EUROPEAN UNION AGENCY
FOR CYBERSECURITY



EUCC SCHEME

GUIDELINES

CSP-based evaluation in the context of Secure Elements

Version 1, November 2026

DOCUMENT HISTORY

Date	Version	Modification	Author's comments
November 2026	V1		Version for publication

LEGAL NOTICE

LEGAL NOTICE

This publication is a guidelines document supporting Commission Implementing Regulation (EU) 2024/482.

This document is established with the support of the European Cybersecurity Certification Group (ECCG) sub-group on EUCC maintenance and review (EsEm) and may be updated whenever needed to reflect the developments and best practices in the field of CSP-based evaluations.

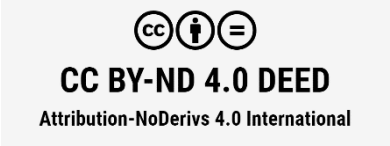
This document should be read in conjunction with Regulation (EU) 2019/881, the Commission Implementing Regulation (EU) 2024/482, its annexes, and where applicable supporting documentation that is made available.

This document is made publicly accessible through the EU cybersecurity certification website and is free of charge.

ENISA is not responsible or liable for the use of the content of this document. Neither ENISA nor any person acting on its behalf or on behalf for the maintenance of the scheme is responsible for the use that might be made of the information contained in this publication.

COPYRIGHT NOTICE

© European Union Agency for Cybersecurity (ENISA), 2026



This publication is licensed under CC-BY-ND 4.0 DEED. Making copies and redistributing this document is permitted. (https://creativecommons.org/licenses/by-nd/4.0/).

CONTACT

Feedback or questions related to this document can be sent via the European Union Certification website Ask questions, find answers - European Union Cybersecurity Certification

TABLE OF CONTENTS

1. INTRODUCTION 6

1.1 OBJECTIVE 6

1.2 REFERENCES 6

1.2.1 Normative references 6
1.2.2 Informative references 7

1.3 ACRONYMS AND DEFINITIONS 7

1.3.1 Acronyms 7
1.3.2 Definitions 8

2. OVERVIEW 10

2.1 Problem description 10

2.2 Objective 10

2.3 Concept of the methodology 10

2.4 Scope 11

2.5 CSP types 12

2.6 Security assurance and composition 14

2.7 Product or platform change 15

2.8 Certificate And Certification Report 16

2.9 Benefits of this methodology 16

3. EVALUATION ACTIVITIES 17

3.1 STEP 1: PREPARATION PHASE 17

3.2 STEP 2: TEST SETUP 17

3.3 STEP 3: SECURITY ASSURANCE EVALUATION (SAR) 17

3.4 STEP 4: COMPOSITION EVALUATION (ACO) 17

3.5 STEP 5: FINAL ASSESSMENT AND CERTIFICATION 17

4. EVALUATION INPUT MATERIAL 18

4.1 REQUIRED SPECIFICATIONS 18

4.2 REQUIRED TOE DELIVERABLES 18

4.3 REQUIRED TESTING DELIVERABLES 19

5. SAR REFINEMENTS 21

5.1 ASE – SECURITY TARGET EVALUATION 22

5.1.1 ASE_SPD – Security problem definition 22

5.1.2 ASE_TSS – TOE summary specification 22

5.2 ADV – DEVELOPMENT 22

5.2.1 ADV_IMP – Implementation representation 22

5.2.2 ADV_TDS – TOE design 23

5.3 ALC – LIFE-CYCLE SUPPORT 23

5.3.1 ALC_CMS – Configuration Management Scope 23

5.3.2 ALC_CMC – Configuration Management Capabilities 23

5.4 AGD – GUIDANCE DOCUMENTS 24

5.4.1 AGD_PRE – Preparative procedures 24

5.5 ATE – TESTS 24

5.5.1 ATE_FUN – Functional Tests 24

5.5.2 ATE_IND – Independent Testing 24

5.6 AVA – VULNERABILITY ASSESSMENT 25

5.6.1 AVA_VAN – Vulnerability Analysis 25

6. ACO REFINEMENTS 26

6.1 ACO_COR COMPOSITION RATIONALE 26

6.2 ACO_DEV DEVELOPMENT EVIDENCE 26

6.3 ACO_REL RELIANCE ON DEPENDENT COMPONENT 27

6.4 ACO_CTT COMPOSED TOE TESTING 27

6.5 ACO_VUL COMPOSITION VULNERABILITY ANALYSIS 28

7. ADDITIONAL WORK UNITS 29

7.1 SAR-RELATED WORK UNITS 29

7.1.1 Action ASE_SPD.1.1E 29
7.1.2 Action ASE_TSS.1.1E 29
7.1.3 Action ADV_IMP.1.1E 30
7.1.4 Action ADV_TDS.3.1E 30
7.1.5 Action ALC_CMS.2.1E 31
7.1.6 Action ALC_CMC.2.1E 31
7.1.7 Action AGD_PRE.1.1E 32
7.1.8 Action ATE_FUN.2.1E 32
7.1.9 Action ATR_IND.1.2E 32

7.2 ACO-RELATED WORK UNITS 32

7.2.1 Action ACO_COR.1.1E 32
7.2.2 Action ACO_DEV.2.2E 33
7.2.3 Action ACO_REL.1.1E 33
7.2.4 Action ACO_CTT.2.1E 34

8. COMBINED CERTIFICATE AND CERTIFICATION REPORT 35

8.1 COMBINED CERTIFICATE CONTENT 35

8.2 COMBINED CERTIFICATION REPORT CONTENT 35

A ANNEX 1: REQUIREMENTS FOR CSPs 37

B ANNEX 2: SAMPLE CSP SERVICES 40

C ANNEX 3: SAMPLE CSP CONFIGURATION 41

D ANNEX 4: SAMPLE CSP MULTI APPLICATIONS SUPPORT 43

E ANNEX 5: SAMPLE CSP USER-LEVEL AUTHORISATION 44

F ANNEX 6: SAMPLE CSP PLATFORM DEPENDENCIES 47

G ANNEX 7: SAMPLE CSP WITHIN eUICC 48

1. INTRODUCTION

1.1 OBJECTIVE

This guidelines document supporting the EUCC scheme provides guidance for the evaluation of applications using a Cryptographic Service Provider (CSP) and describes a methodology for such evaluations.

The use of a CSP addresses the scaling problem of the classical smart card composite approach when applications are intended to run on a variety of chip platforms. Unlike a cryptographic library, a CSP provides a defined set of complete and encapsulated security services such as protocols, signature creation, authentication via a harmonised API. The harmonised API realises a kind of an abstraction layer so that different CSP implementations on different chip platforms present the same behaviour to the applications.

The CSP implementations need to be certified at EAL4 augmented with AVA_VAN.5 using the classical smartcard composite evaluation approach. The advantage of the use of a harmonised CSP is that the applications do not need to implement highly critical security services, e.g., processing of secret keys. This may both:

- help risk owners to accept a lower assurance level for the application.
- avoid going for composite evaluations for each chip platform configuration intended to host the application. Instead of that, one composition certification according to CC class ACO can be conducted with a representative CSP implementation.

Different CSP types with different sets of security services and different API are possible. Within one CSP type, the implemented security services, and the API need to be the same for the different implementations on different chip platforms.

1.2 REFERENCES

1.2.1 Normative references

Regulations

[CSA] Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act).

[EUCC] Implementing Regulation (EU) 2024/482 on establishing the Common Criteria-based cybersecurity certification scheme (EUCC)[1].

Standards

[CC2022-1R1] ISO/IEC 15408-1:2022 - Information security, cybersecurity and privacy protection - Evaluation criteria for IT security - Part 1: Introduction and general model.

[CC2022-3R1] ISO/IEC 15408-3:2022 - Information security, cybersecurity and privacy protection - Evaluation criteria for IT security - Part 3: Security assurance components.

EUCC state-of-the-art documents

[COMP1] Composite product evaluation and certification for CC:2022

[COMP2] Composite product evaluation for Smart Cards and similar devices for CC3.1

[1] Available at http://data.europa.eu/eli/reg_impl/2024/482/oj

Note1: Unless otherwise specified, the latest version of referenced state-of-the-art documents applies.

Note2: both [COMP1] and [COMP2] will be referenced as **[COMP]** in this document and should be considered in accordance with the CC version they refer to.

1.2.2 Informative references

[CCRA] Arrangement on the Recognition of Common Criteria Certificates in the Field of Information Technology Security, Version 16.5.1 or higher, May 2014

[GP-AMD-N] GlobalPlatform Technology Card Specification "Amendment N Cryptographic Service Provider", Version 1.0 or higher, June 2026, GPC_SPE_230

[GP-GUI-CSP] GlobalPlatform Technology “CSP Guidance for Applet Developer”, Version 1.0 or higher, GPC_GUI_237

[GP-CardSpec] GlobalPlatform Technology "Card Specification", Version 2.3.1 or higher, March 2018, GPC_SPE_034

[GP-SE-PP] GlobalPlatform Technology "Secure Element Protection Profile", Version 1.0 or higher, February 2021, GPC_SPE_174

[GP-PP-CSP] GlobalPlatform Technology “PP-Module Amendment N: Cryptographic Service Provider (CSP)”, Version 1.0 or higher, GPC_SPE_238 (publication pending)

[JC-PP] Java Card System – Open Configuration Protection Profile, Version 3.1 or higher, April 2020, BSI-PP-0099

1.3 ACRONYMS AND DEFINITIONS

1.3.1 Acronyms

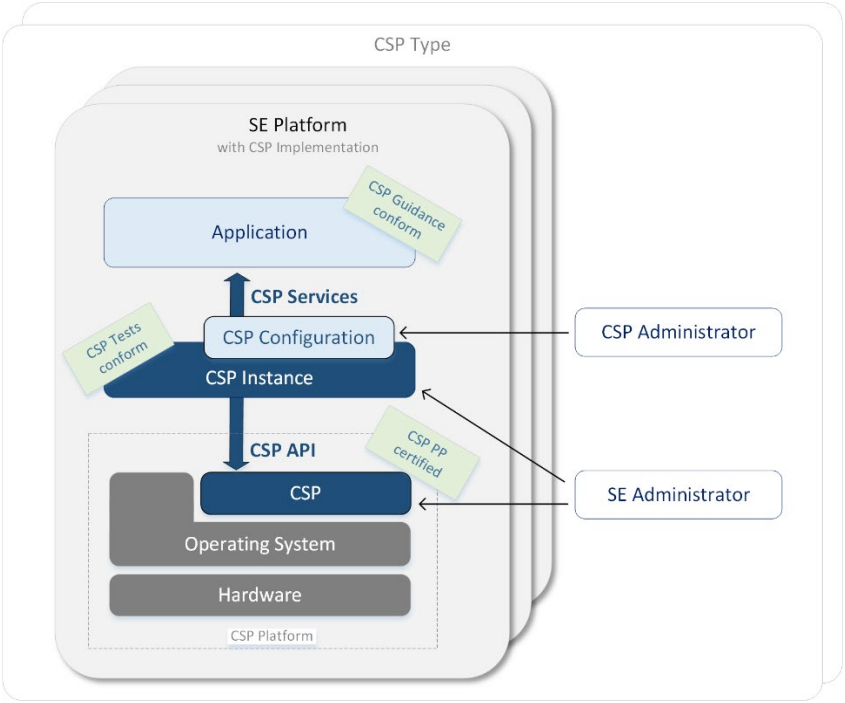
AID	Application Identifier
ACO	Assurance class for Composition
ACR	Access Control Rules
ASP	Application Service Providers
API	Application Programming Interface
AVA	Assessment of Vulnerabilities
CC	Common Criteria
CSP	Cryptographic Service Provider
DEK	Data Encryption Key
eSE	Embedded Secure Element
eUICC	embedded Universal Integrated Circuit Card
EAC	Extended Access Control
EAL	Evaluation Assurance Level (in context of CC)
ECASD	eUICC Capability Access Security Domain
ECKA-DH	Elliptic Curve Key Agreement – Diffie Hellman
ECKA-EG	Elliptic Curve Key Agreement – El Gamal
ETR	Evaluation Technical Report
GP	GlobalPlatform
HKDF	HMAC Key Derivation
ISD-P	Issuer Security Domain – Primary
ISD-R	Issuer Security Domain – Root
KDF	Key Derivation Function
MNO	Mobile Network Operator
MRTD	Machine Readable Travel Documents
MULTOS	Multi-application Operating System
OCE	Off-Card Entity
PACE	Password Authenticated Connection Establishment
PACE-CAM	PACE Connection with Chip Authentication Mapping
PP	Protection Profile
SAM	Secured Applications for Mobile

SAR	Security Assurance Requirements
SCP03	Secure Messaging Protocol 03
SCP11	Secure Channel Protocol 1.1
SD	Security Domain
SE	Secure Element
SFR	Security Function Requirement
ST	Security Target
TA	Terminal Authentication
TOE	Target of Evaluation
TSF	TOE Security Functionality
3S in SoC	Secure Subsystems in System on Chip

1.3.2 Definitions

The definitions used under Article 2 of Regulation (EU) 2019/881 (Cybersecurity Act) and under Article 2 of the Implementing Regulation (EU) 2024/482 apply to this document. The following definitions illustrated in Figure 1 also apply to this document.

Figure 1: Terms



Application
A software component that utilises the CSP services for its security functionalities.

Application developer (role)
An entity developing the application that utilises a CSP and provides the technical documentation required for the evaluation. It may also act as the sponsor of the application evaluation. Corresponds to developer in **[CC2022-1R1]**.

CSP
A component that acts as cryptographic abstraction layer and provides CSP services.

CSP type
A category of CSPs defined by a common CSP PP and CSP API version, along with corresponding CSP guidance and CSP tests.

CSP implementation

A vendor-specific realisation of a CSP Type. It is a subcomponent of a CSP Platform.

CSP service
A group of CSP API operations that provide specific security functionalities (e.g., cipher service, attestation service, logging service), usable only with a valid CSP configuration.

CSP platform
An SE platform equipped with a CSP that provides CSP services for applications.

CSP PP
The Protection Profile that defines the certification requirements for CSP Platforms of a specific CSP type.

CSP API
The interface definition implemented by all CSPs supporting the same CSP type.

CSP guidance
A universal security guidance document defining rules for applications to securely operate on CSP platforms of a given CSP type.

CSP tests
A compliance specification verifying that a CSP platform conforms to a particular CSP API.

