Contents

1 ST Introduction (ASE_INT)

1.1 ST Reference

ST title	HyperG appGuard appShield[1] Security Target
Version	v 2.0
Author	HyperG
Date	12 Jul 2022

1.2 TOE Reference

TOE identification	HyperG appGuard appShield system version 6.6

1.3 TOE Overview

1.3.1 TOE usage and major security features

The mobile application executable and shared library are subject to the following threats

1. Extraction of the source code of the mobile application using static and/or dynamic analysis.
2. Manipulation of the mobile application executable and shared library during run-time using debugging tools.
3. Tamper of mobile application source code or executable to inject malicious code.
4. Disclosure of mobile application local data

The TOE is a software application that addresses the above threats by hardening a mobile application. The TOE hardens a mobile application executable and its shared library with:

1. Reverse engineering protection.
2. Debugging protection - run-time memory manipulation and monitoring protection.
3. Integrity protection.
4. Local data encryption.
5. Application and software library binding

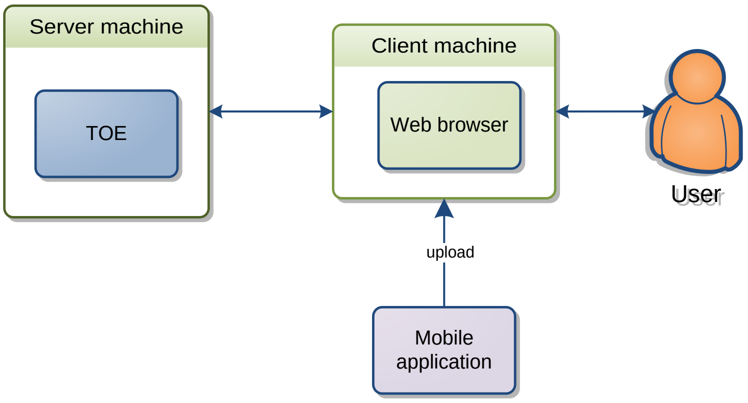


Figure 1: TOE usage

[1] appGuard refers to the product name. appShield refers to the specific product feature that is within the scope of evaluation.

The TOE is deployed in a private cloud environment (Figure 1); users can harden mobile application executable via a web browser on a client machine. The deployed protection mechanism includes:

- Identification and authentication
- Security management
- User data protection
- Cryptographic operations
- Protection of TSF

The TOE hardens mobile application that runs on Android, iOS and H5 platform. However, for the purpose of evaluation, only Android platform is evaluated.

1.3.2 TOE Type

The TOE is a software application that hardens a mobile application.

1.3.3 Required non-TOE hardware/software/firmware

The table below states the hardware and software requirements to support TOE operations.

Hardware		
Server	Processor	Quad core processors – supports hardware virtualization VT-X
	RAM	> 8 GB
	HDD	> 1 TB
Software		
OS	Ubuntu 16.04	
Database	MySQL 8.0.18	
Application Container	Docker 19.03.5	
Service scheduling	NACOS 1.1.4	
File system	minIO 2020-02-27T00:23:05Z	

Table 1: Server machine requirements

Software	
Web browser	IE8 or later
	Google Chrome
	Firefox
	Safari

Table 2: Web browser

The following sections elaborate how each non-TOE component supports TOE operations.

Figure 2 provides an overview of how the various non-TOE components interact and supports TOE operations:

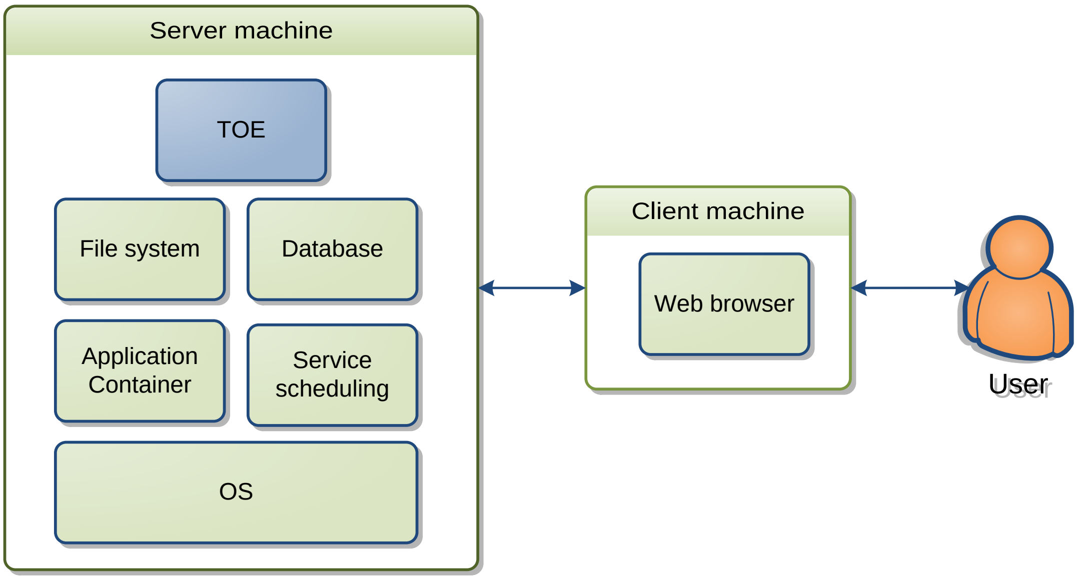


Figure 2: Non-TOE components

1.3.3.1 Operating System

The OS provides a platform on which the application container runs on.

1.3.3.2 Application Container

The application container provides the environment where the TOE runs on.

1.3.3.3 Service Scheduling

The service scheduling supports the TOE in service discovery and configuration.

1.3.3.4 Database

The database provides the framework where the TOE's TSF and user data is stored.

1.3.3.5 File System

It allows the TOE to the access the local storage of the application container.

1.4 TOE Description

1.4.1 Physical Scope

The TOE consists of one component i.e. a software application that hardens mobile applications.

