites: [:

TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,

TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246

] .

FCS_TLSC_EXT.1.2(2) The application shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3(2) The application shall only establish a trusted channel if the peer certificate is valid.

FCS_TLSC_EXT.1.4(2) The application shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.1.5(2) The application shall present the **S**upported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1,***] and no other curves.

5.2.1.6 TLS Client Protocol on Android 5.0 and 5.1 (FCS_TLSC_EXT.1(3))

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

Mandatory Ciphersuites:

TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Ciphersuites: [:

TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,

TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,

TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,

TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,

TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246

] .

FCS_TLSC_EXT.1.2(3) The application shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3(3) The application shall only establish a trusted channel if the peer certificate is valid.

FCS_TLSC_EXT.1.4(3) The application shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.1.5(3) The application shall present the **S**upported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1,***] and no other curves.

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 [***leverage platform-provided functionality to encrypt sensitive data,***] in nonvolatile memory.

5.2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)

FDP_DEC_EXT.1.1 The application shall provide user awareness of its intent to access [

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

] .

FDP_DEC_EXT.1.2 The application shall provide user awareness of its intent to access [

- ***[accounts on device]***

] .

FDP_DEC_EXT.1.3 The application shall only seek access to those resources for which it has provided a justification to access.

FDP_DEC_EXT.1.4 The application shall restrict network communication to **[**

- ***user-initiated communication for [connecting to ACE Server and polling ACE Server for notification],***

] .

FDP_DEC_EXT.1.5 The application shall [***not transmit PII over a network***].

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 [***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 [***none***][1].
- 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.[2]
  - 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. [4]

---

[1] See TD0051

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

[3] ACE Client does not check extended key usage for Email Protection, ACE Client since does not perform email encryption or email signature verification.

[4] ACE Client does not check extended key usage for OCSP Signing, since ACE Client does not use OCSP to check certificate revocation status.

- 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.[5]

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 [***accept the certificate***].[6]

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 only provide 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 it and its data from unauthorized access.

5.2.4.2 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.

5.2.4.3 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 ACE Server]

] .

5.2.5 Protection of the TSF (FPT)

5.2.5.1 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.

---

[5] ACE Client does not check extended key usage for CMC Registration Authority, since ACE Client does not perform Enrollment over Secure Transport.

[6] The ACE Client relies on the Android platform for certificate validation services. Android does not provide revocation checking as part of certificate validation. Hence, there is no communication with a revocation service and no certificate is rejected for lack of revocation check.

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.

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

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

5.2.5.3 Use of Third Party Libraries (FPT_LIB_EXT.1)

FPT_LIB_EXT.1.1 The application shall be packaged with only [

3rd party (non-Android framework) libraries:

java libraries:

1. **protocol buffers (from Google)**
2. **json (via source code, from json.org)**
3. **BadgeView (ui class from readystate software)**

C libraries:

4. **celt (audio compression)**
5. **opus (audio compression)**
6. **gnu libraries (gmodule, gobject, gthread, glib, gio)**
7. **libjpeg-turbo (accelerated image compression)**
8. **libffi (foreign function library required by gobject)**
9. **spice (for legacy spice drawing support)**
10. **pixman (pixel manipulation for legacy spice drawing support)**
11. **glue (2d graphics used by spice)**

Shared libraries included with the TOE:

12. **libdc_10_spicegtk-common-0.11.so**
13. **libdc_1_libglib-2.32.1.so**
14. **libdc_2_libffi-3.0.11.so**
15. **libdc_3_libgobject-2.32.1.so**
16. **libdc_4_libthread-2.32.1.so**
17. **libdc_5_libgmodule-2.32.1.so**
18. **libdc_6_libgio-2.32.1.so**
19. **libdc_7_libpixman-0.24.4.so**
20. **libdc_8_libjpeg-turbo.so**
21. **libdc_9_libglues-v1.4.so**
22. **libdc_libcelt-0.5.1.3.so**
23. **libdc_libopus_1.1-131-ga88d836.so**

].