CSP instance
A specific instantiation of the CSP API, providing CSP services that are pre-configured for a particular application use case.

CSP configuration
The definition of keys, certificates, passwords, algorithms, access right and policies used to configure the CSP services of a CSP instance for a particular application use case. The CSP configuration is part of the application TOE.

CSP administrator
A trusted entity responsible for configuring a CSP instance. This entity may be a human or a trusted application.

SE administrator
A trusted entity authorised to create CSP instances and grant CSP administrator access. This entity may be a human, a trusted application, or a signed script.

2. OVERVIEW

2.1 Problem description

The Secure Element (SE) landscape is evolving from smartcards to embedded Secure Elements (eSE) and Secure Subsystem in System on Chip (3S in SoC). Applications for eSE and 3S in SoC platforms are usually loaded on the platform after their delivery to the end user and are intended to run on a wide range of different platform configurations. Applying the smartcard composite evaluation approach to those applications necessitates multiple re-evaluations for each application **[COMP]**. This approach, requiring each application to be evaluated on every available platform, creates a geometrically scaling problem that is not practically feasible in the market of fast evolving mobile device platforms.

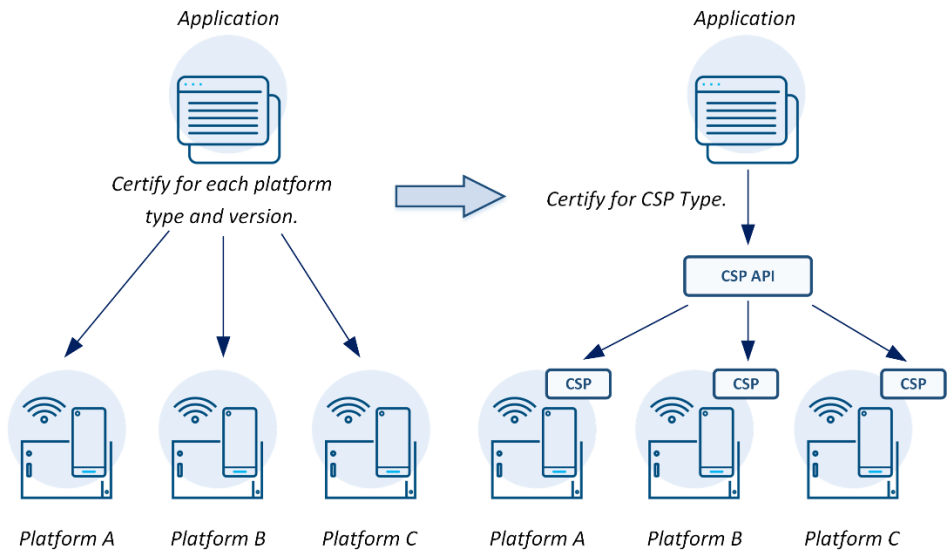
2.2 Objective

This document describes an evaluation approach for SE Applications using composition as an alternative to **[COMP]**. The approach aims to streamline the certification process by introducing the Cryptographic Service Provider (CSP) as a cryptographic abstraction layer. This layer defines a universal security guidance for application developers and provides certified, application-ready security services through a harmonised Application Programming Interface (CSP API) supporting typical application use cases.

The goal of such a CSP-layer is to omit the need for individually physical testing on each platform for applications that employ the CSP API for all their security-related needs.

Applications for Secure Elements and classical chip cards may benefit from such a CSP-layer, supporting portability across platforms without requiring recertification, as illustrated in Figure 2, “Certify once, deploy across platforms”.

Figure 2: Certify once, deploy across platforms



2.3 Concept of the methodology

The Target of Evaluation (TOE) is an application that relies on security services provided by a CSP. These CSP services cover application-specific security functionalities that are logically separated, self-contained and decoupled from the application. Furthermore, they are configured exclusively for the respective application – for example, the services provide cryptographic operations using TOE-specific key material.

As a result, all security functions invoked by the application via the CSP API are certified at EAL 4 + AVA_VAN.5 by the CSP certification. The application inherits de facto the security assurance from the CSP certification for the security functions used via on the CSP API. The TOE may still make use of functionality provided by the underlying platform, such as file management. However, if the TOE directly invokes security-related functionality from a

component other than the CSP, such functionality falls out of the scope of this assurance inheritance and should be addressed through other means (e.g., by excluding the affected functionality through the TOE guidance).

The CSP itself may use the security features of the underlying platform and expose them via the CSP API. Such features are considered in the CSP's own evaluation. Therefore, the CSP is evaluated using composite evaluation according to **[COMP]** including the operating system and hardware. The application, on the other hand, is evaluated following the CSP-based evaluation methodology defined in this document. This methodology combines EAL-based assurance assessment with ACO for the composition of the application and CSP, resulting in a unified evaluation process.

Note: Composite certification, as specified in **[COMP]**, and ACO for composition, as specified in Section 15 of **[CC2022-3R1]**, represent distinct evaluation approaches under Common Criteria (CC).

The required evaluation levels (including all direct and indirect dependencies) are as follows:

The TOE is evaluated at EAL2 augmented with ADV_IMP.1. In addition, it is evaluated as a composition with the corresponding CSP type at ACO_CAP-B.

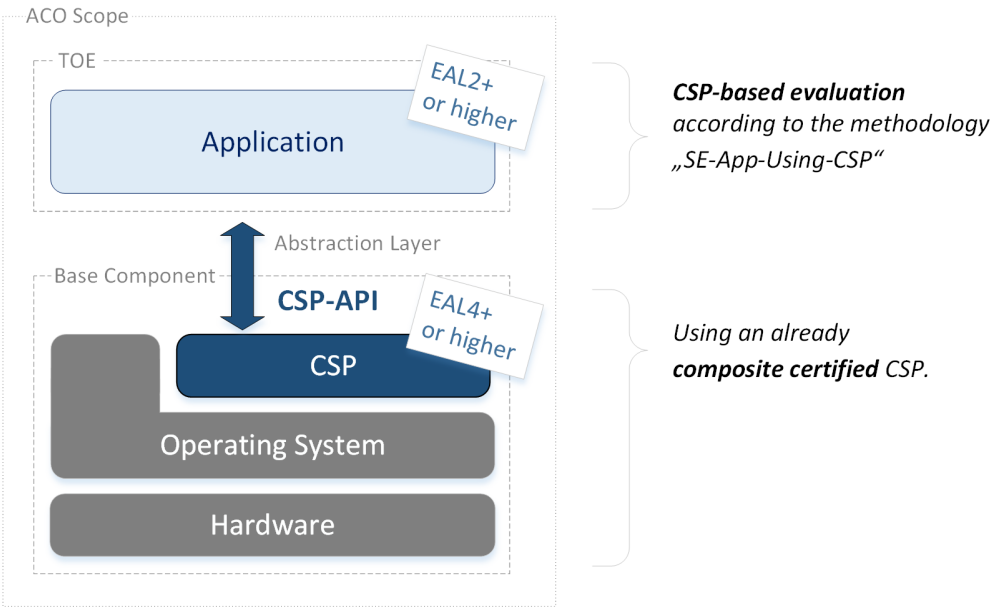
The CSP is certified at EAL4 augmented with AVA_VAN.5.

For both, augmentations and higher assurance levels or packages are permitted.

EAL2 augmented with ADV_IMP.1, with dependencies ADV_TDS.3, ADV_FSP.4 and ALC_TAT.1, defines the minimum assurance level required for the SAR evaluation of the application. CAP-B defines the minimum assurance package required for its composition evaluation with the CSP. If needed, additional augmentations (e.g., ALC_FLR.2 for vulnerability handling or ALC_DVS.1 for development security) may be included. Higher assurance levels (e.g., EAL3 or EAL4) or a higher assurance package for the composition (CAP-C) are also applicable, while the benefit of eliminating the need for individual physical testing remains unchanged.

This approach, illustrated in Figure 3, allows considering the results of physical TOE tests executed on one specific CSP-enabled platform as representative for all other platforms supporting the same CSP.

Figure 3: CSP-based evaluation



2.4 Scope

The CSP-based evaluation methodology described in this document is not restricted to a specific application scenario and may be applied to various use cases, such as digital identity (eID), payment systems, ticketing, access control, healthcare, automotive solutions, and others.

However, this methodology applies only to applications running directly on a CSP-enabled SE platform (CSP platform), meaning the application runs on the same SE platform as the CSP Implementation. Figure 4 illustrates this limitation.

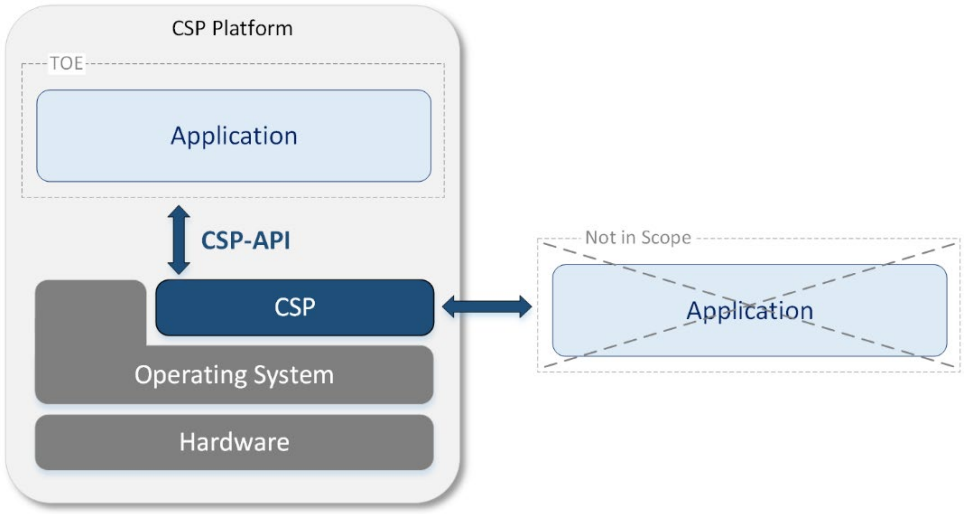
Typical examples of applications supported by this methodology include:

- Java Card applets operated on GlobalPlatform-compatible SEs with CSP support.
- Native applications operated on proprietary SEs with CSP support.

Note: The CSP approach for inheriting security assurance from a cryptographic abstraction layer can also be applied to other system architectures, such as client-server models. A future version of this document may extend support to applications running on separate platforms or to CSP Implementations hosted on non-SE platforms.

Editor's note: This document is written in a general style without a specific focus on Secure Elements (SE). However, we initially limited the scope to SE Platforms to keep the methodology simple as a starting point. Once we gain more confidence in the approach, we could consider expanding the scope by updating Section 1.4 and adjusting the document title.

Figure 4: Scope of the methodology



2.5 CSP types

Distinct system architectures, such as Java Card-based, native or MULTOS-based SE platforms, may require different types of CSPs. In such cases, the certification result of the Application is valid only for platforms that support the same CSP type.

A CSP type is characterized by the combination of a CSP PP and a CSP API specification:

CSP PP + CSP API = CSP Type

Thereby, the CSP API defines the harmonised interface across supported platforms, while the CSP PP specifies the hardware and operating system scope for which the CSP can be certified.

In addition to the CSP PP and CSP API, an applicable CSP guidance document and CSP test specification are required for the evaluation and use of a CSP type under this methodology. The applicable versions of these documents are therefore identified as part of the evaluation.

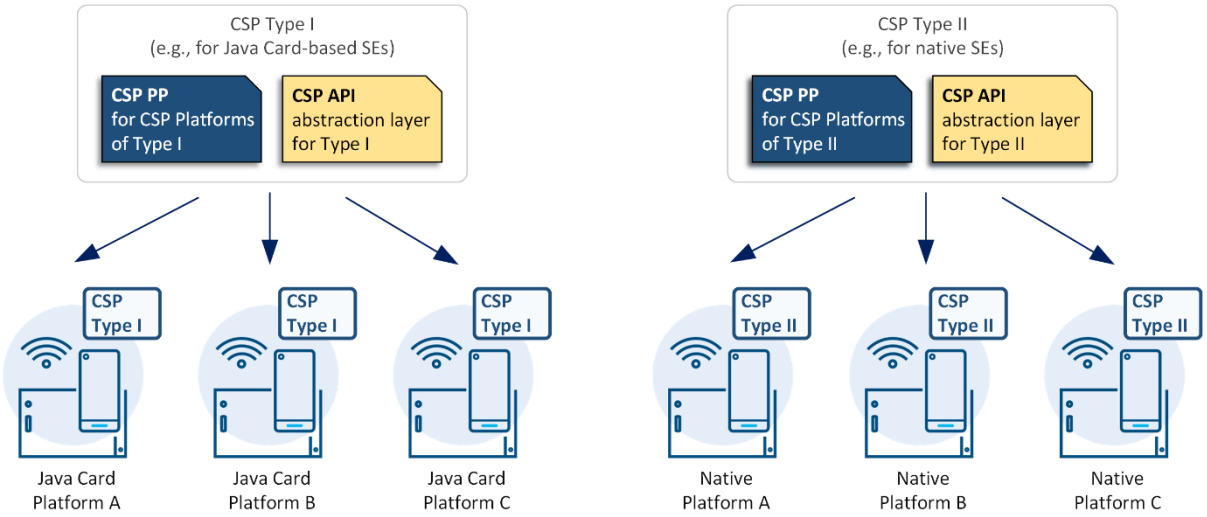
In summary, the following documents are required for the evaluation and use of a CSP type under this methodology:

- **CSP PP**: A Protection Profile that defines the security requirements for CSPs.
- **CSP API:** A standardised interface specification that serves as a security abstraction layer, offering certified security functionalities for applications.

- **CSP guidance**: A universal security guidance document that applies to all CSP products of a specific CSP type. It defines rules for applications to securely operate on all platforms implementing that CSP type.
- **CSP tests**: A CSP compliance test specification to verify whether a CSP-enabled platform is conformant to the CSP API.

In case a CSP API defines optional features or operations, these documents define how to ensure that an application can be operated only on CSP platforms providing all the features required by the application. This may, for example, be achieved through compatibility checks enforced by CSP guidance rules. However, this and other requirements for CSP types are specified in *Annex 1: Requirements for CSPs*. Only CSP types fulfilling these requirements are applicable for this methodology.