Figure 2 illustrates the physical scope of the TOE.

The table below lists the TOE deliverables and their corresponding delivery methods.

Items	Description	Format	Delivery method
Preparative and operational user guidance	White box Crypto System V4.0 User Manual	PDF	Email
TOE installer	install_yyjg.tgz	CD	Delivered and installed by developer at user premise.

Table 3: TOE deliverables and delivery methods

1.4.2 Logical Scope

This section describes the logical security features of TOE.

1.4.2.1 Identification and authentication

The TOE provides the graphical user interface (GUI) for user identification and authentication via a web browser in the client machine. A TOE accepts username and password via the GUI to perform user identification and authentication.

1.4.2.2 Security management

The TOE restricts the access to security management functions to the backend administrator and frontend operator. The security management functions available includes the following:

- TSF/user data protection deployment
- cryptographic operation management
- TSF protection management

1.4.2.3 User data protection

The TOE-deployed hardening protection deploys the following methods of user data protections:

- Mobile application executable, shared library and local data are encrypted; this protects these data from static analysis.
- Before launching the mobile executable, the TOE-deployed hardening mechanism verifies
    - mobile application executable integrity
    - mobile application executable name
- The TOE-deployed mechanism protects the mobile application against extraction of intelligible information about the mobile application source code in-memory during run-time using the following techniques:
    - Randomly allocating memory locations of decrypted mobile application executable.
    - Shared libraries are erased from the memory after use.
    - Disable and monitor debug interfaces.
    - Encryption and decryption at granularity level of classes, methods and strings.

1.4.2.4 Cryptographic operations

The TOE supports the following cryptographic algorithms that are deployed on the target mobile application executable:

- AES
- SHA1

1.4.2.5 Protection of TSF

The TOE-deployed hardening mechanism reduces the risk of an attacker reverse engineer the mobile application executable and shared library to extract the source code of the mobile application using dynamic analysis. The application of white-box cryptography shall also deter attackers from obtaining the key to the encryption/decryption mechanism and hash of integrity protection mechanism.

2 Conformance Claims (ASE_CCL)

2.1 CC Conformance

The Security Target and its TOE conforms with:

- Common Criteria Information Technology Security Evaluation Version 3.1, Revision 5
  - Part 2 extended
  - Part 3 conformant**[CC3]**

2.2 PP Conformance

The Security Target and its TOE does not conform to any Protection Profile (PP).

2.3 Package Conformance

The Security Target and its TOE conforms to Evaluation Assurance Level (EAL) 2.

3 Security Problem Definition (ASE_SPD)

3.1 Introduction

This section shall define TOE's assets, subjects, external entities, and threat agent.

3.1.1 Assets

Table 4 and Table 5 define the assets that are associated to the TOE. These assets originates from trusted sources as defined by OE.Trusted_User and OE.Trusted_IT_Products. It follows that these assets are anticipated to be non-malicious.

Name	Description	Type of protection
Mobile application source code	The implementation representation that developers use to develop the mobile application.	Confidentiality
Mobile application executable	The executable file that is used by a mobile operating system (OS) to run the mobile application.	Integrity
Mobile application shared library	The software libraries that the mobile application depends on to operate.	Confidentiality
Local data used by mobile application	Local data that is accessed by the mobile application.	Confidentiality

Table 4: User data

Name	Description	Type of protection
White-box keys	Keys that are used in white-box cryptography implementation.	Confidentiality
Symmetric keys	Non-white-box keys that are used in symmetric encryption and decryption operations.	Confidentiality

Table 5: TSF data

3.1.2 Subjects

The subjects that the TOE can perceive are shown below. A TOE user is associated the subject below.

Subjects	Description
Backend administrator	The role that performs the security management functions below: • Cryptographic operation management • TSF protection
Frontend operator	The role that performs deployment of hardening mechanism on user data.

Table 6: Subjects

3.1.3 External entities

Table 7 defines the relevant external entities.

External entity	Description
System administrator	This human entity may or may not be a user of the TOE, however, it is a

	collective entity who is responsible for the setting up and management of the IT environment.
User	The human entity that uses the TOE a.k.a. TOE user.
Mobile app user	The human entity that uses the TOE-hardened mobile application.

Table 7: External entities

3.1.4 Threat agent

Table 8 defines the relevant threat agents.

Threat agent	Description
Attacker	An unauthorised human or IT entity that attempts to bypass or tamper the TOE-deployed protection mechanism of the mobile application. In turn, compromising the user data (Table 4).

Table 8: Threat agent

3.1.5 Threat scenario

Figure 3 illustrates the intended threat scenario in which the subsequent sections of SPD are based on. In summary, the attacker has no access to the TOE per se (Figure 3), instead, the attacker is expected to bypass or tamper the TOE-deployed hardening mechanism on the mobile application (Figure 4 and Figure 5). Figure 4 depicts the scenario when the mobile application is running on a mobile platform. Figure 5 depicts the scenario when the mobile application is stored in an online store.

