Security Evaluation Standard for IoT Platforms

Version 1.3

Contents

1 Introduction 6
1.1 Structure of this document 6
1.2 Terms used in this document 6
1.3 IoT use case and threat model 7
1.4 IoT product lifecycle 7
1.5 Additive composition 9
2 Security Functional Requirements 10
2.1 Identification and attestation of platforms and applications 11
2.1.1 Identification of platform type 11
2.1.2 Identification of individual platform 11
2.1.3 Genuine platform instantiation 12
2.1.4 Secure initialization of platform 12
2.1.5 Attested secure state of platform 12
2.1.6 Genuine application 13
2.1.7 Attested state of application 13
2.2 Product lifecycle: Factory reset / Install / Update / Decommission 14
2.2.1 Factory reset of platform 14
2.2.2 Secure install of application 14
2.2.3 Secure update of platform 15
2.2.4 Secure update of application 16
2.2.5 Decommission of platform 16
2.2.6 Field return of platform 17
2.3 Secure communication 18
2.3.1 Secure communication support 18
2.3.2 Secure communication enforcement 18
2.4 Extra attacker resistance 20
2.4.1 Limited physical attacker resistance 20
2.4.2 Physical attacker resistance 20
2.4.3 Software attacker resistance: isolation of platform 21
2.4.4 Software attacker resistance: isolation of application parts 21
2.4.5 Software attacker resistance: isolation of platform parts 22
2.5 Cryptographic functionality 23
2.5.1 Cryptographic operation 23
2.5.2 Cryptographic key generation 24
2.5.3 Cryptographic keystore 24
2.5.4 Cryptographic random number generation 25
2.6 Compliance functionality 26

2.6.1 Secure encrypted storage 26
2.6.2 Residual information purging 26
2.6.3 Audit log generation and storage 27
2.6.4 Reliable time 27
2.6.5 Secure debugging 27
3 Security Assurance Requirements 29
3.1 SESIP Assurance Level 1 (SESIP1) 30
3.1.1 Objectives 30
3.1.2 Assurance components 30
3.1.3 Security Target Requirements 30
3.1.4 Life-cycle Support Requirements 33
3.1.5 Vulnerability Assessment Requirements 33
3.2 SESIP Assurance Level 2 (SESIP2) 34
3.2.1 Objectives 34
3.2.2 Assurance components 34
3.2.3 Security Target Requirements 35
3.2.4 Development Requirements 35
3.2.5 Guidance Documents Requirements 35
3.2.6 Life-cycle Support Requirements 35
3.2.7 Tests Requirements 35
3.2.8 Vulnerability Analysis Requirements 35
3.3 SESIP Assurance Level 3 (SESIP3) 37
3.3.1 Objectives 37
3.3.2 Assurance components 37
3.3.3 Security Target Requirements 38
3.3.4 Development Requirements 38
3.3.5 Guidance Documents Requirements 39
3.3.6 Lifecycle Support Requirements 39
3.3.7 Tests Requirements 39
3.3.8 Vulnerability Analysis Requirements 40
3.4 SESIP Assurance Level 4 (SESIP4) 41
3.4.1 Objectives 41
3.4.2 Assurance components 42
3.4.3 Security Target Requirements 42
3.4.4 Development Requirements 42
3.4.5 Guidance Documents Requirements 43
3.4.6 Lifecycle Support Requirements 43
3.4.7 Tests Requirements 44

3.4.8 Vulnerability Analysis Requirements 44
3.5 SESIP Assurance Level 5 (SESIP5) 45
3.5.1 Objectives 45
3.5.2 Assurance components 45
4 Guidance: Vulnerability analysis and attack potential rating 47
4.1 Attack rating calculation in the sum of the identification and the exploitation phases 47
4.2 Physical (local) attacks and remote attacks 47
4.3 Attack potential rating 47
5 Example use cases 50
5.1 Generic examples 50
5.1.1 IoT cloud connectivity platform 50
5.1.2 Root-of-Trust based on a microcontroller 51
5.2 Examples for specific use cases 53
5.2.1 Secure update of a device (OTA) 53
5.2.2 A blood glucose measurement device (DTSec) 54
6 Guidance for schemes implemented using SESIP 56
6.1 Evaluation lab 56
6.2 Certification body 56
7 References 57
7.1 Terminology 57
7.2 Bibliography 57
7.3 Revision history 58

1 Introduction

This document describes the Security Functional Requirements and Security Assurance Requirements for the Security Evaluation Standard for IoT Platforms (SESIP).