5.2.5.4 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.

5.2.6 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.

5.3 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

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
- Identification and authentication
- Security management
- Protection of the TSF
- Trusted path/channels

6.1 Cryptographic support

ACE Client makes use of the Android platform for cryptographic services. ACE Client uses Android platform TLS services for secure communication with the ACE 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 Android 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

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

6.1.2 FCS_RBG_EXT.1

The ACE Client relies on the Android platform for cryptographic services[7]. Consequently, the ACE Client itself uses no DRBG functions.

6.1.3 FCS_STO_EXT.1

Table 3 lists each ACE 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 ACE client when establishing TLS connection to ACE Server	Android KeyStore
Server account password	Authenticates user to ACE Server	Android KeyStore

6.1.4 FCS_TLSC_EXT.1(1), (2), and (3)

The ACE Client relies on the Android platform for TLS protection of communication with the ACE Server. This includes providing the TLS client certificate to authenticate the client to the server. ACE Server authenticates an

[7] To address a defect in Android 4.3, ACE Client explicitly seeds the OpenSSL PRNG from `/dev/urandom` to address a Java Cryptography Architecture defect as described on http://android-developers.blogspot.com.es/2013/08/some-securerandom-thoughts.html. ACE Client applies the fix to Android 4.3 SecureRandom exactly as posted. The fix does not affect Android 4.4, 5.0, and 5.1, since the defect is not present in those releases.

ACE Client. Different versions of Android support distinct sets of ciphersuites as shown in Table 4. The column labeled Server shows the default ACE Server configuration, which may be changed to accommodate the other identified ciphersuites.

Table 4 Ciphersuite Support by Android Version

	Ciphersuite	**4.3**	**4.4**	**5.0**	**5.1**	**Server**
1.	TLS_RSA_WITH_AES_128_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
2.	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
3.	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256			Yes	Yes	Default
4.	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
5.	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256			Yes	Yes	Default
6.	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
7.	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256			Yes	Yes	Default
8.	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256			Yes	Yes	Default
9.	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
10.	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384			Yes	Yes	Default
11.	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384			Yes	Yes	Default
12.	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
13.	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			Yes	Yes	Configurable
14.	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
15.	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384			Yes	Yes	Default
16.	TLS_RSA_WITH_AES_128_CBC_SHA256			Yes	Yes	Default
17.	TLS_RSA_WITH_AES_256_CBC_SHA	Yes	Yes	Yes	Yes	Configurable
18.	TLS_RSA_WITH_AES_256_CBC_SHA256			Yes	Yes	Default

For elliptic curve ciphersuites, the ACE Client relies on the Android platform for elliptic curves. The Android platform supports NIST curves secp256r1 and secp384r1 and Supported Elliptic Curves Extension for TLS. No configuration is required by an ACE Client user.

ACE Client establishes the reference identifier using the configured server host name. ACE 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.

6.2 User data protection

ACE Client uses Android 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 ACE Server. In general, sensitive data resides on the ACE Device and is not stored on the ACE Client. Sensitive data on the ACE 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

ACE Client sensitive data consist of user TLS client key and server account password credentials. FCS_STO_EXT.1 Storage of Secrets specifies the Android KeyStore for protecting keys and credentials. In accordance with FCS_STO_EXT.1, ACE Client stores these credentials in the Android KeyStore as described in section 6.1.3.

ACE Client stores application account options using Android's SharedPreferences. The SharedPreferences files are accessed using the MODE_PRIVATE flag, even though the application account options do not contain sensitive data.