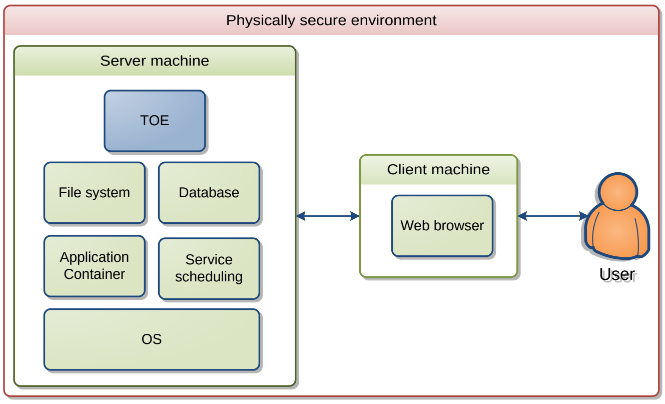


Figure 3: Threat scenario #1

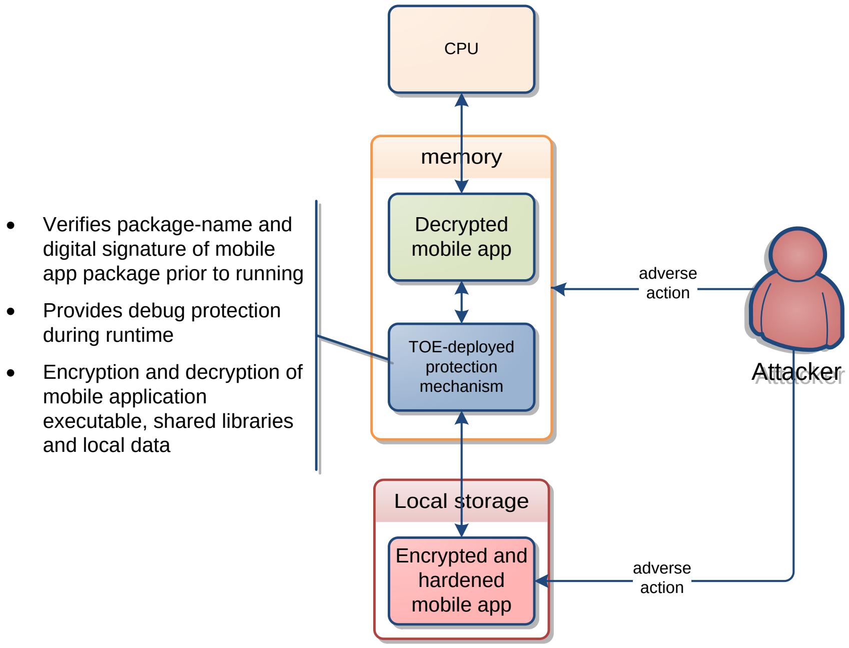

CPU
memory
Decrypted mobile app
TOE-deployed protection mechanism
- Verifies package-name and digital signature of mobile app package prior to running
- Provides debug protection during runtime
- Encryption and decryption of mobile application executable, shared libraries and local data
adverse action
Attacker
Local storage
Encrypted and hardened mobile app
adverse action

Figure 4: Threat scenario #2

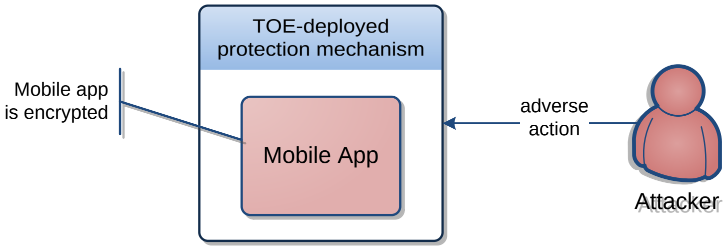


Figure 5: Threat scenario #3

3.2 Threats

Threat	Description
T.ReverseEng	An attacker may reverse engineer the mobile application executable and shared library to extract intelligible information about the mobile application source code using static and/or dynamic analysis.
T.Debugging	An attacker may modify the mobile application executable and shared library during run-time using debugging tools.
T.Integrity	An attacker may tamper the mobile application source code or executable to inject code.
T.LocalData	An attacker may disclose the mobile application local data.

Table 9: Threats

3.3 Assumptions

Assumption	Description
A.Trusted_User	TOE users are well-trained to operate the TOE securely in accordance with the operational guidance. System administrators are well-trained to setup the IT environment in accordance with the preparative guidance. Both TOE users and system administrators are trusted.
A.Trusted_CPU	The CPU and hardware peripherals on the server and client machine are

	trusted and secure i.e. in compliance with organisation's security policy.
A.Trusted_OS	The OSes that run on the server and client machine, respectively, are trusted and secure i.e. in compliance with organisation's security policy.
A.Trusted_IT_Products	The following external IT products that support the TOE operations are trusted and secure i.e. in compliance with organisation's security policy. • Server side ○ Files system ○ Database ○ Application container ○ Service scheduling • Client side ○ Web browser
A.Physical	The TOE and external IT products are deployed in physically secure environment where only authorised TOE users and system administrators have physical access. The interconnect between the server machine and client machine is physically protected from tamper. TOE is logically isolated from external network.
A.Trusted_Channel	The server machine and client machine shall establish a trusted channel.
A.Trusted_Mobile_Platform	The mobile platform, consisting of underlying hardware and mobile OS, which the TOE-hardened mobile application executable and share library are running on, is trusted.

Table 10: Assumptions

3.4 Organisation Security Policy (OSP)

OSP	Description
P.Ident_Auth	The TOE shall enforce user identification and authentication.
P.Sec_Manage	The TOE shall provide the following security management functions • Cryptographic operation management • TSF protection management

Table 11: OSP

4 Security Objectives (ASE_OBJ)

This section identifies the security objectives for the TOE, TOE-deployed hardening mechanism and the operational environment. Security objectives counters the identified threats, upholds the identified OSPs and fulfils the assumptions.

4.1 Security Objectives for the TOE.

Security Objectives	Descriptions
O.EncryptExecutable	The TOE shall encrypt the mobile application executable, including encryption granularity at levels of strings, classes and methods. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the mobile application executable during run-time.
O.ScrambleInMem	The TOE shall deploy mechanisms in the mobile application that randomly

	allocates the memory locations of decrypted mobile application executable during run-time.
O.EncryptLibrary	The TOE shall encrypt the mobile application shared library. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the shared library during run-time.
O.EraseLibraryInMem	The TOE shall deploy mechanisms in the mobile application that erases the memory decrypted shared library after use during run-time.
O.DisableDebugInterface	The TOE shall disable debug interfaces of mobile application executable.
O.MonitorDebugInterface	The TOE shall deploy mechanisms in the mobile application that monitors debug interfaces of mobile application during run-time.
O.Integrity	The TOE shall deploy mechanisms in the mobile application that protects the integrity of the mobile application executable and shared library. The TOE shall deploy mechanisms in the mobile application that binds the mobile application executable and shared library to the TOE-deployed protection mechanism.
O.EncryptLocalData	The TOE shall encrypt the mobile application local data. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the local data during run-time.
O.WhiteBoxCrypto	The TOE shall deploy white-box cryptography in the mobile application to hide all white-box keys. The TOE shall protect the confidentiality of all symmetric keys and hash using white-box cryptography.
O.Ident_Auth	The TOE shall enforce user identification and authentication.
O.Sec_Manage	The TOE shall provide the following security management functions • Cryptographic operation management • TSF protection management