This document is final and for use in SESIP V1.3 evaluations.

1.1 Structure of this document

This document is structured as follows:

Section 1.2 defines a number of terms that are used in this document. Most definitions rely on, and assume knowledge of, definitions in the CC and CCRA.

Chapter 2 defines the Security Functional Requirements. It completely replaces CC Part 2 for this standard. CC Part 2 is not used in SESIP.

Chapter 3 describes the Security Assurance Requirements. It defines five hierarchical levels of assurance for IoT platforms ("alternative EALs"). This chapter also defines additional assurance components that are used in these levels, but are not used in CC part 3.

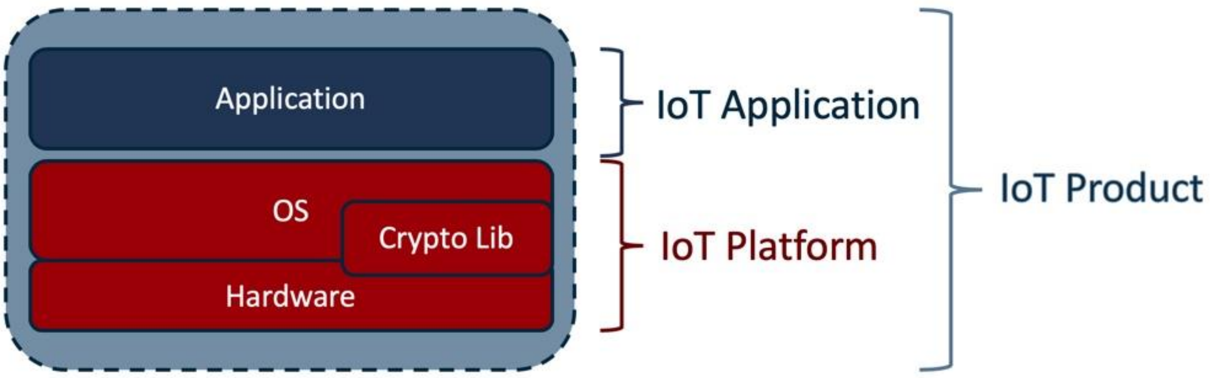
Appendix 4 describes the guidance for the vulnerability analysis and attack potential rating methodology.

Appendix 5 provides a number of use-cases as worked examples.

Appendix 7 provides the guidance for schemes implemented using SESIP.

Appendix 7 contains the references.

1.2 Terms used in this document



An **IoT Platform** is the hardware/software providing an operating environment for an IoT Application. IoT Platforms parts can be developed and evaluated separately, for example by evaluating the cryptographic library, an OS, hardware, and then combining them. In terms of the traditional Common Criteria, the IoT Platform (part) identified in the ST is our TOE.

An **IoT Application** is the software running on the IoT Platform adding domain-specific functionality. An IoT Platform together with an IoT Application in total form an **IoT Product**, providing the user with a complete functionality. From the platform point of view, there is only one IoT Application, even if this IoT Application is separated in many different applications parts from the IoT Application developer point of view.

For brevity, “IoT Platform”, “IoT Application” and “IoT Product” are abbreviated to “platform”, “application” and “product” in this document.

Platform developers build platform parts and supply these to composite developers. They will be able to market more effectively if they can show that their platform provides some functionality in a secure manner, so the composite developer can reduce the cost and risk of composite evaluations based on that platform.

Platform evaluators and certifiers verify whether a platform is secure in accordance with the standard described in this document.