Figure 5: Different SE architectures may require different CSP types



In addition, CSP types may offer different security functionalities, addressing different technologies and/or scenarios. However, all CSP types ensure that the security services they provide meet the following criteria:

- **Logically separated**: The security services provided by the CSP operate independently from the internal logic of the application. This ensures that the CSP's security mechanisms remain unaffected by potential vulnerabilities or errors in the application, clearly separating the CSP's responsibility for security enforcement from the application's operational logic.
- **Self-contained**: The CSP implements its security functionalities as standalone components, independent of the application's state or behaviour, provided that the application adheres to the CSP guidance. It maintains cryptographic keys, security-critical data and its own operational state to ensure integrity and confidentiality under all operational conditions.
- **Decoupled**: Interaction between the application and the CSP is restricted to well-defined interfaces (i.e., the CSP API). This ensures that application changes or updates do not compromise the functionality or security of the CSP. Moreover, it simplifies the evaluation of the application by clearly separating the CSP's security services from the application's logic.

To provide logically separated, self-contained and decoupled CSP services, the CSP is pre-configured with a defined set of cryptographic keys, certificates, corresponding algorithms, etc.— collectively referred to as the CSP configuration. Authorised applications can invoke these pre-configured security services (i.e., the CSP services) without introducing new security risks or requiring custom security functionalities.

For further details on CSP types and corresponding sample architectures, see the annexes, which provide additional information related to Cryptographic Service Providers (CSPs). Annex 1 specifies the requirements for CSPs. All further annexes are informative, as they illustrate typical CSP specifications, services, and configuration options and serve as examples of how CSPs can be used.

The following overview summarises the content of each appendix:

- *Annex 1: Requirements for CSPs*

Lists requirements for CSP PP, CSP API, CSP guidance, CSP tests and CSP platforms that establish the conditions under which an application evaluation according to this methodology would be considered valid.

- *Annex 2: Sample CSP Services*

Lists cryptographic and security-related functionality typically provided by the CSP, such as cipher and signature operations, secure messaging, and PIN management.

- *Annex 3: Sample CSP Configuration*

Provides an example of how to configure a CSP to offer cryptographic operations that already include TOE-specific key material.

- *Annex 4: Sample CSP Multi Applications Support*

Describes how multiple applications with different or conflicting use cases can be supported on the same Secure Element and how isolation is ensured; for example, two eID Applications from different countries using the same CSP platform.

- *Annex 5: Sample CSP User-Level Authorisation*

Explains different concepts for identification, authentication and authorisation of both Applications and real-world users interacting with them.

- *Annex 6: Sample CSP Platform Dependencies*

Details how CSP implementations can expose existing security functionality of the underlying platform without re-implementing cryptography, instead encapsulating it as a cryptographic abstraction layer in line with this methodology.

- *Annex 7: Sample CSP within eUICC*

Provides an overview of a cryptographic abstraction layer in embedded Universal Integrated Circuit Cards (eUICCs), including possible CSP support for Mobile Network Operators (MNOs) with their respective telecom applications and for applications managed within a Secured Applications for Mobile (SAM) scope.

Note: The CSP guidance eliminates the need for platform-specific security instructions by defining CSP type-specific rules for security concerns that cannot be fully enforced by the CSP due to technical limitations or practical constraints. Applications need to adhere to these rules. For example, the guidance may specify CSP API methods where the use of sensitive arrays for input and output parameters is required to ensure data integrity.

2.6 Security assurance and composition

This CSP-based evaluation methodology, as introduced in Section 2.3, combines SAR-based assurance assessment with ACO Composition evaluation into a unified evaluation process with the following characteristics:

- **Combined ST**: SAR and ACO evaluation is performed using a single Security Target (ST).
- **Unified TOE definition**: The TOE is the application using a CSP type, treated as a single entity for both SAR and ACO evaluation.
- **Flexible evaluation levels (EAL2 augmented to EAL4 augmented)**: Certification is available at EAL2, EAL3 or EAL4, each augmented with at least ADV_IMP.1 including dependencies.
- **Relying on CSP functionality**: Security aspects covered by the CSP's certification do not require separate Security Functional Requirements (SFRs) or revalidation.
- **Custom security functionality**: Application SFRs are kept at a minimum. If such SFRs, or any of their dependencies, require platform security mechanisms, they rely exclusively on CSP API operations.

- **Composition evaluation**: The application's integration with the CSP is evaluated against ACO (ACO_COR.1, ACO_CTT.2, ACO_DEV.2, ACO_REL.1, ACO_VUL.2), using one representative CSP platform as the base component, in alignment with the SAR evaluation.
- **Unified testing and documentation**: A single evaluation testing process covers both TOE SFRs and CSP integration, using one representative CSP platform for physical testing.

Note: The representative CSP platform is selected by the application developer with no specific selection criteria, as the CSP provides a uniform abstraction layer with the same API and security services, ensuring identical functional behaviour across different CSP implementations.

Chapter 5 defines SAR , specifying the Application's security evaluation tasks under this methodology. Key aspects include:

- **ASE (Security Target Evaluation)**: The Security Target (ST) specifies the CSP type and versions the application is compatible with, and list the high-level CSP services used.
- **ADV (Development)**: The application's implementation representation demonstrates correct use of the CSP API and adherence to the CSP guidance. The TOE design explains the CSP configuration and the security-critical data flows between application and the CSP.
- **ALC (Life-cycle)**: The CSP configuration, as part of the TOE, is uniquely identified and subject to controlled modification.
- **AGD (Guidance)**: Deployment and initialisation of the TOE is fully specified, including the personalisation of the application and its corresponding CSP configuration.
- **ATE (Tests)**: Functional tests are performed on at least one representative CSP platform. The results are applicable to any other CSP platform with the same configuration.
- **AVA (Vulnerability Assessment)**: Physical vulnerability tests are conducted on one representative CSP platform, following the same approach as functional tests.

Chapter 6 defines ACO specific requirements, specifying composition evaluation of the application with a CSP platform as base component. Key aspects include:

- **ACO (Assurance Composition)**: The application integration is evaluated against harmonised specification documents of the CSP type, including CSP PP, CSP API, CSP guidance and CSP tests, and re-uses test results from ATE and VAN evaluation.

Note: The CSP tests are not re-evaluated during the application evaluation. However, the evaluator checks that they exist and that they cover the CSP functions required by the application (see 6.2).

2.7 Product or platform change

Application certification remains valid on any CSP platform that conforms to the same CSP type, including the CSP PP, CSP API, CSP guidance and CSP tests versions considered during evaluation. This means that a certified application can be transferred to another CSP platform of the same CSP type and versions without requiring re-evaluation. However, this applies only to CSP types that fulfil the requirements defined in *Annex 1: Requirements for CSPs*.

In case of changes to the Application, the standard maintenance, re-evaluation and re-assessment processes according to Annex IV of **[EUCC]** apply.

For CSP Platform changes, re-certification of the application is not required as long as:

- The new CSP platform remains compatible with the CSP type and the applicable versions of its CSP PP, CSP API, CSP guidance and CSP tests considered during evaluation.
- The certification status of the CSP platform remains valid.

If a CSP platform undergoes changes that invalidate its conformance to the CSP type, the application may no longer be operable without re-evaluation.

2.8 Certificate And Certification Report

The Certificate and Certification Report follow the standard EUCC structure and content, providing the formal outcome of the SAR evaluation of the application, combined with the ACO evaluation of its composition with the CSP in a single Certificate and a unified Certification Report.

The Certificate complies with the requirements of **[EUCC]**and additionally includes:

- **Product Name**: <<Application Name>> using <<CSP Type Name>>
- **Composition Statement:** Relies on CSP security functionality as defined by <<CSP PP>>
- **CSP API**: <<CSP API Name + Version>>
- **Evaluation Assurance Level**: <<TOE EAL>> with composition based on <<CSP PP EAL>>
- **Composition Assurance**: <<Composition Assurance Package + Components>>
- **TOE Evaluation Platform**: <<CSP Platform Name + Version>>

The Certification Report also follows **[EUCC]** and further extends it by documenting:

- **Tested CSP platform**: Specification of the exact CSP platform used for testing activities during the evaluation, including details such as platform name, platform version, operating system version, CSP Implementation version and tested configurations.
- **Information that the application can be transferred to compatible CSP platforms**.

For a complete list of Certificate and Certification Report contents, see Chapter 8.

2.9 Benefits of this methodology

This methodology defines a scalable certification approach by certifying applications against a CSP Type rather than a specific CSP implementation. As a result, an application evaluated once under this methodology keeps its certification status when used with any CSP Implementation that conforms to the CSP PP, CSP API, CSP guidance and CSP test specifications.

Compared to traditional composite evaluations according to **[COMP]**, this approach introduces the following structural differences:

- **Certify Once, Deploy Across Platforms** → Applications do not require re-evaluation for each CSP Implementation if the CSP conforms to the defined CSP type.
- **Decoupling of Application and CSP Certification** → Applications and CSP implementations are evaluated separately, reducing dependencies in the certification process.
- **Optimised Certification Effort** → Avoids redundant Common Criteria evaluations for every platform, improving efficiency.
- **Adaptability for CSP Implementations** → CSP vendors can develop new implementations that conform to the CSP type without requiring re-evaluation of certified applications.

This methodology establishes a modular evaluation model, enabling certification of applications relying on CSPs while maintaining compatibility across multiple CSP implementations.

3. EVALUATION ACTIVITIES

This chapter defines the evaluation activities required for the evaluation of an application using a CSP. The evaluation has to ensure, among others, that the application:

- Does not process highly critical assets (e.g., secret keys) on its own.
- Implements the standard security measures as described in the generic CSP guidance.
- Matches the CSP type of the base component.

3.1 STEP 1: PREPARATION PHASE

In this step, the evaluator ensures that the necessary input for the evaluation is available.

The evaluator checks that all EVALUATION INPUT MATERIAL defined in Chapter 4 is available for evaluation.

The evaluator checks that the application ST specifies the versions of all referenced EVALUATION INPUT MATERIAL.

3.2 STEP 2: TEST SETUP

In this step, the evaluator ensures that the test setup is suitable for evaluation activities.

The evaluator checks that the TOE can be deployed and operated using the testing deliverables listed in Section 4.3, ensuring that the test environment is correctly set up.

If deployment issues arise, evaluation activities not requiring direct TOE execution may proceed.

However, ATE and AVA testing will be postponed until the deployment issues are resolved.

3.3 STEP 3: SECURITY ASSURANCE EVALUATION (SAR)

In this step, the evaluator ensures that the evaluation includes the minimum required assurance components and refinements.

The evaluator performs the SAR evaluation considering the SAR REFINEMENTS specified in Chapter 5 and the EAL specified in the ST of the Application.

Supported evaluation levels are EAL2+, EAL3+ and EAL4+, requiring at minimum the augmentations ADV_IMP.1, ADV_TDS.3, ADV_FSP.4 and ALC_TAT.1.

If the EAL in the ST does not meet these requirements, the evaluation cannot proceed.

3.4 STEP 4: COMPOSITION EVALUATION (ACO)

In this step, the evaluator ensures that the evaluation includes the minimum set of ACO components including refinements.

The evaluator performs the ACO evaluation considering the ACO Refinements specified in Chapter 6 and the CAP level specified in the ST of the Application.

Supported composition assurance packages are CAP-B and CAP-C. At minimum, CAP-B is required by this methodology.

If the ST specifies a CAP level lower than CAP-B, the evaluation cannot proceed.

Where the same assurance components are required by both the chosen EAL (including augmentations) and the selected CAP package, the same documentation and results are used. No duplication is required, and higher component levels from the EAL also satisfy the corresponding CAP requirement.

3.5 STEP 5: FINAL ASSESSMENT AND CERTIFICATION

The certification body reviews test results and creates the Certificate And Certification Report according to Section 2.8.

4. EVALUATION INPUT MATERIAL

An evaluation according to this methodology requires the specification, documentation and testing deliverables outlined in this chapter.

Note: The completeness of these materials is verified in Step 1: Preparation Phase, see Section 3.1.

4.1 REQUIRED SPECIFICATIONS

The application developer identifies and provides the specification documents listed in Table 1.

Note: The versions of the specifications are specified in the Application ST under ASE_SPD – Security problem definition, see Section 5.1.1.

Table 1: Specifications relevant for TOE evaluation

Documentation	Description	Elements
CSP API	The harmonised interface documentation defining the CSP API.	ADV_IMP ACO_COR ACO_REL
CSP guidance	The security guidelines required for correct use of the CSP API.	ADV_IMP ADV_TDS ACO_REL
CSP tests	A compliance specification containing test cases to verify that a CSP platform conforms to the CSP API.	ACO_REL
CSP PP	The Protection Profile (PP) that defines the security requirements for CSP platforms supported.	ACO_REL
Other specifications	Specifications referenced by the application or CSP type to fulfil security objectives, including corresponding Protection Profiles.	ACO_REL

4.2 REQUIRED TOE DELIVERABLES

The TOE deliverables listed in Table 2 are provided by the application developer and may be further detailed during the evaluation process.

Table 2: TOE deliverables

Deliverables	Description	Elements
Security Target	The Security Target (ST) of the Application, including: • A TOE Summary Specification (TSS) listing CSP services used by the TOE and explaining how they contribute to meeting the SFRs. • A mapping table linking CSP services to the corresponding security objectives.	ASE_TSS ACO_REL

Deliverables	Description	Elements
	• References the CSP type, CSP PP, CSP API, CSP guidance and CSP tests the TOE is compatible with.	
Implementation Representation	The implementation representation of the application, such as source code, bytecode or equivalent artifacts, that provide exact information when and how the TOE invokes security operations.	ADV_IMP
TOE Design Specification	A specification describing the internal design and architecture of the TOE, including: • Integration and deployment on CSP platforms, ensuring that installation is restricted to platforms supporting the required CSP type, features, algorithms and CSP configuration. • The assets managed by the CSP, such as keys, certificates and passwords, along with their configuration properties. • CSP security assumptions, including expected behaviors and enforced security policies. • Flow of security-critical data between the TOE and the CSP. • Additional documentation required to evaluate adherence to the CSP guidance.	ADV_TDS ACO_COR ACO_REL
Further evaluation evidence	Further evaluation evidence required by the claimed SARs.	ADV_ARC ADV_FSP AGD_OPE AGD_PRE ACO_COR ACO_DEV ACO_VUL

4.3 REQUIRED TESTING DELIVERABLES

The application developer provides the testing deliverables listed in Table 3 to ensure the application evaluator can execute all required tests on a representative CSP Platform.