Table 12: Security Objectives for TOE

4.2 Security Objectives for the Operational Environment

Security Objectives	**Descriptions**
OE.Trusted_User	The operational environment shall ensure: • TOE users are well-trained to operate the TOE securely in accordance with the operational guidance. • System administrators are well-trained to setup the IT environment in accordance with the preparative guidance. • Both TOE users and system administrators are trusted.
OE.Trusted_CPU	The System Administrator shall ensure the CPU and hardware peripherals on the server and client machine are trusted and secure i.e. in compliance with organisation's security policy.
OE.Trusted_OS	The System Administrator shall ensure the server and client machine, respectively, are trusted and secure i.e. in compliance with organisation's security policy.
OE.Trusted_IT_Products	The System Administrator shall ensure the following external IT products that support the TOE operations are trusted and secure i.e. in compliance with organisation's security policy.

	- Server side - Files system - Database - Application container - Service scheduling - Client side - Web browser
OE.Physical	The System Administrator shall ensure the: - TOE and external IT products are deployed in the same physically secure environment where only authorised TOE users and system administrators have access. - interconnect between the server machine and client machine is physically protected from tamper. - TOE is logically isolated from external network.
OE.Trusted_Channel	The System Administrator shall ensure the following: - The server machine and client machine shall establish a trusted channel.
OE.Trusted_Mobile_Platform	The TOE user shall inform the Mobile app user to ensure the following: - The mobile platform, consisting of underlying hardware and mobile OS, which the TOE-hardened mobile application executable and share library are running on, is trusted.

Table 13: Security Objectives for Operational Environment

4.3 Security Objective Rationale

4.3.1 Tracing between security objectives and security problem definition

Threats-OSPs-Assumptions / Security Objectives	O.EncryptExecutable	O.ScrambleInMem	O.EncryptLibrary	O.EraseLibraryInMem	O.DisableDebugInterface	O.MonitorDebugInterface	O.Integrity	O.EncryptLocalData	O.WhiteBoxCrypto	OE.Trusted_User	O.Ident_Auth	O.Sec_Manage	OE.Trusted_CPU	OE.Trusted_OS	OE.Trusted_IT_Products	OE.Physical	OE.Trusted_Channel	OE.Trusted_Mobile_Platform
T.ReversEng	x	x	x	x	x	x			x									
T.Debugging					x	x			x									
T.Integrity							x		x									x
T.LocalData					x	x		x	x									
A.Trusted_User										x								
A.Trusted_CPU													x					
A.Trusted_OS														x				
A.Trusted_IT_Products															x			
A.Physical																x		
A.Trusted_Channel																	x	

A.Trusted_Mobile_Platform																		x
P.Ident_Auth											x							
P.Sec_Manage												x						

Table 14: Tracing between security objectives and SPD

4.3.2 Justification for tracing

This section explains the tracing illustrated in Table 14.

4.3.2.1 Threats-Security Objective Justification

T.ReversEng	An attacker may reverse engineer the mobile application executable and shared library to extract the source code of the mobile application using static and/or dynamic analysis.
O.EncryptExecutable	The TOE encrypts the mobile application executable as well as at granularity levels of strings, classes and methods. This removes the risk of attackers performing static analysis to reverse engineer the mobile application executable to obtain the source code and understand the program flow. By encrypting at granularity levels of strings, classes and methods, the TOE reduces the exposure of the mobile application executable as it is decrypted and run in memory. In turn, this reduces the risk of attacker performing dynamic analysis to reverse engineer the executable to obtain intelligible information about the source code during run-time.
O.ScrambleInMem	This increases the difficulty for attackers in their attempt to disclose and obtain intelligible information about the original source code.
O.EncryptLibrary	This removes the risk of attackers performing static analysis to reverse engineer the mobile application shared library to disclose its operations.
O.EraseLibraryInMem	This reduces the risk of attackers performing dynamic analysis to reverse engineer the mobile application shared library to obtain intelligible information about its operations.
O.DisableDebugInterface	This removes the risk of attacker performing debugging on the mobile application executable during run-time, which in turn, may allow the attacker to gain intelligible information about the mobile application source code.
O.MonitorDebugInterface	This mitigates **O.DisableDebugInterface** in case the attacker manages to enable the debug interface.
O.WhiteBoxCrypto	This reduces the risk of disclosing the symmetric keys, hashes and white-box keys.

T.Debugging	An attacker may manipulate the mobile application executable and shared library during run-time using debugging tools.
O.DisableDebugInterface	This removes the risk of attacker performing debugging on the mobile application executable during run-time, which in turn, may allow the attacker to gain intelligible information about the mobile application

	source code.
O.MonitorDebugInterface	This mitigates **O.DisableDebugInterface** in case the attacker manages to enable the debug interface.
O.WhiteBoxCrypto	This reduces the risk of disclosing the symmetric keys, hashes and white-box keys.

T.Integrity	An attacker may tamper the mobile application source code or executable to inject malicious code.
O.Integrity	This removes risk of tamper on mobile application executable and shared library.
O.WhiteBoxCrypto	This reduces the risk of disclosing the symmetric keys, white-box keys and hashes. By protecting the confidentiality of hashes used for integrity verification, it reduces the risk of an attacker modifying the mobile application executable, shared library and hashes at the same time.
OE.Trusted_Mobile_Platform	The mobile platform shall verify signature of the hardened mobile application prior to launching it. This removes risk of tamper on mobile application executable and shared library.