Composite developers build their own platform extensions or application upon an underlying platform. Composite developers of a product want to provide their users with assurance that the product is secure, and for this reason may submit their product to product-specific schemes. In addition to the Users’ assets, the composite developers may also consider the confidentiality, integrity and authenticity of their platform extensions or applications as assets.

Composite evaluators and certifiers of a platform verify whether a composed platform meets the requirements by reusing the underlying assurance. Composite evaluators and certifiers of a product verify if product is secure in accordance with their product-specific scheme. To save work, composite evaluators and certifiers wish to re-use evaluation results from any platforms that are used by the composite developer.

Product vendors are vendors of complete IoT products based on the IoT Platform (part) certified. Product vendors assemble IoT Platform parts and applications into products, and they deliver these products to the users.

Users are end-users of an IoT product. Users want to know their product is secure in the way they expect for the product type, which at least includes confidentiality of their data, and is likely to also include integrity of their data and the provision of specified security services by the product. They probably neither know nor care what platform the product is built on.

Regulators want to make sure that IoT products are secure. They can therefore mandate that products are approved by a product-specific scheme or use secure platforms as certified against this Standard.

1.3 IoT use case and threat model

IoT is a broad term, but always contains a product (“thing”) and some form of connectivity (“internet”). SESIP focuses on the “thing” side of IoT, and on the security of connected devices.

The minimum and mandatory threat model in SESIP is an attacker with only remote (no physical) access to the platform during the exploitation phase. This addresses the main IoT concern of a scalable attack exploited using only the “internet” connection to the platform. This attacker might identify this possible attack on a platform acquired for this identification purpose, so physical attacks during the identification phase of the attack are in scope.

Platforms can expand the threat model considered with the use of the SFRs “Limited physical attacker resistance”, “Physical attacker resistance”, “Software attacker resistance: isolation of platform“, “Software attacker resistance: isolation of application parts”, and “Software attacker resistance: isolation of platform parts”.

1.4 IoT product lifecycle

Different lifecycle models can be applied to IoT products, and to the IoT platform and application that compose that products. Nevertheless, some patterns can be found in most products, and they are significant for security:

Vendor provisioning is the phase during which the product is provisioned with credentials that are shared with the vendor’s backend, and that allow the product to communicate securely with the backend and to perform management operations. This phase typically concludes with the delivery of the product to the customer.