6.2.2 FDP_DEC_EXT.1

The installer presents to the user the permissions required by the ACE Client. A user must accept the permissions to complete installation. The ACE Client requires the following permissions:

1) Read phone status and identity
2) Take pictures and videos (Camera)
3) Record audio (Microphone)
4) GPS and network-based location
5) Find, use, add, and remove accounts and set passwords
6) Access and change network state
7) Access Wi-Fi connection state information
8) Retrieve running apps
9) Change audio settings
10) Read and enable/disable sync settings
11) Install/uninstall shortcuts
12) Prevent phone from sleeping
13) Display a system alert
14) Receive boot completed
15) Full network access
16) View network connections
17) Read external storage

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

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

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

6.3 Identification and authentication

The Android platform follows RFC 5280 *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* for certification path validation. ACE Client uses the Android certification validation services to authenticate the X.509 certificate the ACE Server presents as part of the establishing a TLS connection.

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

6.3.1 FIA_X509_EXT.1[9]

The Android platform performs certification path validation as part of the TLS service. The platform certificate path algorithm is described by its Android platform source code:

> https://android.googlesource.com/platform/external/conscrypt/+/android-5.0.0_r7/src/platform/java/org/conscrypt/TrustManagerImpl.java

See the checkTrusted() method at line 249 for the algorithm. ACE Client relies on Android for TLS services and package updates. Hence, Android checks extended key usage for Server Authentication, Client Authentication, and Code Signing purposes. ACE Client does not perform email encryption, email signature verification, OCSP checking, and Enrollment over Secure Transport. Consequently, no check is made for extended key usage Email Protection, OCSP Signing, and CMC Registration Authority purposed.

6.3.2 FIA_X509_EXT.2

ACE Client presents TLS client certificate and key to the ACE 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 Android certificate store. The user can change the selection from Client Certificate under Connection on the Settings page. The TLS client certificate is an X509 certificate.

The user stores a CA certificate for the server certificates in the Android key store during installation. (The user need not install a CA certificate when the CA is a platform trusted CA.) ACE Client uses Android platform certificate path validation services with the CA certificate to validate the certificate presented by the ACE Server. As noted above, Android does not provide revocation checking as part of certificate validation. Hence, there is no communication with a revocation service and no certificate is rejected for lack of revocation check.

6.4 Security management

Security management consists of setting ACE Client configuration options. The client uses Android mechanisms for storing the configuration settings.

6.4.1 FMT_CFG_EXT.1

ACE Client credentials consist of user TLS client key and server account password. ACE 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 Android's certificate services. A user's IT group provides the user with server account password.

6.4.2 FMT_MEC_EXT.1

ACE Client invokes the recommended Android mechanisms for storing configuration data. The client uses SharedPreferences and extends PreferenceActivity.

6.4.3 FMT_SMF.1

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

The ACE 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 ACE Client does not listen on any ports for inbound connection requests. The ACE Client interacts only with the ACE Server. When an ACE Device application needs information from a server (such as a map server), the ACE Device – not the ACE Client – communicates with the server (which may be an internal, enterprise server).

[9] TD0051 Android Implementation of TLS in App PP v1.1 applies.

6.5 Protection of the TSF

ACE Client uses security features and APIs that the Android platform provides. This includes address space layout randomization, data execution protection, Security Enhancements for Android, and stack-based buffer overflow protection. The client leverages Android package management for secure installation and updates. The ACE Client package does not includes only those third-party libraries necessary for its intended operation.

6.5.1 FPT_AEX_EXT.1

Hypori enables address space layout randomization (ASLR) in ACE Client using `-fpic` when building the application with Android Native Development Kit (NDK r9d) using gcc. ACE Client is a Java application that includes Java Native Interface (JNI) libraries. Hypori enables stack-based buffer overflow protection using `-fstack-protector`. ACE Client does not invoke `mmap` or `mprotect` from the Android NDK.

6.5.2 FPT_API_EXT.1

ACE Client uses the Android APIs listed in section 9 Appendix: Android APIs

6.5.3 FPT_LIB_EXT.1

ACE Client package includes only the third-party libraries listed in the security functional requirements.