T.LocalData	An attacker may disclose the mobile application local data.
O.EncryptLocalData	This reduces risk of disclosing the mobile application data.
O.DisableDebugInterface	This removes the risk of attacker performing debugging on the mobile application executable during run-time, which in turn, may allow the attacker to gain intelligible information about the mobile application source code.
O.MonitorDebugInterface	This mitigates **O.DisableDebugInterface** in case the attacker manages to enable the debug interface.
O.WhiteBoxCrypto	This reduces the risk of disclosing the symmetric keys, hashes and white-box keys.

4.3.2.2 Assumptions-Security Objective Justification

A.Trusted_User	TOE users are well-trained to operate the TOE securely in accordance with the operational guidance. System administrators are well-trained to setup the IT environment in accordance with the preparative guidance. Both TOE users and system administrators are trusted.
OE.Trusted_User	This directly upholds the assumption.

A.Trusted_CPU	The CPU and hardware peripherals on the server and client machine that the Windows Server and Windows Oses run on, respectively, are trusted and secure i.e. in compliance with organisation's security policy.

OE.Trusted_CPU	This directly upholds the assumption.
A.Trusted_OS	The Windows Server and Windows Oses that runs on the server and client machine, respectively, are trusted and secure i.e. in compliance with organisation's security policy.
OE.Trusted_OS	This directly upholds the assumption.
A.Trusted_IT_Products	The external IT products that support the TOE operations are trusted and secure i.e. in compliance with organisation's security policy.
OE.Trusted_IT_Products	This directly upholds the assumption.
A.Physical	The TOE and external IT products are deployed in the same physically secure environment where only authorised TOE users and system administrators have access.
OE.Physical	This directly upholds the assumption.
A.Trusted_Channel	Trust channel is established for internal TOE transfer and inter TSF transfer.
OE.Trusted_Channel	This directly upholds the assumption.
A.Trusted_Mobile_Platform	The mobile platform, consisting of underlying hardware and mobile OS, which the TOE-hardened mobile application executable and share library are running on, is trusted.
OE. Trusted_Mobile_Platform	This directly upholds the assumption.

4.3.2.3 OSP-Security Objective Justification

P.Ident_Auth	The TOE shall enforce user identification and authentication.
O.Ident_Auth	This directly upholds the OSP.
P.Sec_Manage	The TOE shall provide the following security management functions • Cryptographic operation management • TSF protection management
O.Sec_Manage	This directly upholds the OSP.

5 Extended Component Definition (ASE_ECD)

This Security Target uses components defined as extensions to CC Part 2[CC2]. The component FPT_MUL is a new component to be used for application evaluation used in mobile environment.

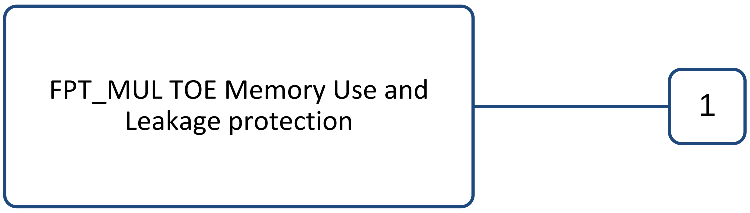
5.1 Definition of the Family FPT_MUL

The family FPT_MUL (TOE Memory Use and Leakage protection) of the Class FPT (Protection of TSF) is defined here to describe the IT security functional requirements of the TOE related to leakage of information based on inspection of memory use during runtime operation and reverse engineering of TOE at rest. The TOE shall prevent attacks against the TOE and other secret data where the attack is based on deduction of useful information from analysis of memory use and reverse engineering of TOE at rest. This family describes the functional requirements for the limitation of intelligible exploitation of usage of memory for runtime operation and protection against TOE reverse engineering at rest which are not directly addressed by any other component of CC Part 2[CC2].

5.1.1 Family Behaviour

This family defines requirements to mitigate intelligible exploitation of usage of memory for runtime operation.

5.1.2 Component Levelling



FPT_MUL.1 TOE Memory Use and Leakage protection related to TSF and user data.

5.1.3 Management

FPT_MUL.1 There are no management activities foreseen.

5.1.4 Audit

FPT_MUL.1 There are no actions defined to be auditable.

FPT_MUL.1 TOE Memory Use and Leakage protection

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_MUL.1.1 The TSF shall avoid leakage of data computed in memory during runtime operation in excess of **[assignment: specified limits]** enabling access to **[assignment: list of types of TSF data]** and **[assignment: list of types of user data]**.

FPT_MUL.1.2 The TSF shall deter inspection of memory usage by protecting simple usage of memory trace to gain access to **[assignment: list of types of TSF data]**and **[assignment: list of types of user data]**.

FPT_MUL.1.3 The TSF shall avoid leakage of data at rest in excess of **[assignment: specified limits]** enabling access to **[assignment: list of types of TSF data]** and **[assignment: list of types of user data]**.

6 Security Requirements (ASE_REQ)

This chapter gives the security functional requirements and the security assurance requirements for the TOE.

Security functional requirements components are stated in section 5.1 Security Functional Requirements. Security assurance components are stated section 5.2 Security Assurance Requirements in are drawn from Common Criteria Part 3**[CC3]**.

Operations for iteration, assignment, selection and refinement have been made. The following textual conventions are used in this chapter as part of every SFR:

- Iteration is represented by a slash ('/') followed by an identifier placed at the end of the component. For example, FDP_ACF.1/Signer.
- Assignment is represented by **bold text**.
- Selection is represented by *italic text*.
- Refinement is represented by <u>underlined text</u>.

6.1 Security Functional Requirements

6.1.1 Identification and authentication

6.1.1.1 FIA_UID (User identification)

FIA_UID.2 **User identification before any action**

Hierarchical to: FIA_UID.1 Timing of identification

Dependencies: No dependencies.