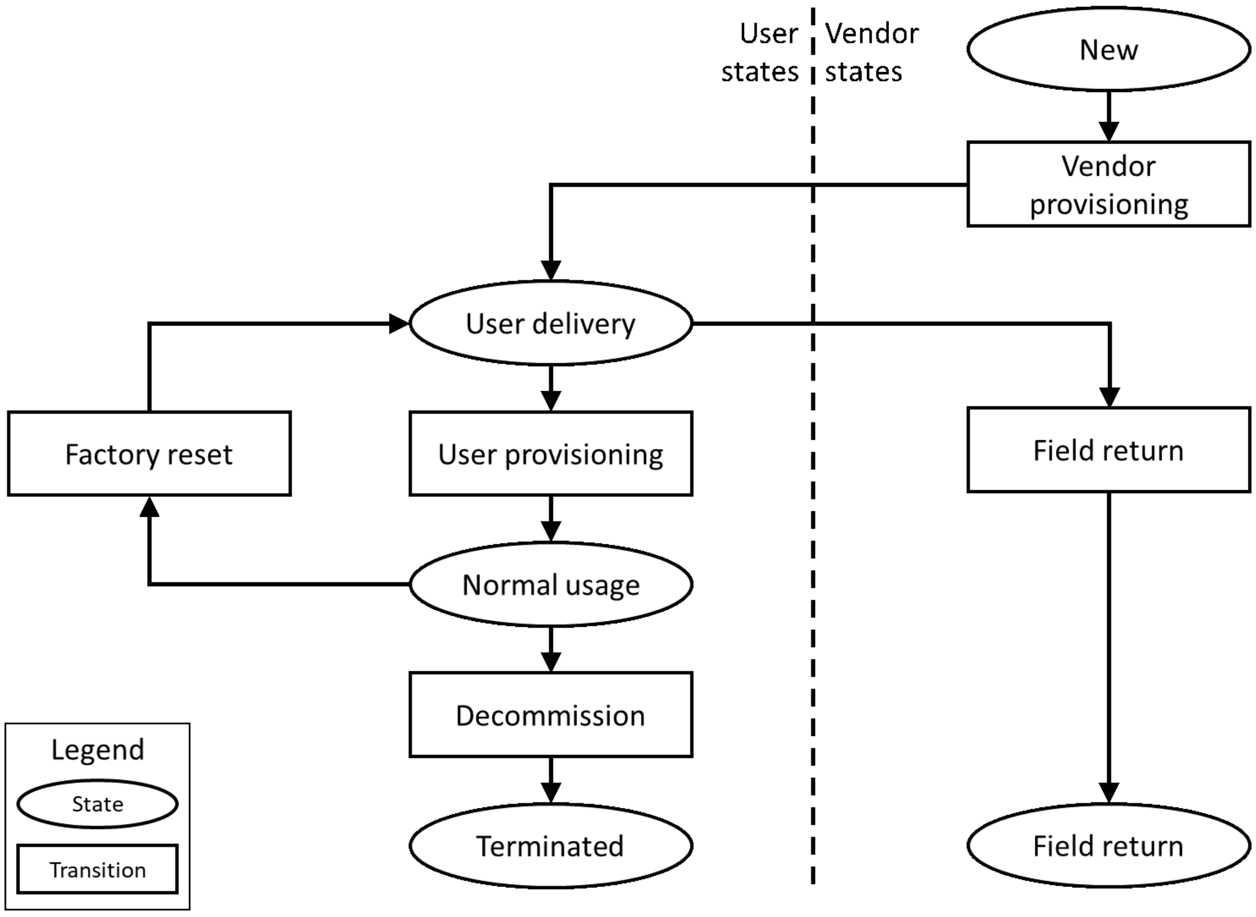
User provisioning is the phase during which the product is provisioned with a user's credentials and specific data, that allow the product to represent that user. This phase typically concludes with the normal usage phase of the product.

Normal usage is supposed to be the product's normal state, until one of the following events occurs:

- The user applies a factory reset, which removes all user-related data and credentials, and prepares the product to be transferred to another entity (*e.g.*, for resale, for return, or even for temporary storage). The product is then ready again for user provisioning, but a user should not have the ability to return the product to an earlier lifecycle phase.
- The user terminates the product, before discarding it. This **Terminated** state is irreversible.

Some devices may include an additional state related to **Field return**, during which specific debugging features may be available. All user data and credentials must have been removed before reaching that state.

The product lifecycle shown in the diagram below is used as a reference in the SFRs when references to a lifecycle are required:



Note that the vendor states are only reachable by the vendor, either before delivery of the product, or after return of the product by the user.

In addition to the product lifecycle, platform and some of its parts may have different lifecycles that are also significant to security. Such security lifecycles are product-specific, and their contribution to the platform's security should be described in the corresponding Security Target.

1.5 Additive composition

The security claims of a platform (part) are valid only when the objectives for the environment (as described in the guidance indicated by the ST) are met. This means that any user of a platform (part) needs to ensure the objectives for the environment are met, also another platform (part) composing with the platform part.

Composition under SESIP therefore is the verification that the objectives for the environment of the parts are met for such a composed platform.

This applies at the moment they are added together, hence the term "additive composition". This way of composition includes the traditional bottom-up composition.

This applies at the moment the platform parts are added together, hence the term "additive composition". This way of composition includes the traditional bottom-up composition common in smart card evaluations. Checking at the moment of addition also allows for composition of platform parts in the form of source code, as the addition moment is the moment where the objectives for the environment ("ensure that the hardware fulfills the requirements of the software platform part") are checked.