Table 3: Testing deliverables

Deliverables	Description	Elements
TOE Executables	The compiled and executable binaries of the TOE.	ATE_FUN ATE_COV AVA_VAN ACO_VUL
CSP Platform	A certified CSP-enabled platform that supports the CSP Type required by the TOE.	ATE_FUN AVA_VAN ACO_COR

Deliverables	Description	Elements
CSP Configuration Script	A machine-readable format defining the CSP Configuration parameters necessary for TOE testing.	ATE_FUN AVA_VAN ACO_CTT
TOE Deployment Script	Setup files used to install the TOE Executables on the CSP Platform.	ATE_FUN ACO_CTT
TOE Tests	Test cases and execution scripts required to verify TOE security functionalities and assess vulnerability resistance.	ATE_FUN AVA_VAN ACO_VUL

5. SAR REFINEMENTS

This Chapter extends SAR evaluation elements from **[CC2022-3R1]** using the refinement operation as specified in CC. The refinements are presented using the following notation:

- Text copied from [CC2022-3R1] appears as cited text.
- Text added or modified is in **bold**.
- Text removed is crossed ~~crossed-out~~.

All evaluation requirements (e.g., ASE_SPD.1.1C) not mentioned in this Chapter remain unchanged and apply as defined in **[CC2022-3R1]**.

The following summary highlights key aspects of these refinements. The complete refinements are specified in the corresponding subsections:

Key aspects of Section 5.1, ASE – Security Target evaluation:

- ASE_SPD: The ST assumes CSP platform compliance with CSP PP, CSP API, CSP guidance, CSP tests and their respective versions.
- ASE_TSS: The ST lists, for each SFR, the high-level CSP services and assets used.

Key aspects of Section 5.2, ADV – Development:

- ADV_IMP: Evaluators have access to source code or an equivalent representation to verify the correct use of the CSP API and that the TOE adheres to CSP guidance.
- ADV_TDS: The TOE design describes the CSP configuration, explains the security-critical data flows between the application and the CSP, and demonstrates that critical security assets are managed by the CSP and not by the application.

Key aspects of Section 5.3, ALC – Life-cycle support:

- ALC_CMS: The CSP configuration is included within the configuration management scope.
- ALC_CMC: The CSP configuration, as part of the TOE, is uniquely identified and subject to controlled modification.

Key aspects of Section 5.4, AGD – Guidance Documents:

- AGD_PRE: The preparative procedures describe the deployment and initialisation of the TOE, including the personalisation of the Application and its corresponding CSP configuration.

Key aspects of Section 5.5, ATE – Tests:

- ATE_FUN: Functional testing is performed at least on one representative CSP platform.
- ATE_IND: Independent testing is performed at least on one representative CSP platform.

Key aspects of Section 5.6, AVA – Vulnerability Assessment*:*

- AVA_VAN: Vulnerability testing is performed at least on one representative CSP platform.

Editor's note: Despite this document is a guidelines document and does not enforce any additional requirement to EUCC, the following ACO refinements are using the CC terminology and therefore "shall" statements.

5.1 ASE – SECURITY TARGET EVALUATION

These refinements ensure the Security Target (ST) correctly specifies CSP dependencies and usage.

5.1.1 ASE_SPD – Security problem definition

The Security problem definition (SPD) complies with ASE_SPD as defined in [CC2022-3R1], with the following refinement for CSP-based evaluation:

ASE_SPD.1.4C:

> The security problem definition shall describe the assumptions about the operational environment of the TOE, including an explicit assumption (A.Plat-CSP) that the CSP platform used by the TOE is assumed to conform to the required CSP type, including compliance with the specified versions of the CSP PP, CSP API, CSP guidance, and CSP tests, with their versions explicitly specified in the ST.

Rationale:

This refinement is required because correct operation of the TOE under this methodology requires a CSP platform that is consistent with the CSP PP, CSP API, CSP guidance, and CSP tests. Adherence of the TOE to the CSP API and CSP guidance, as verified in refinements to ADV, is important but not sufficient.

5.1.2 ASE_TSS – TOE summary specification

The TOE Summary Specification (TSS) complies with ASE_TSS as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ASE_TSS.1.1C:

> The TOE summary specification shall describe how the TOE meets each SFR and, for each SFR, shall list the high-level CSP services used to fulfil it, including the assets used by these services**.**

Rationale:

This information is necessary for the evaluator to obtain an overview that is required to verify that the application correctly uses the high-level CSP services, as needed for the evaluation of the refinements to ADV.

5.2 ADV – DEVELOPMENT

These refinements ensure that the application correctly integrates with the CSP type.

5.2.1 ADV_IMP – Implementation representation

The Implementation representation (IMP) complies with ADV_IMP as defined in **[CC2022-3R1]**, with the following refinements for CSP-based evaluation:

ADV_IMP.1.1C:

> The implementation representation shall define the TSF to a level of detail such that the TSF may be generated without further design decisions, and shall additionally
>
> - include exact information on when and how the application invokes API operations relevant for fulfilling SFRs (covering security-relevant API operations provided by the CSP, the operating system, third-party libraries, and hardware)**,**
> - for all SFRs that rely on CSP services, show that the application does not directly invoke security mechanisms provided by the operating system, third-party libraries, or hardware,
> - show that the SFR-enforcing functionality of the application does not re-implement security functions already provided by the CSP, and
> - show that the application adheres to the CSP guidance.

ADV_IMP.1.3C:

> The mapping between the TOE design description and the sample of the implementation representation shall demonstrate their correspondence, and for all SFRs that rely on CSP services, show that the CSP API

operations invoked by the application are consistent with the high-level CSP services and associated assets described in the TOE summary specification.

Rationale:

These refinements are required to ensure that the evaluator verifies the correct and secure use of CSP services by the application. In particular, they clarify that the application consistently invokes CSP API operations as described in the TSS, avoids bypassing or duplicating CSP security functionality, and adheres to the CSP guidance.

5.2.2 ADV_TDS – TOE design

The TOE design (TDS) complies with ADV_TDS as defined in **[CC2022-3R1]**, with the following refinements for CSP-based evaluation:

ADV_TDS.3.4C:

The design shall provide a description of each subsystem of the TSF, including a detailed description of the CSP configuration defining the CSP-managed assets (e.g., cryptographic keys, certificates, passwords) and their security-relevant parameters (e.g., algorithms, key sizes, security policies).

ADV_TDS.3.5C:

The design shall provide a description of the interactions among all subsystems of the TSF, including the flow of security-critical data between the application and the CSP**.**

ADV_TDS.3.7C:

The design shall describe each SFR-enforcing module in terms of its purpose and relationship with other modules and shall demonstrate that the application does not manage critical security assets, which are expected to be managed exclusively through the CSP configuration**.**

Rationale:

These refinements are required to ensure that the TOE design provides a structured and complete description of how the application uses the CSP, its interaction with the CSP, and the use-case-specific CSP configuration used to define and manage security-relevant assets.

5.3 ALC – LIFE-CYCLE SUPPORT

These refinements ensure proper configuration management of the application-specific CSP configuration throughout the TOE life cycle.

5.3.1 ALC_CMS – Configuration Management Scope

The Configuration Management Scope (CMS) complies with ALC_CMS as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ALC_CMS.2.1C:

The configuration list shall include the TOE itself; the evaluation evidence required by the SARs; and the parts that comprise the TOE, including the CSP configuration that is delivered with the TOE for the application use case.

Rationale:

This refinement ensures that the CSP configuration delivered with the TOE is included within the configuration management scope.

5.3.2 ALC_CMC – Configuration Management Capabilities

The Configuration Management Capabilities (CMC) complies with ALC_CMC as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ALC_CMC.2.2C:

The CM documentation shall describe the method used to control changes to the CSP configuration, including version control and the secure deactivation or deletion of configuration items.

Rationale:

This refinement is required to ensure that the CSP configuration is version controlled and subject to controlled modification throughout the TOE life cycle.

5.4 AGD – GUIDANCE DOCUMENTS

These refinements ensure correct deployment of the TOE.

5.4.1 AGD_PRE – Preparative procedures

AGD_PRE.1.1C:

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures, including the security-relevant assets required for TOE deployment and their role in the secure initialisation of the TOE**.**

AGD_PRE.1.2C:

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST, including the deployment of the application, and its security-related personalisation through the CSP configuration**.**

Rationale:

These refinements are required to ensure that the TOE, including the CSP configuration, is correctly and securely deployed to the CSP platform.

5.5 ATE – TESTS

These refinements ensure all testing is performed on a single CSP platform that serves as a representative implementation of the CSP type.

5.5.1 ATE_FUN – Functional Tests

The Functional Tests (FUN) complies with ATE_FUN as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ATE_FUN.2.1D:

The developer shall test the TSF and document the results on at least one representative CSP platform.

Rationale:

This refinement allows developer functional testing on one representative CSP platform, since (ASE_SPD.1.4C) all CSP platforms conform to the same CSP API and PP, (ADV_IMP.1.1E) the application follows the CSP guidance and implements no cryptography, and (ADV_TDS.3.2E) no highly critical assets exist at the application layer.

5.5.2 ATE_IND – Independent Testing

The Independent Testing (IND) complies with ATE_IND as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ATE_IND.1.2E:

The evaluator shall test a subset of the TSF on at least one representative CSP platform to confirm that the TSF operates as specified.

Rationale:

This refinement allows evaluator independent testing on one representative CSP platform, because the same rationale as for ATE_FUN.2.1D applies.

5.6 AVA – VULNERABILITY ASSESSMENT

Vulnerability testing is performed on a single CSP platform that serves as a representative implementation of the CSP type.

5.6.1 AVA_VAN – Vulnerability Analysis

The Vulnerability Analysis (VAN) complies with AVA_VAN as defined in **[CC2022-3R1]**.

No refinements are introduced for AVA_VAN in this methodology.

Rationale:

Vulnerability Analysis applies only to the application as the TOE, with testing performed on one representative CSP platform, for the same reasons given in the rationale for ATE_FUN.2.1D. Platform-specific vulnerabilities are covered by the independent evaluation of the CSP.

6. ACO REFINEMENTS

This Chapter extends ACO evaluation elements from **[CC2022-3R1]** using the refinement operation as specified in CC using the same notation as defined in Chapter 5, SAR .

All evaluation requirements (e.g., ACO_DEV.1.1D) not mentioned in this Chapter remain unchanged and apply as defined in **[CC2022-3R1]**.

The following summary highlights key aspects of the ACO refinements. The complete refinements are specified in the corresponding subsections:

Key aspects of Section 6, ACO Refinements*:*

- ACO_COR: Verifies the CSP PP referred in A.Plat-CSP fulfils Table 6: Requirements for CSP PP.
- ACO_DEV: Verifies CSP SFRs relied on by the application are covered by CSP tests, and that these CSP tests fulfil Table 9: Requirements for CSP tests.
- ACO_REL: Evaluates that the application performs runtime checks to ensure that the CSP platform used supports all features and the correct versions of the CSP API and CSP configuration required by the TOE, and that the CSP API and CSP guidance fulfil Table 7: Requirements for CSP API and Table 8: Requirements for CSP guidance.
- ACO_CTT: Allows reuse of ATE results from SAR evaluation to demonstrate correct behaviour of the composed TOE.
- ACO_VUL: Allows reuse of VAN results from SAR evaluation for the composed TOE.

Editor's note: as in previous Chapter, despite this document is a guidelines document and does not enforce any additional requirement to EUCC, the following ACO refinements are using the CC terminology and therefore "shall" statements.

6.1 ACO_COR COMPOSITION RATIONALE

The Correctness of Composition (COR) complies with ACO_COR as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ACO_COR.1.1C:

> The composition rationale shall demonstrate that a level of assurance at least as high as that of the dependent component has been obtained for the supported functionality of the base component, when the base component is configured as required to support the TSF of the dependent component, including **by referencing** that the Protection Profile **claimed by** the CSP Platform fulfils the requirements listed in Table 6: Requirements for CSP PP.

Rationale:

The purpose of this refinement is not to re-evaluate the CSP, but to verify that it is suitable for this methodology by verifying the CSP PP.

6.2 ACO_DEV DEVELOPMENT EVIDENCE

The Development evidence (DEV) complies with ACO_DEV as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ACO_DEV.1.2C:

> The development information shall show correspondence between the interfaces, used in the composed TOE, of the base component and the dependent component to support the TSF of the dependent component, and shall show that the security functions of the base component relied upon by the dependent component can be enforced through CSP tests that fulfil the requirements listed in Table 9: Requirements for CSP tests.

Rationale:

The purpose of this refinement is not to require execution of CSP tests, but to ensure that for each application SFR relying on CSP services the corresponding CSP SFRs have appropriate test coverage, and that such coverage is assumed to have been executed for any CSP platform used.

6.3 ACO_REL RELIANCE ON DEPENDENT COMPONENT

The Reliance of Dependent Component (REL) complies with ACO_REL as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ACO_REL.1.1C:

The reliance information shall describe the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF**, and shall show that**

- **the CSP API of the CSP platform fulfils the requirements listed in Table 7: Requirements for CSP API,**
- **the CSP guidance required by the CSP API and used by the application fulfils the requirements listed in** Table **8: Requirements for CSP guidance,**
- **the CSP API provides the necessary security functionalities required by the TOE**.

ACO_REL.1.2C:

The reliance information shall describe all interactions through which the dependent component TSF requests services from the base component, and shall show that

- **the application includes runtime checks to ensure that the CSP platform supports the required CSP API and version,**
- **the application includes runtime checks to ensure that the CSP platform supports the required cryptographic algorithms, key sizes, and security features needed by the application,**
- **the application includes runtime checks to ensure the CSP configuration is compatible with the one specified in the TOE design,**
- **the security functions claimed in the TOE's ST, which rely on CSP functionality, align with the security mechanisms available through the CSP API.**

Rationale:

The purpose of this refinement is to verify that the application relies only on a CSP API and CSP guidance accepted by this methodology, and that runtime checks ensure the required CSP API version, features, and configuration are available.

6.4 ACO_CTT COMPOSED TOE TESTING

The Composed TOE Testing (CTT) complies with ACO_CTT as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

ACO_CTT.2.1C:

The composed TOE and base component interface test documentation shall consist of test plans, expected test results and actual test results. **Developer functional test documentation required by ATE_FUN and ATE_IND may be reused, provided it fulfils the needs of the CTT requirements**.