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.1.1.2 FIA_UAU (User authentication)

FIA_UAU.2 **User authentication before any action**

Hierarchical to: FIA_UAU.1 Timing of authentication

Dependencies: FIA_UID.1 Timing of identification

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

6.1.2 Security management

6.1.2.1 FMT_SMR (Security management roles)

FMT_SMR.1 **Security roles**

Hierarchical to: No other components.

Dependencies: FIA_UID.1 Timing of identification

FMT_SMR.1.1 The TSF shall maintain the roles **backend administrator and frontend operator**.

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

6.1.2.2 FMT_SMF (Specification of management functions)

FMT_SMF.1	**Specification of Management Functions**
	Hierarchical to: No other components.
	Dependencies: No dependencies.

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

- **TSF/user data protection deployment**
- **cryptographic operation management**
- **TSF protection management**

6.1.2.3 FMT_MOF (Management of function in TSF)

FMT_MOF.1/backend	**Management of security functions behaviour**
	Hierarchical to: No other components.
	Dependencies: FMT_SMR.1 Security roles FMT_SMF.1 Specification of Management Functions

FMT_MOF.1.1/backend The TSF shall restrict the ability to *disable, enable* the functions **cryptographic operation**s **and TSF protection** to **backend administrator**.

FMT_MOF.1/frontend	**Management of security functions behaviour**
	Hierarchical to: No other components.
	Dependencies: FMT_SMR.1 Security roles FMT_SMF.1 Specification of Management Functions

FMT_MOF.1.1/frontend The TSF shall restrict the ability to *disable, enable* the functions **TSF/user data protection deployment** to **frontend operator**.

6.1.3 User data protection

6.1.3.1 FDP_RIP (Residual information protection)

FDP_RIP.1	**Subset residual information protection**
	Hierarchical to: No other components.
	Dependencies: No dependencies.

FDP_RIP.1.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the *deallocation of the resource from* the following objects: **mobile application shared library**.

Application notes: The TSF shall destroy the memory content of memory resources that have been deallocated from shared library.

6.1.3.2 FDP_SDI (Store data integrity)

FDP_SDI.2	**Stored data integrity monitoring and action**
	Hierarchical to: FDP_SDI.1 Stored data integrity monitoring
	Dependencies: No dependencies.

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for

integrity errors on all objects, based on the following attributes:

- **mobile application executable name**
- **mobile application executable tag according to FCS_COP.1/SHA**

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **not launch mobile application executable**.

Application notes: In the context of TOE usage, the integrity protection of mobile application depends on second pre-image resistance property of FCS_COP.1/SHA.

6.1.4 Cryptographic operations

6.1.4.1 FCS_COP (Cryptographic operation)

FCS_COP.1/AES **Cryptographic operation**

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/AES The TSF shall perform **encryption and decryption** in accordance with a specified cryptographic algorithm **AES-CBC** and cryptographic key sizes **256 bits** that meet the following: **FIPS 140-2**.

Application notes: The TOE encrypts mobile application executable, shared library and local data to safeguard against T.ReverseEng , T.Debugging and T.LocalData.

FCS_COP.1/WB **Cryptographic operation**

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/WB The TSF shall perform **encryption and decryption** in accordance with a specified cryptographic algorithm **AES-CBC** and cryptographic key sizes **256 bits** that meet the following: **FIPS 140-2**.

Application notes: FCS_COP.1/WB encrypts symmetric keys used in FCS_COP.1/AES. Correspondingly, the cryptographic key of FCS_COP.1/WB is protected by FPT_MUL.1.

FCS_COP.1/SHA **Cryptographic operation**

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/SHA The TSF shall perform **integrity verification** in accordance with a specified cryptographic algorithm **SHA1** and cryptographic key sizes **none** that meet the following: **FIPS 180-4**.

Application notes: Tag of FCS_COP.1/SHA is one of the security attributes described in FDP_SDI.1. The hash is encrypted by FCS_COP.1/WB. Correspondingly, the cryptographic key of FCS_COP.1/WB is protected by FPT_MUL.1. The hashing is done over the FCS_COP.1/AES-encrypted mobile application package and shared library.

6.1.4.2 FCS_CKM (Cryptographic key destruction)

FCS_CKM.4/AES **Cryptographic key destruction**

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1/AES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **zeroization** that meets the following: **none**

Application notes: This SFR shall destroy the in-memory symmetric keys that are used by FCS_COP.1/AES during runtime when the mobile application is no long running.

6.1.5 Protection of TSF

6.1.5.1 FPT_MUL.1 (TOE Memory Use and Leakage protection)

FPT_MUL.1 **TOE Memory Use and Leakage protection**

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_MUL.1.1 The TSF shall avoid leakage of data computed in memory during runtime operation in excess of **non-intelligible information** enabling access to **white-box keys** and **mobile application source code, shared library**.

FPT_MUL.1.2 The TSF shall deter inspection of memory usage by protecting simple usage of memory trace to gain access to **white-box keys** and **mobile application source code, shared library**.

FPT_MUL.1.3 The TSF shall avoid leakage of data at rest in excess of **non-intelligible information** enabling access to **white-box keys** and **mobile application source code, shared library**.

Application notes: During runtime, the mobile application source code and shared library are obfuscated by random memory allocation. The white-box cryptography implementation shall also hide the white-box key during runtime and at rest. Debug protection shall deter inspection of memory usage during runtime.