At the time of composition, the developer of the composed platform must show that all identified objectives for the environment are either listed for the platform, or fulfilled by one of the platform parts. The developer should show that the guidance of all platform parts has been fulfilled. The evaluator must verify that all objectives for the environment of the platform parts are fulfilled or clearly identified for the user in the ST. The evaluator must verify that all other guidance has been fulfilled or that not fulfilling that guidance does not lead to vulnerabilities.

Composition may occur between platform parts that are evaluated at different assurance levels. By default, the composed platform can claim at most the lowest assurance level of the platform parts it is composed of. For instance, a platform composed of a SESIP2 part and a SESIP5 platform part, cannot be certified at a level higher than SESIP2 without providing additional evidence about the SESIP2 part.

Nevertheless, in such a case, the Security Target may identify a subset of the SFRs, requiring higher assurance. In that case, the Security Target can claim the high-assurance part provided the SFRs covered by the higher part are clearly identified to the reader. Using the same example as above, the platform could claim being at SESIP2 with a SESIP5 part covering the SFRs of the SEPIP5 part.

In exceptional cases, a higher-SESIP composed platform encapsulates the lower-SESIP platform part in a way that protects the lower-SESIP platform part from higher-SESIP attackers, and protects the higher-SESIP composed platform from higher-SESIP attackers in the lower-SESIP platform part. If this is sufficiently shown, a fully authorized SESIP CB may in such exceptional cases assign a higher SESIP level to the platform (at most the SESIP level of the highest-SESIP part). More than 1 SESIP level difference shall require exceptionally strong assurance.

2 Security Functional Requirements

IoT platforms need to provide developers and evaluators of IoT applications with functionality to build secure products, and allow efficient verification of the accurate use of these secure platforms by users, developers and evaluators/certifiers.

The SFRs are described with the wording of the requirement, the “value”, and the “considerations”.

The “value” section describes what is added in value by a platform providing this functionality, to the users, evaluators and developers of composite IoT products and platforms.

The “considerations” section describes what aspects should be considered in the evaluation and certification of this security functional requirement.

The “Traditional CC” section describes how the optimised CC encoding in SESIP and the encoding in traditional CC are related, allowing translation between the two.

Only the Security Functional Requirements listed in this chapter shall be used in Security Targets. Should you have a functionality that cannot be captured in the below Security Functional Requirements, contact the owner of the SESIP Standard or the appropriate workgroup. To maintain the single-source simplicity and recognition of SESIP, only fully accredited SESIP CBs may extend the SFRs beyond the currently published version of SESIP.

Note that all Security Targets should include the “Identification of platform type” SFR, and all should either include the “Secure update of platform” SFR or argue under ALC_FLR.2 why updates are not applicable.

SFR identification is the subsection title, e.g. “Identification of platform type”, not the number before it (as this will change when SESIP changes).

2.1 Identification and attestation of platforms and applications

Identification and attestation allow customers, developers and evaluators to verify they have the evaluated product. More complex attestation allows for a wider scope of what is attested, and higher assurance on that which is attested. The SFRs in this subsection are to be used to express identification and attestation requirements.

2.1.1 Identification of platform type

The platform provides a unique identification of the platform type, including all its parts and their versions.

Value

Users can only verify they have a secure product, if they can obtain the identifications of the product parts (application and platform). The platform is a crucial building block of that identification process.

Evaluators and developers of composites need to be able to verify the identity too (but might require attestation for higher assurance).

Considerations

The ST describes the platform (=TOE) scope, thereby defining all its parts.

Developers of platform (parts) are required to keep the identification (globally) unique: all products from that developer that identify in the way the certified product is identified, must be the certified product.

This functional requirement is mandatory for all platforms, to ensure customers can verify that they have the correct certified platform. This includes any other composition of platform parts.

Traditional CC

Traditional CC has the requirement for identification of the TOE distributed over ASE_INT, AGD_PRE, and the identification tasks in ALC_CMC, ATE and AVA. SESIP centralises this to a function in the platform to enable automated checking of the platform identification, allowing for more efficient compositions checks and future compliance checks.