6.5.4 FPT_TUD_EXT.1

Hypori distributes ACE Client as an .APK file. A user may obtain the installation package through Google Play or the enterprise IT group of the user. A user obtains ACE Client updates using the Android update mechanism or from the user's IT group. Hypori digitally signs the installation package as well as updates and includes the corresponding public key certificate in the package. Android will install an update only when the certificate in the update matches the certificate in the installed client. The client is signed with a unique certificate. It can be delivered via the Google Play store, MDM, or other enterprise app stores. The certificate information:

```
X.509, CN=Brian Vetter, OU=DroidCloud, O=DroidCloud, L=Austin, ST=Tx, C=US
      [certificate is valid from 3/20/13 1:03 PM to 8/5/401:03 PM]
      [CertPath not validated: Path does not chain with any of the trust
anchors]
```

A user can query the current version of the by checking the ACE Client Version setting under Debug Settings from the Settings screen.

Hypori provides customers with timely updates. A customer chooses their preferred communication. 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 Google Play store like any other Android 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 customers preferred communication mechanism. Hypori 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.

6.6 Trusted path/channels

ACE Client uses TLS 1.2 for all communication with ACE Server.

6.6.1 FTP_DIT_EXT.1

The ACE Server is the only trusted IT product the ACE Client communicates with. For all communication with the ACE Server, the ACE Client connects to the server using TLS 1.2 provided by the Android platform.

7. Protection Profile Claims

This ST conforms to the *Protection Profile for Application Software,* Version 1.1, 5 November 2014 (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 5 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 Secrets	PP APP SW
	FCS_TLSC_EXT.1 TLS Client Protocol (including element 1.4)	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
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
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 ath/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 element (FCS_TLSC_EXT.1.4) and no objective requirements.

Table 6 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 satisfied or not applicable along with brief rationale

Table 6 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.1	SB	FCS_TLSC_EXT.1.5	Select elliptic curve ciphersuites	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 TLS	Not applicable	Not selected

Requirement	Type	Dependency	Condition	Satisfied	Rationale
FCS_TLSC_EXT.1	SB	FIA_X509_EXT.1	Unconditional	Satisfied	Included in ST
FCS_TLSC_EXT.1.4	Op	FIA_X509_EXT.2	Unconditional	Satisfied	Included in ST
FCS_TLSC_EXT.1.6	*Ob*	*None*	*None*	*Not applicable*	
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
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	
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 Rationale for Operations

This section provides justification for iteration and refinement operations where exact compliance to PP APP SW might be an issue.

8.2.1 FCS_TLSC_EXT.1(1), (2), and (3) Rationale

ACE Client satisfies FCS_TLS_EXT.1 on all Android platforms claims. However, different versions of Android support distinct sets of ciphersuites. This security target iterates FCS_TLSC_EXT.1 to specify the distinct sets of ciphersuites.

8.2.2 Rationale for TD0051

TD0051 Android Implementation of TLS in App PP v1.1 applies to ACE Client. Android does not provide revocation checking as part of certification path validation. Hypori provides countermeasures to the threat of a compromised server certificate that are at least as effective as certificate revocation checking. In the evaluated configuration, it is best to use an enterprise CA signing certificate for ACE Server certificates and use a service such as GlobalSign's trusted root certificate service to anchor its trust chain.

Using the Hypori server configuration tools, an administrator schedules the provisioning server to generate server certificates on a prescribed schedule. This schedule should correspond to a typical CRL update schedule. For example, new certificates would be generated daily.

These server certificates would be set to expire after a fixed number of hours based upon the same schedule along with some margin for delivering the updates. For the example of daily update schedule, certificates would be configured to expire every 25 hours.

Using this practice, a server's certificate will only be good for a short period of time - typically the time it takes to roll out a CRL update throughout the enterprise. The server certificates are updated regularly. If a server certificate is compromised, it will only be usable for a very short time if at all. That time period is likely less than it takes for an enterprise to detect that a server's key/certificate is compromised and to revoke the certificate and update the CRLs and OCSP responders.

8.3 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 work 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 7 demonstrates the relationship between security requirements and security functions.