6.2 Security Assurance Requirements

The assurance level for this TOE is EAL2

Assurance Class	Assurance Components
ADV: Development	ADV_ARC.1 Security architecture description
	ADV_FSP.2 Security-enforcing functional specification
	ADV_TDS.1 Basic design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.2 Use of a CM system
	ALC_CMS.2 Parts of the TOE CM coverage
	ALC_DEL.1 Delivery procedures
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction
	ASE_OBJ.2 Security objectives
	ASE_REQ.2 Derived security requirements
	ASE_SPD.1 Security problem definition
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.1 Evidence of coverage
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

Table 15: Assurance requirements for EAL2

6.3 Security Requirement Rationale

6.3.1 Tracing between SFR and security objectives of TOE

SFR/Security Objectives	O.EncryptExecutable	O.ScrambleInMem	O.EncryptLibrary	O.EraseLibraryInMem	O.DisableDebugInterface	O.MonitorDebugInterface	O.Integrity	O.EncryptLocalData	O.WhiteBoxCrypto	O.Ident_Auth	O.Sec_Manage
FDP_RIP.1				x							
FDP_SDI.2							x				
FCS_COP.1/AES	x		x					x			
FCS_CKM.4/AES	x		x					x			
FCS_COP.1/WB									x		
FCS_COP.1/SHA							x				
FPT_MUL.1	x	x	x	x	x	x			x		
FIA_UID.2										x	
FIA_UAU.2										x	
FMT_SMR.1											x
FMT_SMF.1											x
FMT_MOF.1/backend											x
FMT_MOF.1/frontend											x

Table 16: Tracing between SFR and security objectives of TOE

6.3.2 Justification for tracing

The following section provides justification for the tracing in Table 13.

O.EncryptExecutable	The TOE shall encrypt the mobile application executable, including encryption granularity at levels of strings, classes and methods. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the mobile application executable during run-time.
FCS_COP.1/AES	This SFR encrypts and decrypts the mobile application executable and its corresponding strings, classes and methods during run-time
FCS_CKM.4/AES	This SFR destroys all in-memory symmetric keys related to FCS_COP.1/AES when the mobile application is no longer running.
FPT_MUL.1	This SFR avoids intelligible leakage of data in memory which will enable

	access to white-box keys and mobile application source code and its shared library.
O.ScrambleInMem	The TOE shall deploy mechanisms in the mobile application that randomly allocates the memory locations of decrypted mobile application executable during run-time.
FPT_MUL.1	This SFR avoids intelligible leakage of data in memory which will enable access to white-box keys and mobile application source code and its shared library.
O.EncryptLibrary	The TOE shall encrypt the mobile application shared library. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the shared library during run-time.
FCS_COP.1/AES	These SFRs encrypt and decrypt the share library during run-time.
FCS_CKM.4/AES	This SFR destroys all in-memory symmetric keys related to FCS_COP.1/AES when the mobile application is no longer running.
FPT_MUL.1	This SFR avoids intelligible leakage of data in memory which will enable access to white-box keys and mobile application source code and its shared library.
O.EraseLibraryInMem	The TOE shall deploy mechanisms in the mobile application that erases the memory decrypted shared library after use during run-time.
FDP_RIP.1	This SFR ensures that shared libraries that were previously running in the memory were erased after use. In turn, this helps to support **FPT_MUL.1** at the same time.
FPT_MUL.1	This SFR avoids intelligible leakage of data in memory which will enable access to white-box keys and mobile application source code and its shared library.
O.DisableDebugInterface	The TOE shall disable debug interfaces of mobile application executable
FPT_MUL.1	This SFR deters inspection of memory usage via debugging interfaces to gain access to white-box keys and mobile application executable.
O.MonitorDebugInterface	The TOE shall deploy mechanisms in the mobile application that monitors debug interfaces of mobile application during run-time.
FPT_MUL.1	This SFR deters inspection of memory usage via debugging interfaces to gain access to white-box keys and mobile application executable.
O.Integrity	The TOE shall deploy mechanisms in the mobile application that protects the integrity of the mobile application executable and shared library.

	The TOE shall deploy mechanisms in the mobile application that binds the mobile application executable and shared library to the TOE-deployed protection mechanism.
FDP_SDI.2 FCS_COP.1/SHA	This SFR ensures that the integrity of mobile application executable and shared library are maintained. Upon detection of the tamper, the TSF shall stop the launching of the mobile application executable.
O.EncryptLocalData	The TOE shall encrypt the mobile application local data. The TOE shall also deploy mechanisms in the mobile application that encrypts and decrypts the local data during run-time.
FCS_COP.1/AES	These SFRs encrypt and decrypt the mobile application local data during run-time.
FCS_CKM.4/AES	This SFR destroys all in-memory symmetric keys related to FCS_COP.1/AES when the mobile application is no longer running.
O.WhiteBoxCrypto	The TOE shall deploy white-box cryptography in the mobile application to hide all white-box keys. The TOE shall protect the confidentiality of all symmetric keys and hash using white-box cryptography.
FCS_COP.1/WB	This SFR shall protect the confidentiality of symmetric keys and hash using white-box cryptography.
FPT_MUL.1	This SFR avoids intelligible leakage of data in memory and at rest which will enable access to white-box keys and mobile application source code and its shared library.
O.Ident_Auth	The TOE shall enforce user identification and authentication.
FIA_UID.2 FIA_UAU.2	These SFRs enforces user identification and authentication.
O.Sec_Manage	The TOE shall provide the following security management functions • TSF/user data protection deployment • Cryptographic operation management • TSF protection management
FMT_SMR.1 FMT_SMF.1 FMT_MOF.1/backend FMT_MOF.1/frontend	These SFRs enforces the following security management functions • TSF/user data protection deployment • Cryptographic operation management • TSF protection management