2.1.2 Identification of individual platform

The platform provides a unique identification of that specific instantiation of the platform, including all its parts and their versions.

Value

Developers of composites may need a unique identifier for a specific instantiation of a platform.

Considerations

Developers of platform (parts) are required to keep the individual instantiation identification (globally) unique in combination with the “Identification of platform type”.

Traditional CC

Traditional CC does not have an equivalent security requirement. [PP-0084] has the SFR FAU_SAS.1 with an open operation allowing encoding as such:

FAU_SAS.1 Audit storage

FAU_SAS.1 The TSF shall provide the test process before TOE Delivery with the capability to store [selection: the Initialisation Data, Pre-personalisation Data] , *and an unique identification of the specific instantiation of the TOE* in the [assignment: type of persistent memory].

2.1.3 Genuine platform instantiation

The platform provides an attestation of the “Identification of platform type” and “Identification of individual platform”, in a way that cannot be cloned or changed without detection.

Value

Users, developers and evaluators can verify that they have the genuine (secure) product, not an insecure/incomplete clone. This gives more assurance to those parties and makes the genuine secure product more distinctive and valuable.

Users of the platform certificates issued against this standard, such as dedicated schemes for products, can use this attestation as their way of identifying the genuine product.

Considerations

The genuine identification is used to ensure a given platform is genuine instantiation of the platform type and not a device posturing as one (clone). Hence the mechanism and its keys are valuable to an attacker seeking to clone the platform or otherwise misuse the developer’s brand and reputation. As the developer is potentially held accountable for clones identifying as the genuine platform, or at least suffers reputation damage for them, such an attacker goal must be considered as part of the evaluation.

Traditional CC

Traditional CC does not have a singular way to encode this. Commonly this is a challenge-response protocol encoded with FCS_COP.

2.1.4 Secure initialization of platform

The platform ensures its authenticity and integrity during the platform initialization. If the platform authenticity or integrity cannot be ensured, the platform will go to a secure state.

Value

Users, developers and evaluators can trust that the platform verified its authenticity and integrity at start-up, hence an operational product is running on a secure platform.

Considerations

A platform detecting a breach of authenticity or integrity may offer “Factory reset of platform”, “Secure update of platform”, or “Decommission of platform” functionality to recover a given product.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. The notion of secure initialisation is considered under ADV_ARC.1. However, that is not prescriptive about ensuring authenticity and integrity during the platform initialization, leaving the definition of what is considered to be *secure initialization* up to the developer to define. SESIP defines explicitly that the authenticity and integrity is checked.

2.1.5 Attested secure state of platform

The platform provides an attestation of the state of the platform, such that it can be determined that the platform is in a secure state.

Value

Users, evaluators and developers of composites can verify that the platform is still in the evaluated secure state.

Consideration

Implies that “Genuine platform” and “Secure initialization of platform” are also implemented.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual checks as described in guidance documentation and considered under AGD_PRE.1, and possibly also AGD_OPE.1.

2.1.6 Genuine application

The platform provides an attestation of the application, in a way that cannot be cloned or changed without detection.

Value

Users can determine that they have the genuine (secure) product by verifying the application, not an insecure/incomplete clone. This gives more assurance to the user and makes the genuine (secure) product more distinctive and valuable.

Considerations

Implies that "Genuine platform" and "Secure initialization of platform" are also implemented.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual checks as described in guidance documentation and considered under AGD_PRE.1, and possibly also AGD_OPE.1.

2.1.7 Attested state of application

The platform provides an attestation of state of the application.

Value

Users, evaluators and developers of composites can verify that the application is in a specific state. This specific state could then be compared to the evaluated secure state.

Considerations

The attested state of the application is here limited to information that is available to the platform, such as the application's static code and configuration, and other platform-managed information, *e.g.*, an application lifecycle state.

Implies that "Genuine application" and "Attested secure state of platform" are also implemented.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual checks as described in guidance documentation and considered under AGD_PRE.1, and possibly also AGD_OPE.1.