Table 7 Security Functions vs. Requirements Mapping

	Cryptographic support	User data protection	Identification and authentication	Security management	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					
FDP_DAR_EXT.1		X				
FDP_DEC_EXT.1		X				

	Cryptographic support	User data protection	Identification and authentication	Security management	Protection of the TSF	Trusted path/channels
FIA_X509_EXT.1			X			
FIA_X509_EXT.2			X			
FMT_CFG_EXT.1				X		
FMT_MEC_EXT.1				X		
FMT_SMF.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: Android APIs

ACE Client uses the following Android APIs:

1. android.accounts.AbstractAccountAuthenticator
2. android.accounts.Account
3. android.accounts.AccountAuthenticatorActivity
4. android.accounts.AccountAuthenticatorResponse
5. android.accounts.AccountManager
6. android.accounts.AccountManagerCallback
7. android.accounts.AccountManagerFuture
8. android.accounts.AccountsException
9. android.accounts.NetworkErrorException
10. android.animation.Animator
11. android.animation.AnimatorListenerAdapter
12. android.animation.ValueAnimator
13. android.annotation.SuppressLint
14. android.annotation.TargetApi
15. android.app.Activity
16. android.app.ActivityManager
17. android.app.ActivityManager.RunningTaskInfo
18. android.app.AlertDialog
19. android.app.Application
20. android.app.Dialog
21. android.app.DialogFragment
22. android.app.KeyguardManager
23. android.app.ListActivity
24. android.app.Notification
25. android.app.NotificationManager
26. android.app.PendingIntent
27. android.app.ProgressDialog
28. android.app.SearchManager
29. android.app.SearchableInfo
30. android.app.Service
31. android.app.admin.DeviceAdminReceiver
32. android.app.admin.DevicePolicyManager
33. android.content.AbstractThreadedSyncAdapter
34. android.content.ActivityNotFoundException
35. android.content.BroadcastReceiver
36. android.content.ClipData
37. android.content.ComponentName
38. android.content.ContentProvider
39. android.content.ContentProviderClient
40. android.content.ContentResolver
41. android.content.ContentUris
42. android.content.ContentValues
43. android.content.Context
44. android.content.DialogInterface
45. android.content.DialogInterface.OnClickListener
46. android.content.Intent
47. android.content.Intent.ShortcutIconResource
48. android.content.IntentFilter
49. android.content.ServiceConnection
50. android.content.SharedPreferences
51. android.content.SharedPreferences.Editor

52. android.content.SharedPreferences.OnSharedPreferenceChangeListener
53. android.content.SyncResult
54. android.content.pm.ActivityInfo
55. android.content.pm.PackageInfo
56. android.content.pm.PackageManager
57. android.content.pm.PackageManager.NameNotFoundException
58. android.content.pm.ResolveInfo
59. android.content.res.AssetFileDescriptor
60. android.content.res.Configuration
61. android.content.res.Resources
62. android.content.res.TypedArray
63. android.database.ContentObserver
64. android.database.Cursor
65. android.database.MatrixCursor
66. android.graphics.Bitmap
67. android.graphics.Bitmap.CompressFormat
68. android.graphics.BitmapFactory
69. android.graphics.Canvas
70. android.graphics.Color
71. android.graphics.ImageFormat
72. android.graphics.Matrix
73. android.graphics.Paint
74. android.graphics.Path
75. android.graphics.PixelFormat
76. android.graphics.PorterDuff
77. android.graphics.Rect
78. android.graphics.RectF
79. android.graphics.SurfaceTexture
80. android.graphics.Typeface
81. android.graphics.YuvImage
82. android.graphics.drawable.BitmapDrawable
83. android.graphics.drawable.ColorDrawable
84. android.graphics.drawable.Drawable
85. android.graphics.drawable.ShapeDrawable
86. android.graphics.drawable.TransitionDrawable
87. android.graphics.drawable.shapes.RoundRectShape
88. android.hardware.Camera
89. android.hardware.Camera.Area
90. android.hardware.Camera.CameraInfo
91. android.hardware.Camera.Face
92. android.hardware.Camera.FaceDetectionListener
93. android.hardware.Camera.Parameters
94. android.hardware.Camera.PictureCallback
95. android.hardware.Camera.Size
96. android.hardware.Sensor
97. android.hardware.SensorEvent
98. android.hardware.SensorEventListener
99. android.hardware.SensorManager
100. android.location.Criteria
101. android.location.GpsSatellite
102. android.location.GpsStatus
103. android.location.GpsStatus.Listener
104. android.location.GpsStatus.NmeaListener
105. android.location.Location
106. android.location.LocationListener
107. android.location.LocationManager