ACO_CTT.2.2C:

The test documentation from the developer execution of the composed TOE tests shall demonstrate that the TSF behaves as specified and is complete**, including the effective operation of the runtime checks defined in ACO_REL.1.2C**.

Rationale:

The purpose of this refinement is to reuse results from ATE_FUN and ATE_IND to demonstrate correct integration of the application with the CSP, avoiding redundant evaluator testing in ACO_CTT.

6.5 ACO_VUL COMPOSITION VULNERABILITY ANALYSIS

The Composition Vulnerability Analysis (VUL) complies with ACO_VUL as defined in **[CC2022-3R1]**, with the following refinement for CSP-based evaluation:

Rationale:

Vulnerability analysis and testing are already covered by the refinements to AVA_VAN – Vulnerability Analysis (Section 5.6.1), which restrict testing to a representative CSP platform and relies on platform-specific vulnerability analysis covered by the independent evaluation of the CSP.

7. ADDITIONAL WORK UNITS

This chapter defines additional evaluator work units required to operationalise the SAR and ACO refinements defined in Chapters 5 and 6. All work units not explicitly mentioned in this chapter remain unchanged and apply as defined in **[CC2022-3R1]**.

Editor's note: as in previous Chapters, despite this document is a guidelines document and does not enforce any additional requirement to EUCC, the following additional work units are using the CC terminology and therefore "shall" statements.

7.1 SAR-RELATED WORK UNITS

7.1.1 Action ASE_SPD.1.1E

Section 5.1.1 ASE_SPD.1.4C: *The security problem definition shall describe the assumptions about the operational environment of the TOE, including an explicit assumption (A.Plat-CSP) that the CSP platform used by the TOE is assumed to conform to the required CSP type, including compliance with the specified versions of the CSP PP, CSP API, CSP guidance, and CSP tests, with their versions explicitly specified in the ST.*

7.1.1.1 Work Unit ASE_SPD.1-5 (A.Plat-CSP)

The evaluator shall check the Security Target to determine whether the security problem definition includes an explicit assumption (A.Plat-CSP) that the CSP platform used by the TOE conforms to the required CSP type.

7.1.1.2 Work Unit ASE_SPD.1-6 (CSP versions)

The evaluator shall check that the assumption A.Plat-CSP explicitly specifies the required versions of the CSP Protection Profile, CSP API, CSP guidance, and CSP tests.

7.1.1.3 Work Unit ASE_SPD.1-7 (Consistency)

The evaluator shall examine the Security Target to determine whether the versions of the CSP Protection Profile, CSP API, CSP guidance, and CSP tests specified in the assumption A.Plat-CSP are presented consistently and without ambiguity.

In particular, the evaluator verifies that:

- References to CSP artefacts are sufficiently precise to uniquely identify the expected CSP type.
- Exactly one unambiguous version is specified for each CSP artefact throughout the Security Target.
- Version ranges, wildcard expressions, or open-ended version statements (e.g., "or later", "compatible with") are not used.
- The specified versions of the different CSP artefacts are mutually compatible.

7.1.2 Action ASE_TSS.1.1E

Section 5.1.2 ASE_TSS.1.1C: *The TOE summary specification shall describe how the TOE meets each SFR and, for each SFR, shall list the high-level CSP services used to fulfil it, including the assets used by these services.*

7.1.2.1 Work Unit ASE_TSS.1-3 (SFR to CSP services)

The evaluator **shall check** that the TOE summary specification provides a mapping between each SFR that relies on CSP functionality and the high-level CSP services used to fulfil it.

7.1.2.2 Work Unit ASE_TSS.1-5 (Assets per SFR)

The evaluator shall **check that**, for each SFR that relies on CSP functionality, the TOE summary specification identifies the assets used by the referenced CSP services.

7.1.2.3 Work Unit ASE_TSS.1-6 (Completeness and accuracy)

The evaluator **shall examine** the TOE summary specification to determine whether all SFRs that rely on CSP functionality are covered by the mappings to CSP services and associated assets.

The evaluator determines that the TOE summary specification, for each SFR, provides a description of if, which, and how CSP functionality is used within the SFR. This description shall include which assets are used and how they are accessed through the CSP.

The evaluator determines whether the assumptions about the CSP, its expected behaviour, and the security policies enforced by the CSP are appropriate for the application-specific use case.

7.1.3 Action ADV_IMP.1.1E

Section 5.2.1 ADV_IMP.1.1C: *The implementation representation shall define the TSF to a level of detail such that the TSF may be generated without further design decisions, and shall additionally:*

- *include exact information on when and how the application invokes API operations relevant for fulfilling SFRs (covering security-relevant API operations provided by the CSP, the operating system, third-party libraries, and hardware),*
- *for all SFRs that rely on CSP services, show that the application does not directly invoke security mechanisms provided by the operating system, third-party libraries, or hardware,*
- *show that the SFR-enforcing functionality of the application does not re-implement security functions already provided by the CSP, and*
- *show that the application adheres to the CSP guidance.*

7.1.3.1 Work Unit ADV_IMP.1-4 (No bypass of CSP security mechanisms)

The evaluator **shall examine** the implementation representation to determine whether, for SFRs that rely on CSP services, the application does not directly invoke security mechanisms other than those provided by the CSP, such as security mechanisms provided by the operating system, third-party libraries, or hardware.

7.1.3.2 Work Unit ADV_IMP.1-5 (No re-implementation of CSP security)

The evaluator **shall examine** the implementation representation to determine whether the SFR-enforcing functionality of the application does not re-implement security functions already provided by the CSP.

7.1.3.3 Work Unit ADV_IMP.1-6 (Adherence to CSP guidance)

The evaluator **shall examine** the implementation representation to determine whether the application adheres to the CSP guidance.

Section 5.2.1 ADV_IMP.1.3C: *The mapping between the TOE design description and the sample of the implementation representation shall demonstrate their correspondence, and for all SFRs that rely on CSP services, show that the CSP API operations invoked by the application are consistent with the high-level CSP services and associated assets described in the TOE summary specification.*

7.1.3.4 Work Unit ADV_IMP.1-7 (CSP API usage consistency)

The evaluator **shall examine** the implementation representation to determine whether, for all SFRs that rely on CSP services, the CSP API operations invoked by the application are consistent with the high-level CSP services and associated assets described in the TOE summary specification.

7.1.4 Action ADV_TDS.3.1E

Section 5.2.2 ADV_TDS.3.4C: *The design shall provide a description of each subsystem of the TSF, including a detailed description of the CSP configuration defining the CSP-managed assets (e.g., cryptographic keys, certificates, passwords) and their security-relevant parameters (e.g., algorithms, key sizes, security policies).*

7.1.4.1 Work Unit ADV_TDS.3-19 (CSP configuration)

The evaluator shall examine the CSP configuration to determine whether the CSP-managed assets referenced in the TOE summary specification (see ASE_TSS.1.1C) are uniquely identified.

The evaluator determines that the cryptographic and security-relevant parameters, such as algorithms, key sizes, and security policies, of these assets are explicitly defined.

The evaluator further determines whether the specified parameters are consistent with the intended use of the CSP-managed assets as described in the TOE summary specification.

The evaluator is reminded that granting more privileges than necessary for the intended use of the assets may indicate inconsistencies with the defined security objectives.

Section 5.2.2 ADV_TDS.3.5C: *The design shall provide a description of the interactions among all subsystems of the TSF, including the flow of security-critical data between the application and the CSP.*

7.1.4.2 Work Unit ADV_TDS.3-20 (Security-critical data flows)

The evaluator shall examine the TOE design to determine whether the described interactions between the application and the CSP define security-critical data flows that are consistent with the security requirements of the assets involved.

The evaluator determines that the TOE design identifies the security-critical data exchanged between the application and the CSP and relates these data flows to the corresponding CSP-managed assets and their security properties.

The evaluator further determines whether the handling of security-critical data within the described data flows is appropriate with respect to the required security properties of these assets, such as confidentiality, integrity, authenticity, or availability. In particular, the evaluator examines whether the data flows are constrained to CSP services that provide the required protection properties for the respective assets.

The evaluator is reminded that the objective of this examination is to assess the consistency of the TOE design with the defined security requirements of the assets. The evaluator does not assess the correctness of the technical implementation or the effectiveness of the underlying protection mechanisms.

Section 5.2.2 ADV_TDS.3.7C: *The design shall describe each SFR-enforcing module in terms of its purpose and relationship with other modules and shall demonstrate that the application does not manage critical security assets, which are expected to be managed exclusively through the CSP configuration.*

7.1.4.3 Work Unit ADV_TDS.3-21 (Management of critical assets)

The evaluator **shall examine** the TOE design to determine whether critical security assets are not managed by the application and are instead managed through the CSP.

The evaluator determines that the TOE design shows that critical security assets are created, stored, processed and controlled by the CSP, and not by the application. The application may only initiate requests for security-relevant operations, which are executed by the CSP.

For the purpose of this examination, critical security assets are assets whose compromise would directly affect the enforcement of the SFRs (e.g., cryptographic keys, credentials, or security-relevant configuration parameters); the examples given are not intended to be exhaustive.

The evaluator is reminded that this examination focuses on the responsibilities of the application and the CSP and does not assess the implementation or effectiveness of CSP services.

7.1.5 Action ALC_CMS.2.1E

Section 5.3.1 ALC_CMS.2.1C: *The configuration list shall include the TOE itself; the evaluation evidence required by the SARs; and the parts that comprise the TOE, including the CSP configuration that is delivered with the TOE for the application use case.*

7.1.5.1 Work Unit ALC_CMS.2-4 (CSP configuration scope)

The evaluator **shall examine** the configuration list to determine whether the CSP configuration delivered with the TOE for the application-specific use case is included in the configuration list.

The evaluator determines that the CSP configuration is listed as a configuration item of the evaluated TOE.

7.1.6 Action ALC_CMC.2.1E

Section 5.3.2 ALC_CMC.2.2C: *The CM documentation shall describe the method used to control changes to the CSP configuration, including version control and the secure deactivation or deletion of configuration items.*

7.1.6.1 Work Unit ALC_CMC.2-6 (CSP configuration change control)

The evaluator **shall examine** the configuration management documentation to determine whether the method used to control changes to the CSP configuration are described, including version control and the secure deactivation or deletion of configuration items.

The evaluator determines that the configuration management documentation describes how changes to the CSP configuration are controlled.

The evaluator determines that version control is applied to the CSP configuration.

The evaluator determines that, where applicable, the configuration management documentation describes the secure deactivation or deletion of configuration items.

7.1.7 Action AGD_PRE.1.1E

Section 5.4.1 AGD_PRE.1.1C: *The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures, including the security-relevant assets required for TOE deployment and their role in the secure initialisation of the TOE.*

7.1.7.1 Work Unit AGD_PRE.1-4 (Deployment assets)

The evaluator **shall examine** the preparative procedures to determine whether all security-relevant assets required for secure acceptance of the delivered TOE, including CSP configuration and security-related personalisation assets, are specified.

The evaluator determines whether the preparative procedures describe the purpose of these assets in the secure acceptance of the TOE and clearly identify which assets are required prior to installation.

Section 5.4.1 AGD_PRE.1.2C: *The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST, including the deployment of the application, and its security-related personalisation through the CSP configuration.*

7.1.7.2 Work Unit AGD_PRE.1-5 (Secure initialisation)

The evaluator **shall examine** the preparative procedures to determine whether the steps for secure installation and initialisation of the TOE describe how the specified security-relevant assets are used to prepare the operational environment and to deploy the application and the CSP configuration.

The evaluator determines whether the described steps are ordered, complete, and consistent with the security objectives for the operational environment.

7.1.8 Action ATE_FUN.2.1E

Section 5.5.1 ATE_FUN.2.1D: The developer shall test the TSF and document the results on at least one representative CSP platform.

7.1.8.1 Work Unit ATE_FUN.2-9 (Representative CSP platform)

The evaluator shall check that the developer functional tests were performed on one representative CSP platform.

Testing on additional CSP platforms may be performed but is not required, as all CSP platforms of the same CSP type are assumed to conform to the same CSP API, CSP guidance, and CSP Protection Profile.

7.1.9 Action ATR_IND.1.2E

Section 5.5.2 ATE_IND.1.2E: The evaluator shall test a subset of the TSF on at least one representative CSP platform to confirm that the TSF operates as specified.

7.1.9.1 Work Unit ATE_IND.1-9 (Representative CSP platform)

The evaluator shall conduct independent testing on one representative CSP platform.

Independent testing on additional CSP platforms may be performed but is not required, as all CSP platforms of the same CSP type are assumed to provide equivalent CSP functionality.

7.2 ACO-RELATED WORK UNITS

7.2.1 Action ACO_COR.1.1E

Section 6.1 ACO_COR.1.1C: *The composition rationale shall demonstrate that a level of assurance at least as high as that of the dependent component has been obtained for the support functionality of the base component, when the*

base component is configured as required to support the TSF of the dependent component, including by referencing that the Protection Profile claimed by the CSP platform fulfils the requirements listed in Table 6: Requirements for CSP PP.

7.2.1.1 Work Unit ACO_COR.1-4 (CSP PP claim)

The evaluator **shall examine** the evidence provided to determine whether the CSP platform claims compliance with a clearly identified Protection Profile covering the requirements listed in *Table 6* that are relevant to the CSP functionality relied upon by the TOE to enforce its SFRs.

The evaluator determines whether the claim explicitly states that the relevant requirements are fulfilled, without exclusions or limitations.

7.2.2 Action ACO_DEV.2.2E

Section 6.2 ACO_DEV.1.2C: *The development information shall show correspondence between the interfaces, used in the composed TOE, of the base component and the dependent component to support the TSF of the dependent component, and shall show that the security functions of the base component relied upon by the dependent component can be enforced through CSP tests that fulfil the requirements listed in Table 9: Requirements for CSP tests.*

7.2.2.1 Work Unit ACO_DEV.2-6 (CSP test coverage)

The evaluator shall examine the development evidence to determine whether the security functions of the CSP relied upon by the TOE to enforce its SFRs are covered by CSP tests that fulfil the requirements listed in *Table 9*.

The evaluator determines whether the provided evidence demonstrates that the relevant CSP Tests are applicable to the claimed CSP platform and cover the CSP functionality on which the TOE depends.