2.2 Product lifecycle: Factory reset / Install / Update / Decommission

The platform must always maintain the security functional requirements claimed, including the boot and shutdown stages.

The SFRs in this subsection are to be used to express other common life cycle steps such as secure installation, update and decommission of the platform and application.

The confidentiality of the platform or application may be important, for example to protect intellectual property rights or because this confidentiality has been assumed during a vulnerability analysis. As these life cycle steps may happen in the field, confidentiality needs to be maintained.

2.2.1 Factory reset of platform

The platform can be reset to the state in which it exists when the composite product embedding the platform is delivered to the user, before any personal user data, user credentials, or user configuration is present on the platform.

Value

The user invokes this functionality prior to disposing of the product instance or otherwise potentially allowing an attacker physical access to the product instance, such as reselling it. The user's data is guaranteed to be destroyed, independently of the application running on the platform.

Considerations

The platform must still be functional after the factory reset, allowing the user to initialize the composite product embedding the platform.

This reset must destroy all data (including the data of the application) received after user delivery, such that none of this data can be compromised even when the product is also physically accessible to an attacker.

This functionality still allows storage of platform instance unique data, such as data needed for "Genuine platform instantiation" and "Attested secure state of platform", allowing the platform and product to be operational still.

This functionality is not intended to counter attacks where an attacker has temporary physical access and then returns it to the user, those are countered by "Physical attacker resistance".

If the application is (partially) defective, it must still be possible to factory reset the product and then throw the product away, without the (user) data being recoverable in the product still. Platforms with functionality from "Secure encrypted storage", "

Residual information purging", or "Cryptographic keystore" typically will be able to fulfil this requirement easier.

The destruction method must be appropriate for the persistent memory technology and attack potential. See also "

Residual information purging".

Traditional CC

Traditional CC has a requirement for residual information protection (FDP_RIP.1) which is intended to provide functionality to ensure content of a resource is made unavailable either upon allocation or deallocation of the resource to an object. However, that does not specifically address the scenario where an action is invoked (factory reset) by the user nor the enforcement of active data destruction.

2.2.2 Secure install of application

The application can be installed in the field such that the integrity, authenticity *<and confidentiality>* of the application is maintained.

Value

An application may be installed in the field, including at the final end-user and at unsecured product production sites. The composite developers and evaluators can be ensured that the application is not modified (or optionally: disclosed) during this installation.

Consideration

The installation mechanism must ensure that an application is compatible with the underlying platform in its current version before installing the application.

A platform offering this SFR should consider also offering "Secure update of application" as a complement to the "Secure update of platform", allowing the product vendor to provide flaw remediation procedures that cover the entire product.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual configuration of the application in accordance to Preparative Guidance, as considered under AGD_PRE.1.

2.2.3 Secure update of platform

The platform can be updated to a newer version in the field such that the integrity, authenticity and confidentiality of the platform is maintained.

Value

Addressing security flaws, functional bugs or improvements, may require an update of the platform in the field. The update mechanism should not in itself enable an attack.

Considerations

Note that absence of this functionality must be explained in the ST under ALC_FLR.2.

The update mechanism must counter against downgrade attacks ("updating" to an older, potentially more vulnerable version). What is an "older" or "newer" version is defined in ALC_FLR.2.

Note also that updates of the platform must have a (globally) unique identifier as per "Identification of platform type", and may require their own (re-)evaluation and (re-)certification.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. The notion of secure update is typically considered across a number of security functional requirements, such as trusted channel, FTP_ITC.1, together with the ALC_FLR assurance requirements for timely updates (although that does not address the updating of a deployed instance of the TOE).

A number of collaborative Protection Profiles (e.g. Network Device cPP v2.1) include extended components, such as FTP_TUD_EXT.1 to specify requirements for administrator initiated update, with open operations:

FPT_TUD_EXT.1.1 The TSF shall provide Security Administrators the ability to query the currently executing version of the TOE firmware/software and [selection: the most recently installed version of the TOE firmware/software; no other TOE firmware/software version].