108. android.location.LocationProvider
109. android.media.AudioFormat
110. android.media.AudioManager
111. android.media.AudioRecord
112. android.media.AudioTrack
113. android.media.CamcorderProfile
114. android.media.CameraProfile
115. android.media.MediaActionSound
116. android.media.MediaCodec
117. android.media.MediaCodec.BufferInfo
118. android.media.MediaCodecInfo
119. android.media.MediaCodecList
120. android.media.MediaFormat
121. android.media.MediaMetadataRetriever
122. android.media.MediaRecorder
123. android.media.ThumbnailUtils
124. android.media.audiofx.AcousticEchoCanceler
125. android.media.audiofx.AutomaticGainControl
126. android.media.audiofx.NoiseSuppressor
127. android.net.ConnectivityManager
128. android.net.LocalSocket
129. android.net.LocalSocketAddress
130. android.net.NetworkInfo
131. android.net.Uri
132. android.net.Uri.Builder
133. android.net.wifi.WifiInfo
134. android.net.wifi.WifiManager
135. android.net.wifi.WifiManager.WifiLock
136. android.os.AsyncTask
137. android.os.BatteryManager
138. android.os.Binder
139. android.os.Build
140. android.os.Bundle
141. android.os.Environment
142. android.os.Handler
143. android.os.HandlerThread
144. android.os.IBinder
145. android.os.Looper
146. android.os.Message
147. android.os.Parcel
148. android.os.ParcelFileDescriptor
149. android.os.ParcelFileDescriptor.AutoCloseOutputStream
150. android.os.Parcelable
151. android.os.PowerManager
152. android.os.Process
153. android.os.StatFs
154. android.os.SystemClock
155. android.preference.CheckBoxPreference
156. android.preference.EditTextPreference
157. android.preference.Preference
158. android.preference.Preference.OnPreferenceChangeListener
159. android.preference.Preference.OnPreferenceClickListener
160. android.preference.PreferenceActivity
161. android.preference.PreferenceCategory
162. android.preference.PreferenceFragment
163. android.preference.PreferenceGroup

164.android.preference.PreferenceManager
165.android.provider.MediaStore
166.android.provider.MediaStore.Images
167.android.provider.MediaStore.Images.ImageColumns
168.android.provider.MediaStore.MediaColumns
169.android.provider.MediaStore.Video
170.android.provider.MediaStore.Video.VideoColumns
171.android.provider.Settings
172.android.security.KeyChain
173.android.security.KeyChainAliasCallback
174.android.security.KeyChainException
175.android.service.notification.StatusBarNotification
176.android.support.v4.app.Fragment
177.android.support.v4.app.FragmentActivity
178.android.support.v4.app.FragmentManager
179.android.support.v4.app.FragmentStatePagerAdapter
180.android.support.v4.app.NotificationCompat
181.android.support.v4.content.LocalBroadcastManager
182.android.support.v4.view.MotionEventCompat
183.android.support.v4.view.ViewConfigurationCompat
184.android.support.v4.view.ViewPager
185.android.telephony.PhoneStateListener
186.android.telephony.ServiceState
187.android.telephony.SignalStrength
188.android.telephony.TelephonyManager
189.android.text.InputFilter
190.android.text.Spanned
191.android.text.TextUtils
192.android.text.format.DateFormat
193.android.text.format.Time
194.android.util.AttributeSet
195.android.util.Base64
196.android.util.Base64OutputStream
197.android.util.DisplayMetrics
198.android.util.FloatMath
199.android.util.Log
200.android.util.TypedValue
201.android.util.Xml
202.android.view.Display
203.android.view.DragEvent
204.android.view.GestureDetector
205.android.view.Gravity
206.android.view.InflateException
207.android.view.KeyEvent
208.android.view.LayoutInflater
209.android.view.Menu
210.android.view.MenuItem
211.android.view.MenuItem.OnMenuItemClickListener
212.android.view.MotionEvent
213.android.view.OrientationEventListener
214.android.view.SoundEffectConstants
215.android.view.Surface
216.android.view.SurfaceHolder
217.android.view.SurfaceView
218.android.view.VelocityTracker
219.android.view.View