7.2.3 Action ACO_REL.1.1E

Section 6.3 ACO_REL.1.1C: *The reliance information shall describe the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF, and shall show that*

- *the CSP API of the CSP platform fulfils the requirements listed in Table 7: Requirements for CSP API,*
- *the CSP guidance required by the CSP API and used by the application fulfils the requirements listed in* Table 8: Requirements for CSP guidance*,*
- *the CSP API provides the necessary security functionalities required by the TOE.*

7.2.3.1 Work Unit ACO_REL.1-5 (CSP API claim)

The evaluator **shall examine** the reliance information to determine whether it identifies the CSP API methods relied upon by the TOE to enforce its SFRs and shows that these API methods fulfil the requirements listed in *Table 7*.

7.2.3.2 Work Unit ACO_REL.1-6 (CSP guidance claim)

The evaluator **shall examine** the reliance information to determine whether the CSP guidance required by the CSP API and used by the application is identified and shown to fulfil the requirements listed in *Table 8*.

The evaluator determines whether the reliance information explains how the referenced CSP guidance supports the secure use of the CSP API by the application for the relied-upon functionality.

7.2.3.3 Work Unit ACO_REL.1-7 (Required security functionality)

The evaluator **shall examine** the reliance information to determine whether it shows that the CSP API provides the security functionalities required by the TOE to enforce its SFRs.

The evaluator determines whether the reliance information establishes a clear relationship between the TOE SFRs and the security functionalities provided by the CSP API on which the TOE depends.

Section 6.3 ACO_REL.1.1C: *The reliance information shall describe all interactions through which the dependent component TSF requests services from the base component, and shall show that*

- *the Application includes runtime checks to ensure that the CSP platform supports the required CSP API and version,*

- *the application includes runtime checks to ensure that the CSP platform supports the required cryptographic algorithms, key sizes, and security features needed by the application,*
- *the application includes runtime checks to ensure the CSP configuration is compatible with the one specified in the TOE design,*
- *the security functions claimed in the TOE's ST, which rely on CSP functionality, align with the security mechanisms available through the CSP API.*

7.2.3.4 Work Unit ACO_REL.1-8 (CSP API runtime check)

The evaluator **shall examine** the reliance information to determine whether it describes that the application includes runtime checks to verify that the CSP platform supports the required CSP API and API version relied upon by the TOE.

7.2.3.5 Work Unit ACO_REL.1-9 (Feature runtime check)

The evaluator **shall examine** the reliance information to determine whether it describes that the application includes runtime checks to verify that the CSP platform supports the cryptographic algorithms, key sizes, and security features required by the application.

7.2.3.6 Work Unit ACO_REL.1-10 (CSP configuration runtime check)

The evaluator **shall examine** the reliance information to determine whether it describes that the application includes runtime checks to verify that the CSP configuration in use is compatible with the CSP configuration specified in the TOE design.

The evaluator determines whether the reliance information explains how configuration mismatches are detected and handled to prevent reliance on an invalid or outdated CSP configuration.

7.2.3.7 Work Unit ACO_REL.1-11 (CSP TOE SFR alignment)

The evaluator **shall examine** the reliance information to determine whether it shows that the security functions claimed in the TOE's Security Target that rely on CSP functionality are aligned with the security mechanisms provided through the CSP API.

The evaluator determines whether the reliance information establishes that the CSP mechanisms available at runtime are sufficient to support the enforcement of the TOE SFRs that depend on CSP functionality.

7.2.4 Action ACO_CTT.2.1E

Section 6.4 ACO_CTT.2.1C: *The composed TOE and base component interface test documentation shall consist of test plans, expected test results and actual test results. Developer functional test documentation required by ATE_FUN and ATE_IND may be reused, provided it fulfils the needs of the CTT requirements*.

7.2.4.1 Work Unit ACO_CTT.2-12 (Reuse of ATE test documentation)

The evaluator **shall examine** any developer functional test documentation reused from ATE_FUN or ATE_IND to determine whether it fulfils the needs of the ACO_CTT requirements for testing the interfaces between the composed TOE and the base component.

Section 6.4 ACO_CTT.2.2C: *The test documentation from the developer execution of the composed TOE tests shall demonstrate that the TSF behaves as specified and is complete, including the effective operation of the runtime checks defined in ACO_REL.1.2C.*

7.2.4.2 Work Unit ACO_CTT.2-13 (Runtime checks)

The evaluator **shall examine** the test documentation from the developer execution of the composed TOE tests to determine whether it demonstrates the effective operation of the runtime checks defined in ACO_REL.1.2C.

8. COMBINED CERTIFICATE AND CERTIFICATION REPORT

8.1 COMBINED CERTIFICATE CONTENT

Table 4 specifies the required fields of the Certificate presenting the result of the SAR and ACO evaluation.

Table 4: Content of the Combined SAR and ACO Certificate

Field	Source	Content
Certification/Validation Identification	[EUCC]	Unique identifier of the certificate
Product Name	2.8	"<<Application Name>> using <<CSP Type Name>>"
Composition Statement	2.8	"Relies on CSP security functionality as defined by <<CSP PP>>"
Type of Product		Type of Product
Application Version	[EUCC]	Application Version and Release Numbers
CSP API Version	2.8	"<<CSP API Name + Version>>"
Evaluation Assurance Level	2.8	"<<TOE EAL + Augmentations>> with composition based on <<CSP PP EAL + Augmentations>>"
Composition Assurance	2.8	"<<Composition Assurance Package>> (<<Composition Assurance Components>>)"
Evaluation Platform of TOE	2.8	<<Platform Version>>, <<OS Version>>, <<CSP Version>>
Certificate holder		Name of the Application Provider
Protection Profile Conformance	[EUCC]	PP name, version, Certification ID; if applicable
Name of the ITSEF		If applicable
Name of Certification/Validation Body	[EUCC]	As required
Certification Report Identifier		Unique report reference
Date Issued	[EUCC]	Date of issue
Period of validity		Period of validity
Logos/Marks	[CCRA] J.1-J.3	CC mark, scheme logo, optional RA service mark

8.2 COMBINED CERTIFICATION REPORT CONTENT

The Certification Report includes the assurance results from both the SAR evaluation and the ACO (composition) evaluation. Information common to both may be presented once in a single section and need not be repeated.

Table 5 specifies the content of the Certification Report containing both the results of the SAR and ACO evaluations.

Table 5: Content of the Combined SAR and ACO Certification Report

Section	Content
Executive Summary	• High-level overview of the combined TOE with CSP • certification results and statement
Identification of the TOE	• Application name and version • version of CSP PP, CSP API, CSP Guidance and CSP Tests
Assurance Level of the TOE	• Application EAL with augmentations • CAP package with components • CSP EAL with augmentations
TOE Security Policy	• Description of the security policies enforced by the TOE
Assumptions and Clarification of Scope	• Environmental assumptions • intended usage • limitations of the evaluation scope
Architectural Information	• Overview of the TOE architecture and its CSP integration
Documentation	• Evaluated and certified documentation, such as guidance for users and administrators
IT Product Testing	• Summary of evaluator's independent testing and confirmation of developer tests for both SAR and ACO testing
Evaluated Configuration	• SE platform(s) used for testing, including platform name, operating system, and corresponding versions • CSP implementation version and tested configurations • This section includes the following statement: "The Application was tested on one platform, but can be transferred to other CSP platforms."
Results of the Evaluation	• Overall results, coverage of assurance classes/components, strengths and limitations
Comments/Recommendations	• Advice for secure use or scheme-specific recommendations
Glossary & References	• List of referenced documents and definitions of terms and abbreviations
Annexes (if applicable)	• Additional details, e.g., about evaluation environment, or extended assurance activities

A ANNEX 1: REQUIREMENTS FOR CSPs

This methodology applies only to CSP types and CSP platforms that meet the requirements defined in this appendix. Compliance with these requirements is verified by the evaluator as part of the ACO evaluation activities refined in Chapter 6, ACO , covering the following areas:

- Table 6: Requirements for CSP PP.
- Table 7: Requirements for CSP API.
- Table 8: Requirements for CSP guidance.
- Table 9: Requirements for CSP tests.
- Table 10: Requirements for CSP platforms.

These requirements do not redefine the CSP evaluation but specify the conditions under which an application evaluation according to this methodology is valid. They ensure that all conformant CSP implementations provide the same abstraction, behaviour and asset protection.

If the CSP does not meet these requirements, it is considered non-compliant with this methodology, and application evaluation cannot proceed until compliance is achieved.

Note: Evaluation work may be simplified if the CSP type is already listed on a recognised catalogue of approved CSP types. However, suitability is still application-specific: a CSP type may be acceptable for one application use case but not for another. Therefore, verification of these requirements is always performed in relation to the SFRs of the application, as defined by the ACO Refinements.

Table 6 lists requirements to be fulfilled by the CSP Protection Profile (PP).

Table 6: Requirements for CSP PP

Requirement No	Requirement for CSP PP
REQ-CSP-PP-01	The CSP PP requires CSP certification at least at EAL4 augmented with AVA_VAN.5 and COMP package in a composite evaluation including operating system & hardware.
REQ-CSP-PP-02	The CSP PP requires that the CSP API implementation is included in the scope of the CSP TOE. Note: The CSP PP itself can but does not need to reference the CSP API or CSP guidance or their versions. However, the CSP PP ensures that each CSP ST includes the CSP API implementation and its version, and that the CSP API references the applicable CSP guidance and version.
REQ-CSP-PP-03	The CSP PP requires strict conformance of any ST claiming conformance to this CSP PP.
REQ-CSP-PP-04	The CSP PP requires the CSP to securely store and process its configuration, assets and sensitive data within its tamper-proof environment, ensuring they are not directly accessible by the application and cannot be exposed.
REQ-CSP-PP-05	The CSP PP ensures strict isolation and encapsulation of CSP configurations, assets, CSP services, and authentication mechanisms when multiple application use cases are simultaneously supported on the same platform.
REQ-CSP-PP-06	The CSP PP enforces authorization mechanisms to ensure that only authorised entities (e.g., when applications act as proxy for remote entities) can invoke the CSP services for their dedicated use case.

Requirement No	Requirement for CSP PP
REQ-CSP-PP-07	The CSP PP supports secure CSP configuration deactivation or deletion as part of its lifecycle management to prevent residual sensitive data exposure or misuse.
REQ-CSP-PP-08	The CSP PP includes an AGD_OPE refinement or an equivalent mechanism that prohibits CSP platform ST authors from introducing vendor-specific security rules beyond the generic CSP guidance that could impact Applications. This may be supported by: An AVA_VAN evaluation confirming that no additional vendor-specific CSP usage guidelines are required to mitigate vulnerabilities. Evaluating the CSP without dependencies on vendor-specific guidance or additional Evaluation Technical Reports for Composition.

Table 7 lists requirements to be fulfilled by the CSP API.

Table 7: Requirements for CSP API

Requirement No	Requirement for CSP API
REQ-CSP-API-01	The CSP API defines a harmonised interface for CSP security functionalities that align with the SFRs specified in the CSP PP and enable applications to rely on them.
REQ-CSP-API-02	The CSP API unambiguously references the CSP guidance and its version, to be followed to ensure consistent security behaviour across application implementations.
REQ-CSP-API-03	The CSP API uses handles for asset interaction and enforce access control, preventing direct exposure of sensitive assets to applications.
REQ-CSP-API-04	The CSP API provides mechanisms for applications to ensure that the required API version, features and algorithms are available on the CSP platform in use.
REQ-CSP-API-05	The CSP API provides mechanisms for applications to ensure that the active CSP configuration is compatible with its use case.

Table 8 lists requirements to be fulfilled by the CSP guidance.

Table 8: Requirements for CSP guidance

Requirement No	Requirement for CSP guidance
REQ-CSP-GDN-01	The CSP guidance defines harmonised security rules for applications using a CSP type.
REQ-CSP-GDN-02	The CSP guidance ensures that applications adhering to these rules run securely on all platforms supporting this CSP type.
REQ-CSP-GDN-03	The CSP guidance specifies requirements for the secure installation and initialisation of the CSP.
REQ-CSP-GDN-04	The CSP guidance defines security considerations for the lifecycle of applications using a CSP type, ensuring that critical security aspects remain consistent across different lifecycle phases.
REQ-CSP-GDN-05	The CSP guidance defines rules for the CSP Admin on how to manage and configure the CSP, including configuration versioning and minimal access control settings.

Table 9 lists requirements to be fulfilled by the CSP tests.

Table 9: Requirements for CSP tests

Requirement No	Requirement for CSP tests
REQ-CSP-TST-01	The CSP tests are assumed to verify conformance of the CSP to the CSP API.
REQ-CSP-TST-02	The CSP tests are assumed to be part of a structured compliance process, including at least self-assessment by the manufacturer.
REQ-CSP-TST-03	The CSP tests provide sufficient coverage to verify conformance of the CSP to the CSP API for security features required by the application.

Table 10 lists requirements to be fulfilled by the CSP platforms.

Note: In the context of the application evaluation under this methodology, the representative CSP platform is considered as the base component for composition. However, its conformance to the CSP PP, CSP API, and CSP tests is assumed (A.Plat-CSP) rather than being subject to direct verification or testing.

Table 10: Requirements for CSP platforms

Requirement No	Requirement for CSP platforms
REQ-CSP-PLT-01	The CSP platform is assumed to be certified with strict conformance to the CSP PP.
REQ-CSP-PLT-02	The CSP platform is assumed to implement the CSP API.
REQ-CSP-PLT-03	The CSP platform is assumed to prove CSP API conformance using CSP tests.
REQ-CSP-PLT-04	The CSP platform is assumed to ensure that applications adhering to the CSP guidance do not introduce security issues.