6.3.3 SFR Dependency Fulfilment

SFR	Dependencies	Fulfilment
FIA_UID.2	No dependencies	Not applicable

FIA_UAU.2	FIA_UID.1	FIA_UID.2 is hierarchical to FIA_UID.1
FMT_SMR.1	FIA_UID.1	FIA_UID.2 is hierarchical to FIA_UID.1
FMT_SMF.1	No dependencies.	Not applicable
FMT_MOF.1/backend	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MOF.1/frontend	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FDP_RIP.1	No dependencies.	Not applicable
FDP_SDI.2	No dependencies.	Not applicable
FCS_COP.1/AES	[FDP_ITC.1, FDP_ITC.2 or FCS_CKM.1]	TOE depends on the external IT components for key generation. **OE.Trusted_IT_Products** provides the key generation function, hence, FCS_CKM.1 is not required..
	FCS_CKM.4	FCS_CKM.4/AES
FCS_COP.1/WB	[FDP_ITC.1, FDP_ITC.2 or FCS_CKM.1]	TOE depends on the external IT components for key generation. **OE.Trusted_IT_Products** provides the key generation function, hence, FCS_CKM.1 is not required.
	FCS_CKM.4	The white-box key is embedded as part of the cryptographic implementation. Hence, the cryptographic key shall never be destroyed.
FCS_COP.1/SHA	[FDP_ITC.1, FDP_ITC.2 or FCS_CKM.1]	There is no key required for hash computation. Hence, these dependencies are not required.
	FCS_CKM.4	There is no key required for hash computation. Hence, this dependency is not required.
FPT_MUL.1	No dependencies.	Not applicable

Table 17: SFR dependency fulfilment

6.3.4 Rationale for EAL2

The assurance level for this protection profile is EAL2. EAL2 allows a developer to attain a reasonably high assurance level without the need for highly specialized processes and practices. It is the highest level that could be applied to an existing product line without undue expense and complexity. As such, EAL2 is appropriate for commercial products that can be applied to basic security functions.

7 TOE Summary Specification (ASE_TSS)

7.1 Identification and authentication

FIA_UID.2 User identification before any action	The TOE performs user identification and authentication using username and password before allow user access to any TSF-mediated action
FIA_UAU.2 User authentication before any action	

7.2 Security management

FMT_SMR.1 Security roles	The TOE maintains on security management role i.e. backend administrator.
FMT_SMF.1 Specification of Management Functions	The TOE provides the following security management functions • TSF/user data protection deployment • cryptographic operation management • TSF protection management
FMT_MOF.1/backend Management of security functions behaviour	The TOE restricts the following security management functions to the backend administrator • cryptographic operation management • TSF protection management
FMT_MOF.1/frontend Management of security functions behaviour	The TOE restricts the following security management functions to the frontend operator • TSF/user data protection deployment

7.3 User data protection

FDP_RIP.1 Subset residual information protection	The TOE-deployed hardening mechanism erases shared library from memory after use during run-time.
FDP_SDI.2 Stored data integrity monitoring and action	The TOE-deployed hardening mechanism verifies the mobile application executable name and integrity before launching it. If the verification fails, the hardening mechanism shall not launch the mobile executable.

7.4 Cryptographic Operations

FCS_COP.1/AES Cryptographic operation	The TOE encrypts the mobile application executable, its shared library and local data with these cryptographic algorithms. The TOE-deployed mechanism implements these cryptographic algorithms to perform encryption and decryption of mobile application executable, its shared library and local data during run-time.
FCS_CKM.4/AES Cryptographic key destruction	The TOE-deployed mechanism destroys in-memory symmetric keys when the mobile application is no longer running.
FCS_COP.1/SHA Cryptographic operation	The TOE hashes the mobile application executable and its shared library with this cryptographic algorithm. The TOE-deployed mechanism implements this cryptographic algorithm to verify the integrity of mobile application executable and its shared library during run-time.

Table 18: SFR related to Cryptographic Operation

7.5 Protection of TSF

FPT_MUL.1 TOE Memory Use and Leakage protection	The TOE-deployed mechanism protects the mobile application against extraction of intelligible information about the mobile application source code in-memory during run-time using the following techniques: • Randomly allocating memory locations of decrypted mobile application executable. • Shared libraries are erased from the memory after

	use. - Monitor debug interfaces. - Encryption and decryption at granularity level of classes, methods and strings. - Applying white-box cryptography to hide symmetric keys and hashes.

Table 19: SFR related to Protection of TSF

8 References

[CC1] Common Criteria Information Technology Security Evaluation, Part 1: Introduction and general model, April 2017, Version 3.1, Revision 5

[CC2] Common Criteria Information Technology Security Evaluation, Part 2: Security Functional Components, April 2017, Version 3.1, Revision 5

[CC3] Common Criteria Information Technology Security Evaluation, Part 3: assurance components, April 2017, Version 3.1, Revision 5

9 Glossary

Compromise — The unauthorized disclosure, modification, substitution or use of sensitive data (including plaintext cryptographic keys and other CSPs).

Confidentiality — The property that sensitive information is not disclosed to unauthorized individuals, entities or processes.

Digital signature — A non-forgeable transformation of data that allows proof of the source (with non-repudiation) and verification of the integrity of that data.

Firmware — The programs and data stored in hardware (e.g., ROM, PROM, or EPROM) such that the programs and data cannot be dynamically written or modified during execution. Hardware: the physical equipment used to process programs and data in a CIMC.

Integrity — The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner.

Password — A string of characters (letters, numbers, and other symbols) used to authenticate an identity or to verify access authorization.

Plaintext key — An unencrypted cryptographic key.

Private key — A cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity, and not made public.

Protection Profile — An implementation-independent set of security requirements for a category of Targets of Evaluation (TOEs) that meet specific consumer needs.

Security policy — A precise specification of the security rules under which a CIMC shall operate, including the rules derived from the requirements of this document and additional rules imposed by the vendor.

Software — The programs and associated data that can be dynamically written and modified.

Target of Evaluation (TOE) — An information technology product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.

TOE Security Functions (TSF) — A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.

TOE Security Policy (TSP) — A set of rules that regulate how assets are managed, protected, and distributed within a TOE.

10 Acronym

CC	Common Criteria
CEM	Common Evaluation Methodology
EAL	Evaluation Assurance Level
GUI	Graphical User Interface
IP	Internet Protocol
ISO	International Organization for Standardization
OS	Operating System
PP	Protection Profile
RFC	Request for Comment
SAR	Security Assurance Requirement
SF	Security Functions
SFP	Security Functions Policy
SFR	Security Functional Requirement
SQL	Structured Query Language
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
VM	Virtual Machine