220.android.view.View.DragShadowBuilder
221.android.view.View.OnClickListener
222.android.view.View.OnDragListener
223.android.view.View.OnTouchListener
224.android.view.ViewConfiguration
225.android.view.ViewGroup
226.android.view.ViewGroup.LayoutParams
227.android.view.ViewParent
228.android.view.Window
229.android.view.WindowManager
230.android.view.accessibility.AccessibilityEvent
231.android.view.animation.AccelerateInterpolator
232.android.view.animation.AlphaAnimation
233.android.view.animation.Animation
234.android.view.animation.Animation.AnimationListener
235.android.view.animation.AnimationUtils
236.android.view.animation.DecelerateInterpolator
237.android.view.inputmethod.EditorInfo
238.android.widget.AbsListView
239.android.widget.AdapterView
240.android.widget.AdapterView.OnItemClickListener
241.android.widget.AdapterView.OnItemLongClickListener
242.android.widget.ArrayAdapter
243.android.widget.BaseAdapter
244.android.widget.Button
245.android.widget.CompoundButton
246.android.widget.CompoundButton.OnCheckedChangeListener
247.android.widget.CursorAdapter
248.android.widget.EditText
249.android.widget.FrameLayout
250.android.widget.GridView
251.android.widget.HorizontalScrollView
252.android.widget.ImageView
253.android.widget.ImageView.ScaleType
254.android.widget.LinearLayout
255.android.widget.ListView
256.android.widget.PopupMenu
257.android.widget.PopupWindow
258.android.widget.ProgressBar
259.android.widget.RelativeLayout
260.android.widget.SearchView
261.android.widget.SimpleAdapter
262.android.widget.Switch
263.android.widget.TabWidget
264.android.widget.TextView
265.android.widget.Toast
266.java.beans.PropertyChangeEvent
267.java.beans.PropertyChangeListener
268.java.io.BufferedInputStream
269.java.io.BufferedOutputStream
270.java.io.BufferedReader
271.java.io.BufferedWriter
272.java.io.ByteArrayInputStream
273.java.io.ByteArrayOutputStream
274.java.io.Closeable
275.java.io.DataInputStream

276.java.io.DataOutputStream
277.java.io.File
278.java.io.FileDescriptor
279.java.io.FileInputStream
280.java.io.FileNotFoundException
281.java.io.FileOutputStream
282.java.io.FileReader
283.java.io.FileWriter
284.java.io.FilenameFilter
285.java.io.IOException
286.java.io.InputStream
287.java.io.InputStreamReader
288.java.io.ObjectInputStream
289.java.io.ObjectOutputStream
290.java.io.OutputStream
291.java.io.PrintStream
292.java.io.PrintWriter
293.java.io.RandomAccessFile
294.java.io.Serializable
295.java.io.StringWriter
296.java.io.UnsupportedEncodingException
297.java.io.Writer
298.java.lang.Math
299.java.lang.Thread.UncaughtExceptionHandler
300.java.lang.annotation.ElementType
301.java.lang.annotation.Retention
302.java.lang.annotation.RetentionPolicy
303.java.lang.annotation.Target
304.java.lang.reflect.Array
305.java.lang.reflect.Constructor
306.java.lang.reflect.Field
307.java.lang.reflect.InvocationTargetException
308.java.lang.reflect.Method
309.java.math.BigInteger
310.java.net.HttpURLConnection
311.java.net.InetAddress
312.java.net.InetSocketAddress
313.java.net.MalformedURLException
314.java.net.Socket
315.java.net.SocketException
316.java.net.URL
317.java.net.URLEncoder
318.java.net.UnknownHostException
319.java.nio.BufferOverflowException
320.java.nio.BufferUnderflowException
321.java.nio.ByteBuffer
322.java.nio.ByteOrder
323.java.nio.CharBuffer
324.java.nio.DoubleBuffer
325.java.nio.FloatBuffer
326.java.nio.IntBuffer
327.java.nio.LongBuffer
328.java.nio.ShortBuffer
329.java.nio.channels.SelectionKey
330.java.nio.channels.Selector
331.java.nio.channels.ServerSocketChannel