B ANNEX 2: SAMPLE CSP SERVICES

CSP services encapsulate security functionalities so that applications can rely on them and inherit the CSP's security assurance (see Sections 2.5). Amendment N – CSP **[GP-AMD-N]** of the GlobalPlatform Card Specification describes how to implement such services using Java Card and GlobalPlatform technologies. Examples include:

- **Cipher and signatures**: Ready-to-use cryptographic operations for encryption, decryption, signing and verification that utilise pre-configured cryptographic keys and fixed algorithms. Authenticated Applications can securely invoke them in a controlled manner, allowing them to inherit CSP security assurance.
- **Encryption transformations**: An atomic cryptographic operation that converts already-encrypted data from one key and algorithm to another, using pre-configured keys and fixed algorithms. This enables authenticated applications to securely handle sensitive data without ever accessing it in plaintext, ensuring protection even if the application is compromised.
- **Secure messaging**: A CSP-managed secure channel protocol flow that ensures the correct sequence of steps for each authentication protocol while using pre-configured cryptographic keys and certificates. Typical protocols include EACv1 for MRTD, EACv2 for eID, PACE-only, PACE-CAM and SCP03.
- **Attestations**: Provides mechanisms to retrieve signed evidence from the CSP that verifies the authenticity and identity of critical components, such as platform attestation to prove the identity of the SE platform, CSP self-attestation, proof-of-possessions of key pairs, and attestations for counter values or other elements managed by the CSP.
- **Counter and limits**: Support for secure counters that can be incremented either by the application or automatically by the CSP. Examples include usage counters that track the number of times a specific key is used in CSP services. Fixed limits on these counters enforce security policies, such as preventing the use of keys whose usage limits have been exceeded.
- **Timer and time management**: Supports a system time that the CSP can use to evaluate timers, such as validity periods for keys, certificates or passwords. It also enables use-case-specific timers with fixed expiration dates, for example, to verify whether a person is over 18.
- **PIN and passwords**: Manages PINs and passwords, including authentication status, retry counters, blocking and PUK-based unblocking, all handled within the CSP's secure environment.
- **Secure auditing**: Logs asset usage events (e.g., signature creations), security incidents (e.g., password failures) and system events (e.g., time updates). The CSP ensures the integrity of these logs by incorporating a counter and/or timestamp and signing it.

Such CSP services are provided through the CSP API, ensuring that applications can securely use them without directly handling assets or impacting security properties. This enables the reuse of CSP's security functionalities without requiring their separate evaluation under this methodology.

The CSP API defined in GlobalPlatform Amendment N **[GP-AMD-N]** is complemented by the CSP Guidance for Applet Developers **[GP-GUI-CSP]**, which provides security rules for Java Card applets using the GlobalPlatform CSP. Such rules are, for example, intended to ensure the secure interaction between the Client Application and the CSP. For instance, security-sensitive results returned by CSP operations shall be verified using the CSP's sensitive result verification mechanism. Non-compliance with this guidance may invalidate assumptions on which the application evaluation relies and may therefore affect the applicability of the CSP security guarantees to the application.

Together, the GlobalPlatform documents **[GP-PP-CSP], [GP-AMD-N]** and **[GP-GUI-CSP]** constitute a concrete CSP Type, the "GlobalPlatform CSP for Java Card Applets". Other CSP Types, such as a "CSP for HSM solutions" or a "CSP for non-Java Card-based SE Applications", may use the same CSP API as defined in **[GP-AMD-N]** but different Protection Profiles to support different hardware platforms.

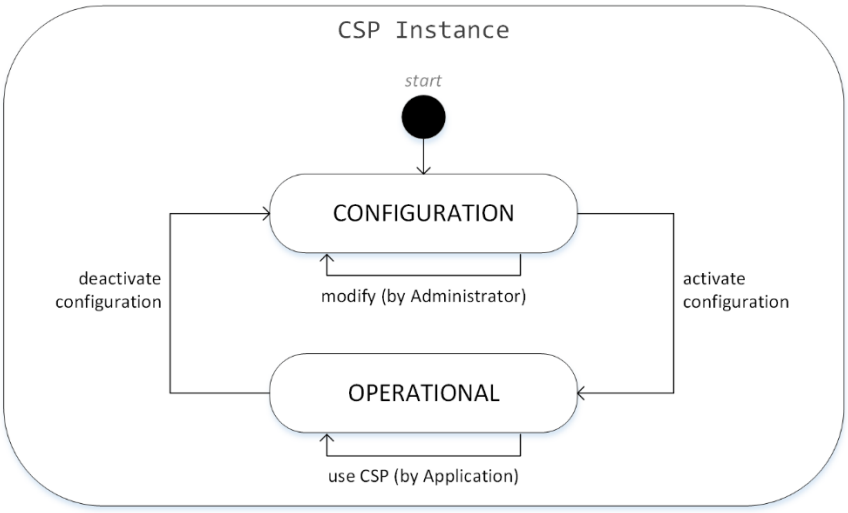
C ANNEX 3: SAMPLE CSP CONFIGURATION

Typical CSP services, such as those in Annex 2: Sample CSP Services, rely on pre-configured cryptographic assets, such as keys, certificates and passwords. These assets are securely managed within the CSP, preventing exposure or misuse.

Before Applications can use CSP services, the CSP is first initialised with these assets. This is referred to as the CSP configuration, which is typically done using a script.

Different CSP types manage their CSP configuration(s) in various ways. Simple CSP types may have a fixed, hardcoded configuration embedded in the CSP, with no possibility for modification. Most CSP types offer an administration interface, allowing administrators to adjust the CSP configuration based on the application needs, following the lifecycle phases shown in Figure 6.

Figure 6: Sample lifecycle for typical CSP types



In the configuration phase, an administrator sets up the CSP configuration. Once configured, the operational phase begins, allowing the application to use cryptographic services with the predefined assets.

Examples of a CSP configuration include:

- Creating slots for assets such as cryptographic keys, certificates or passwords.
- Configuring asset properties, including key size, curve, password length and certificate type.
- Restricting asset usage to a specific CSP service and algorithm.
- Configuring authentication and access control per application and CSP configuration.
- Importing certificates, initialising passwords, and setting cryptographic key values via import, random key generation or key derivation functions.

To ensure that assets from the CSP configuration remain securely managed within the CSP while still being usable by applications, CSPs support asset handles. Each asset is assigned a unique identifier, managed within the CSP and shared with the application.

Using this identifier, applications can invoke CSP API operations without directly handling sensitive assets, ensuring controlled access to CSP services while keeping all processing within the CSP's tamper-proof environment, preventing unauthorised exposure or misuse.

More advanced CSP types may allow CSP configuration modifications even during the operational phase. However, CSP services offering such CSP management operations still adhere to the encapsulation principle, ensuring security and preventing direct asset manipulation by the application. This can be achieved, for example, through:

- Fine-grained access control at the asset level, e.g., restricting key generation within the CSP to specific assets and asset states.
- Support for key agreement (e.g., ECKA-DH) and key derivation (e.g., AES-CMAC KDF, HKDF) through CSP services, while ensuring that all key manipulations remain within the CSP.
- Double encryption layers for importing certificates or keys, using both a secure channel and asset-level encryption.
- Cryptographically secured rollover mechanisms for certificate management (e.g., TA Linked).
- Secure password change procedures, ensuring that passwords are updated only in a single atomic operation after verifying the old password, using an encryption mechanism.
- Secure offloading, allowing encrypted assets to be exported and later re-imported, ensuring that only the originating or clustered CSP instances can decrypt them.

Figure 7 illustrates a sample CSP configuration with two slots for cryptographic keys (one imported, one generated), two slots for passwords (one PIN, one PUK) and fine-grained access control configuration for each asset. Unique asset identifiers are set within the script.

Figure 7: Sample CSP configuration

```
Install CSP // install a new CSP for each use-case
Install Client Application "AID1" // install the Client Application with AID "AID1"
Register Client Application "AID1" // register the AID of Client Applet to use the CSP

Create Resource           "1, TYPE_ECC_KEY, USAGE_KEYMGT, ALG_ECDH,.." //create ECC key
Import Key Value          "1, data" // import ECC value that shall be used for ECDH  key agreement

Create Resource           "2, TYPE_AES, USAGE_CIPHER, ALG_AES_CBC,.." // create AES key
Generate Key Value        "2" // generate a AES key value
Configure Access Control "2, GRANT_CLEAR,.." // allow Client Applet clear&derive key "2"

Create Resource           "3, TYPE_PASSWORD, tryLimit=4,.." // create PIN as resource id "3"
Create Resource           "4, TYPE_PASSWORD, tryLimit=10,.." // create PUK as resource id "4"
Configure Access Control "3, UNBLOCK_PUK=4,.." // configure PIN/PUK dependency

Activate Configuration // finalize the configuration
```

Note that some assets exist outside the CSP configuration but are required for CSP initialisation, such as for deployment and administrative authentication. Protection and verification of these assets are the responsibility of the CSP Admin in the operational environment, not of the CSP TOE itself. Related aspects are addressed in the refinements to ADV_TDS (see Section 5.2.2).

Examples of such deployment-related assets include:

- Keys for remote management of the SE platform.
- Keys to access the Security Domain (SD) of the CSP instance and/or application.
- Data Encryption Keys (DEK) used for loading sensitive data into the CSP instance.
- Assets imported into the CSP instance during initialisation or operational use.

Conclusion: Assets managed by the CSP enable applications to rely on CSP services without requiring direct handling or additional security measures. While these assets are securely encapsulated within the CSP, deployment and authentication assets may require additional safeguards to ensure a secure initialisation process.

D ANNEX 4: SAMPLE CSP MULTI APPLICATIONS SUPPORT

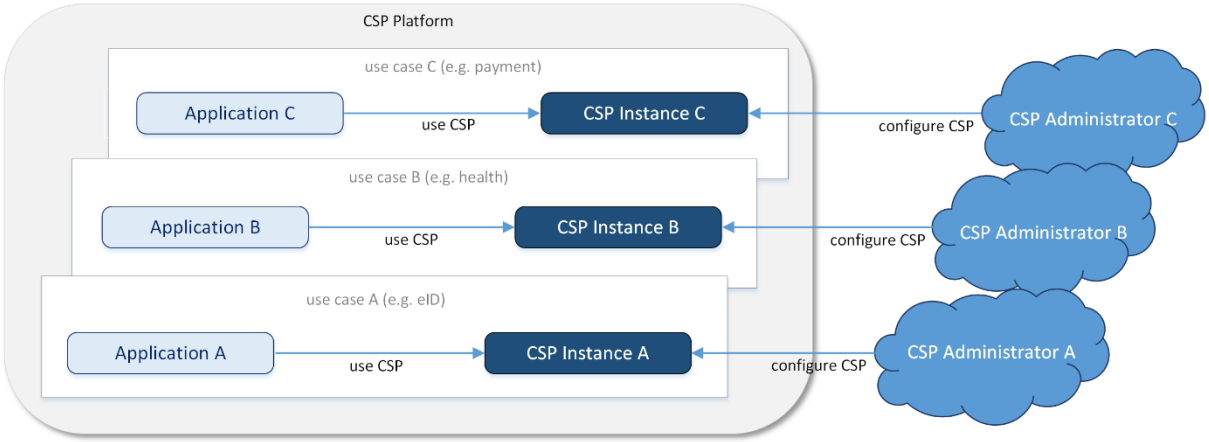
A key architectural decision for CSP types is whether they should support multiple applications use cases running in parallel on the same CSP platform.

To ensure strict separation between different application use cases — such as eID applications from various countries running on the same SE — the CSP supports the concept of CSP instances, as illustrated in Figure 8.

Each CSP instance has its own dedicated CSP configuration, ensuring that only authorised applications can access the relevant CSP services for their specific use case. When an application is removed or revoked, its CSP instance can be securely deactivated or deleted, ensuring no residual sensitive data remains accessible.

A CSP instance can be implemented as a separate system process or as a virtual instance within a shared process (e.g., as separate threads). Regardless of the implementation approach, the CSP should ensure that assets, configuration data, services and authentication for each use case remain strictly isolated. This requirement is addressed in the CSP PP (see Annex 1: Requirements for CSPs).

Figure 8: Sample CSP multi applications support



For example, the GlobalPlatform-CSP, as specified in **[GP-AMD-N]**, utilises the install "for install and make selectable" command from **[GP-CardSpec]** to create dedicated CSP instances for each application use case. This process involves different types of administrators, including:

- SE administrator: Typically the SE owner with management privileges over the SE. This role is responsible for creating a new CSP instance dedicated to an application use case and granting access rights to manage its Security Domain (SD) to a CSP administrator.
- CSP administrator: Typically the application developer responsible for a specific application use case. The CSP administrator creates the CSP configuration, authorises applications and transitions the CSP instance to OPERATIONAL state, allowing authorised applications to use its CSP services for the intended use case.

In this architecture, isolation between different Application use cases is enforced through platform mechanisms such as GlobalPlatform Security Domains (SDs), as defined in **[GP-CardSpec]**. This is achieved through dedicated key management, strict access control policies and cryptographic domain separation, ensuring that administrators and applications can only interact with their respective CSP instance.

E ANNEX 5: SAMPLE CSP USER-LEVEL AUTHORISATION

In addition to authentication and authorisation mechanisms for applications using CSP services, a CSP type may also need to support user-level authentication and authorisation. This is particularly relevant in scenarios where an application using a CSP is accessible by multiple users.

These scenarios can be addressed through the following CSP security features **[GP-AMD-N]**:

- Access Control Rules (ACRs): Fine-grained access control per asset and application restricts CSP services to authorised Applications. The CSP needs to be able to unambiguously identify Applications to enforce these ACRs. Remote Applications running on a different platform than the CSP even require additional authentication mechanisms.
- Secure Channels: User authentication via secure channels between Off-Card Entities (OCE) and the CSP, routed through the application. The CSP needs to enforce that its CSP services can only be invoked when a secure channel is established (e.g., PACE authentication). CSPs that support remote applications need to restrict CSP service invocation to occur exclusively through the secure channel.
- Policies: Dynamic, asset-dependent policy rules, configured and enforced within the CSP, can restrict CSP services based on conditions. For example, access to a CSP's signing service may require successful verification of a user PIN (e.g., via PACE authentication or basic password verification).

Note: This annex describes exemplary approaches for CSP support in authentication and authorisation. The actual implementation depends on the CSP type and its specifications, as defined in the corresponding CSP Protection Profile (CSP PP), and may vary across different CSP types.

Sample ACRs:

Table 11 presents an example bitmask for Access Control Rules (ACRs) that can be applied to assets such as keys, passwords, certificates or data managed within the CSP. The defined rights include:

- USE: Permits the asset to be used in cryptographic operations.
- SETUP: Allows asset initialisation via a CSP service (e.g., key generation, derivation, key agreement or changing a password).
- CLEAR: Enables asset un-initialisation via a CSP service (e.g., remove password or key value).
- MOVE: Allows the asset, along with its properties, to be exported (encrypted) for offloading or clustering.

The USE right applies generally to CSP security operations and does not specify the exact CSP service an asset is used for. This is intentional, as assets within the CSP are typically pre-configured for a specific purpose and a dedicated algorithm, resulting in a one-to-one relationship between the asset and the CSP service. Consequently, service-specific access rights are usually not required.