332.java.nio.channels.SocketChannel
333.java.nio.channels.spi.SelectorProvider
334.java.security.GeneralSecurityException
335.java.security.InvalidKeyException
336.java.security.KeyFactory
337.java.security.KeyManagementException
338.java.security.KeyPair
339.java.security.KeyPairGenerator
340.java.security.KeyStore
341.java.security.KeyStoreException
342.java.security.MessageDigest
343.java.security.NoSuchAlgorithmException
344.java.security.NoSuchProviderException
345.java.security.Principal
346.java.security.PrivateKey
347.java.security.Provider
348.java.security.SecureRandom
349.java.security.SecureRandomSpi
350.java.security.Security
351.java.security.Signature
352.java.security.SignatureException
353.java.security.UnrecoverableKeyException
354.java.security.cert.CertPathValidatorException
355.java.security.cert.Certificate
356.java.security.cert.CertificateException
357.java.security.cert.X509Certificate
358.java.security.interfaces.RSAPrivateKey
359.java.security.interfaces.RSAPublicKey
360.java.security.spec.AlgorithmParameterSpec
361.java.security.spec.X509EncodedKeySpec
362.java.text.DateFormat
363.java.text.SimpleDateFormat
364.java.util.ArrayDeque
365.java.util.ArrayList
366.java.util.Arrays
367.java.util.Calendar
368.java.util.Collection
369.java.util.Collections
370.java.util.Comparator
371.java.util.Date
372.java.util.Enumeration
373.java.util.Formatter
374.java.util.HashMap
375.java.util.HashSet
376.java.util.Hashtable
377.java.util.Iterator
378.java.util.LinkedList
379.java.util.List
380.java.util.Map
381.java.util.Queue
382.java.util.Random
383.java.util.Set
384.java.util.StringTokenizer
385.java.util.TimeZone
386.java.util.Timer
387.java.util.TimerTask

388.java.util.TreeMap
389.java.util.WeakHashMap
390.java.util.concurrent.ArrayBlockingQueue
391.java.util.concurrent.ConcurrentLinkedQueue
392.java.util.concurrent.CopyOnWriteArrayList
393.java.util.concurrent.LinkedBlockingQueue
394.java.util.regex.Pattern
395.javax.crypto.Cipher
396.javax.crypto.KeyGenerator
397.javax.crypto.SecretKey
398.javax.crypto.spec.IvParameterSpec
399.javax.crypto.spec.SecretKeySpec
400.javax.net.ssl.HandshakeCompletedEvent
401.javax.net.ssl.HandshakeCompletedListener
402.javax.net.ssl.HttpsURLConnection
403.javax.net.ssl.KeyManager
404.javax.net.ssl.SSLContext
405.javax.net.ssl.SSLException
406.javax.net.ssl.SSLHandshakeException
407.javax.net.ssl.SSLPeerUnverifiedException
408.javax.net.ssl.SSLProtocolException
409.javax.net.ssl.SSLSocket
410.javax.net.ssl.SSLSocketFactory
411.javax.net.ssl.TrustManager
412.javax.net.ssl.TrustManagerFactory
413.javax.net.ssl.X509ExtendedKeyManager
414.javax.security.auth.x500.X500Principal