Table 11: Sample Access Control Rules Bit Positions

Role	USE	SETUP	CLEAR	MOVE
Local applications	1	2	3	4
Owner application	5	6	7	8
Administrators	9	10	11	12
Remote applications	13	14	15	16

These rights are assigned per asset to the following roles:

- Local applications: Registered applications using the CSP on the same platform.
- Owner application: A specific application with exclusive access.
- Administrators: Entities authorised to perform CSP management operations, such as asset creation and security configuration.
- Remote applications: Authenticated applications using the CSP from a different platform.

To identify applications, the CSP provides a registration mechanism that enables it to implement appropriate security measures, such as:

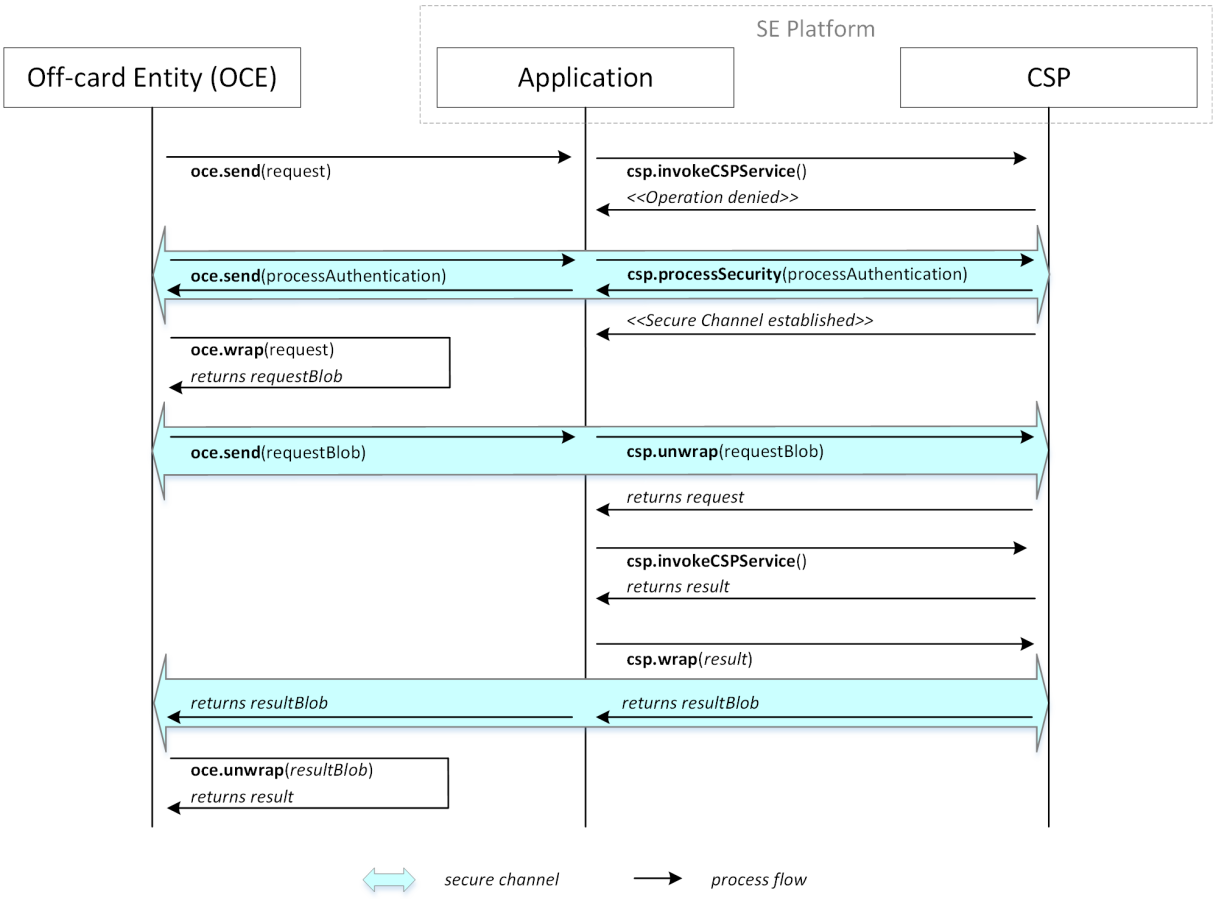
- AID Check: Using a platform-specific identification mechanism, such as comparing the application Identifier (AID) on the SE to determine the application and its associated SD.
- Hash Check: Verifying an application hash to ensure integrity and authenticity.
- Authentication: Establishing a secure channel between the application and CSP for authentication (e.g., for applications not operating on the same platform as the CSP).

Sample Secure Channel Authentication:

While Access Control Rules (ACRs) define which roles can use specific assets for CSP services, a secure channel ensures that such access is only granted under authenticated conditions.

Figure 8 illustrates a typical secure channel flow between an Off-Card Entity (OCE) and an application using a CSP, where both the application and CSP operate on the same platform.

Figure 9: Sample CSP Secure Channel Flow



Before processing a CSP service, the CSP requires a secure communication channel to be established. If an OCE sends a request to the application, and the application attempts to invoke a CSP service, the CSP will deny the request unless authentication has been completed.

To authenticate, the OCE initiates the process by sending messages based on a supported authentication protocol (e.g., PACE). The application forwards these messages to the CSP, which processes the authentication and generates response messages. The application then sends these responses back to the OCE. This exchange continues until the secure channel is successfully established.

Once the secure channel is active, the OCE sends session-encrypted requests to the application. The authorised application (see ACR) decrypts the request using the CSP unwrap operation, processes it, and invokes the necessary CSP services. After processing, it re-encrypts the response using the CSP wrap operation and returns it to the OCE.

Sample policy concept:

- Two-Person Verification: Requires verification of two or more passwords before allowing the requested action.

Figure 10 illustrates how policies can be used to restrict CSP service usage based on specific internal states known by the CSP. This policy concept eliminates the need for security conditions within application code, mitigating the risk of external attacks targeting these conditional statements.

A policy rule defines dependencies for associated key and/or password assets and is configured as part of the CSP configuration. The CSP enforces these rules whenever the application initiates relevant operations, ensuring that security conditions are evaluated securely within the CSP.

Example policies:

- Unblock: Allows unblocking a password only when an associated PUK is verified.
- Signing/Decryption PIN: Permits signing or decryption operations only if an associated password is verified.
- Signing/Decryption if secure channel established: Permits signing or decryption only if an associated secure channel is established.
- Decryption with TA access: Permits decryption only if a specific access flag is provided by the OCE through Terminal Authentication 2 (TA2).
- Two-Person Verification: Requires verification of two or more passwords before allowing the requested action.

Figure 10: Sample CSP policies

Application using Java Card API:

```
keyPair = new KeyPair(algorithm, size);
keyPair.genKeyPair();
pin = new OwnerPin(tryLimit, maxSize);

sig = Signature.getInstance(algorithm);
sig.init(keyPair, signMode);

if (pin.check(inputPin)) {
  output = sig.sign(inputData);
}
```

Application using CSP API supporting Policies:

```
pinService = csp.getPasswordService();
sigService = csp.makeSignatureService();
sigService.init(keyId, signMode);
..
pinService.check(pinId, inputPinData);
try {
  output = sigService.sign(inputData);
} catch (CSPException e) {
  // handle PIN is not in state authenticated
}
```

F ANNEX 6: SAMPLE CSP PLATFORM DEPENDENCIES

A CSP serves as an abstraction layer that standardises cryptographic operations through a unified API, to provide secure CSP services.

Instead of implementing custom security functionalities, the CSP can rely on security mechanisms provided by the underlying platform. This is acceptable since the CSP undergoes a composite AVA_VAN.5 evaluation, which includes the operating system and hardware, as mandated by the requirements for the CSP PP (see AAnnex 1: Requirements for CSPs).

For example, the GlobalPlatform-CSP, as specified in **[GP-AMD-N]**, relies on security functionalities from the SE platform, which are evaluated under **[GP-SE-PP]**. These include Card Management, Secure Communication, Privilege Management, Life Cycle Management, ELF Loading, Data Loading and Key Loading. By leveraging these evaluated platform functionalities, the CSP can:

- Remotely create new Security Domains (SDs) dedicated for the application's use case.
- Assign privileges to these SDs, granting control to the CSP administrator, who acts on behalf of the application developer.
- Change the access keys for the newly created SDs, ensuring that only the CSP administrator has access while the SE administrator no longer retains knowledge of these keys.
- Create a new CSP instance associated with one of the newly created SDs.
- Configure the CSP instance according to the application's requirements.
- Securely import sensitive resource values into the CSP instance, such as passwords and private or symmetric keys.
- Load and install the application within one of the new SDs.
- Ensure transport-layer encryption (e.g., SCP03 or SCP11) is applied to all these operations.

Security functionalities from the Java Card operating system, evaluated under **[JC-PP]**, include Application Post-Issuance, Firewall, Sharable Interfaces, Global Arrays, Sensitive Array and Sensitive Result. These functionalities support the CSP by enabling:

- Hosting the CSP instance within the Java Card environment.
- Hosting the application instance alongside the CSP.
- Enabling the CSP to verify whether an application is authorised to utilise CSP services.
- Ensuring integrity checks on CSP service results.
- Validating the integrity of parameters passed to CSP services.

By relying on evaluated platform security functionalities, the CSP simplifies its implementation, being able to focus on exposing a harmonised, secure API for cryptographic service. This reduces the need for custom security development while ensuring a standardised and secure interface for applications.

G ANNEX 7: SAMPLE CSP WITHIN eUICC

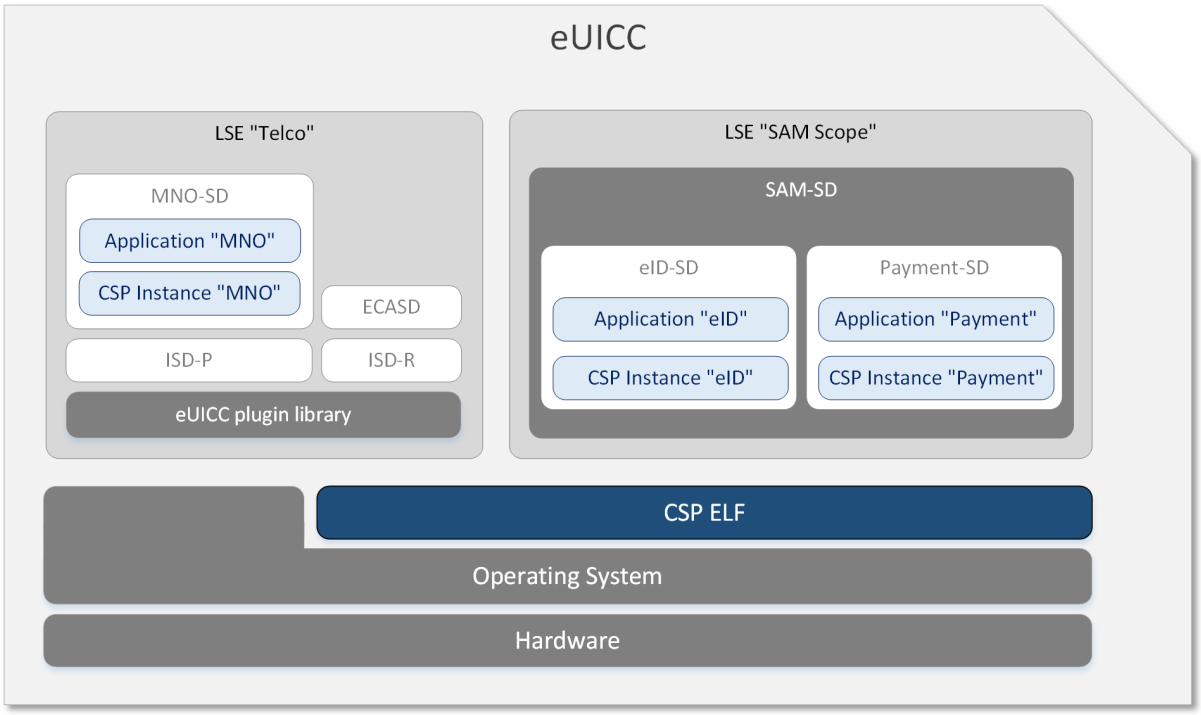
An eUICC (embedded Universal Integrated Circuit Card) is a specialised type of SE designed specifically for mobile network applications (Telco) and, with the Secured Applications for Mobile (SAM) standard, may support additional application scenarios such as eID or payment.

To ensure isolation between different Mobile Network Operators (MNOs) and Application Service Providers (ASPs), the eUICC provides multiple levels of logically separated secure areas, ensuring independent operation, deployment, isolation and security enforcement. These include:

- Logical Secure Elements (LSEs): Extend isolation beyond SDs by creating separate logical SE instances within a single physical SE. This enables multiple stakeholders, typically mobile operators, to coexist securely with independent security policies and key management.
- Security Domain (SD): A dedicated secure space within an SE that manages application installations, enforces security policies and controls access to the applications it contains.
- SAM SD: Builds on LSEs by extending their security model beyond telecommunications to applications such as payments and eID solutions. While LSEs separate stakeholders, SAM SDs enforce stricter cryptographic controls and management functionalities.

The GlobalPlatform-conform CSP, as specified in **[GP-AMD-N]**, is designed to support both Telco and SAM use cases. The CSP Executable Load File (ELF), which is the binary of the CSP implementation, can be stored on the eUICC platform layer as part of the operating system, as shown in Figure 11. Alternatively, a separate CSP ELF may exist within each LSE. In either case, the CSP ELF enables the instantiation of new CSP instances in dedicated sub-SDs within both MNO-SD and SAM-SD, allowing different service providers to operate across LSEs.

Figure 11: Sample eUICC architecture with CSP



ABOUT ENISA

The European Union Agency for Cybersecurity, ENISA, is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe. Established in 2004 and strengthened by the EU Cybersecurity Act, the European Union Agency for Cybersecurity contributes to EU cyber policy, enhances the trustworthiness of ICT products, services and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. Through knowledge sharing, capacity building and awareness raising, the Agency works together with its key stakeholders to strengthen trust in the connected economy, to boost resilience of the Union's infrastructure, and, ultimately, to keep Europe's society and citizens digitally secure. More information about ENISA and its work can be found here: www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
Agamemnonos 14
Chalandri 15231, Attiki, Greece

Heraklion Office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

Brussels Office
Rue de la Loi 107
1049 Brussels, Belgium

enisa.europa.eu