FPT_TUD_EXT.1.2 The TSF shall provide Security Administrators the ability to manually initiate updates to TOE firmware/software and [selection: support automatic checking for updates, support automatic updates, no other update mechanism].

FPT_TUD_EXT.1.3 The TSF shall provide means to authenticate firmware/software updates to the TOE using a [selection: digital signature mechanism, published hash] prior to installing those updates.

2.2.4 Secure update of application

The application can be updated to a newer version in the field such that the integrity, authenticity and confidentiality of the application is maintained.

Value

Addressed security flaws, functional bugs or improvements may require an update of the application in the field. The composite developers and evaluators can be ensured that the application is not modified or disclosed during this update.

Consideration

The update mechanism must counter against downgrade attacks ("updating" to an older, potentially more vulnerable version). What is an "older" or "newer" version is defined in ALC_FLR.2.

The update mechanism must ensure that the new version of the application is compatible with the underlying platform in its current version before updating the application.

A platform offering this may consider also offering "Secure install of application".

Traditional CC

As stated in "Secure update of platform", traditional CC does not have an equivalent security functional requirement to encode this. The notion of secure update is typically considered across a number of security functional requirements (such as trusted channel, FTP_ITC.1, and extended components, such as FTP_TUD_EXT.1 as used in a number of collaborative Protection Profiles) together with the ALC_FLR assurance requirements for timely updates (although that does not address the updating of a deployed instance of the TOE).

2.2.5 Decommission of platform

The platform can be decommissioned.

Value

The user invokes this functionality prior to disposing of the product instance or otherwise potentially allowing an attacker physical access to the product instance. As all data is destroyed, the user's data is also destroyed.

Considerations

The platform must not be functional after the decommissioning.

The decommissioning must destroy all data (including the data of the application) received after production. An application developer may mark the application object as exempt from destruction during decommissioning (such that the application code is available even after decommissioning has been performed. However, decommissioning must destroy all application parts not explicitly marked as exempt of decommissioning. Destruction must be such that the data and application parts cannot be compromised even when the product is also physically accessible to an attacker.

After destroying all data, the platform may offer a limited diagnostic mode for post-failure analysis.

If the application is (partially) defective, it must still be possible to decommission the product and then throw the product away, without the (user) data being recoverable in the product still. Platforms with functionality from "Secure encrypted storage", "Residual information purging", or "Cryptographic keystore" typically will be able to fulfil this requirement easier.

The destruction method must be appropriate for the persistent memory technology and attack potential. See also "Residual information purging".

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual checks as described in guidance documentation and considered under AGD_OPE.1.

2.2.6 Field return of platform

The platform can be returned to the vendor without user data.

Value

The user invokes this functionality prior to returning of the product instance or otherwise potentially allowing an attacker physical access to the product instance. As the user's data is destroyed, the user can be assured that not even the vendor can access personal information.

Considerations

The field return must destroy all data (including the data of the application) received after user delivery, such that none of this data can be compromised even when the product is also physically accessible to an attacker or the vendor.

This functionality still allows storage of platform instance unique data, such as data needed for "Genuine platform instantiation" and "Attested secure state of platform", allowing the platform and product to be operational still.

This functionality is not intended to counter attacks where an attacker has temporary physical access and then returns it to the user, those are countered by "Physical attacker resistance".

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Commonly this is left as an item to be addressed through manual checks as described in guidance documentation and considered under AGD_OPE.1.

2.3 Secure communication

Typically a product will use secure channels to communicate with a server, another IoT device or a cloud service. These SFRs can be used to describe the platform providing such functionality to the application.

Note that the secure communication SFRs should be used when the platform provides the secure channels, the "Cryptographic operation" SFRs can be used if the platform provides

2.3.1 Secure communication support

The platform provides the application with one or more secure communication channel(s).

The secure communication channel authenticates <*list of endpoints*> and protects against <*list of attacks including disclosure, modification, replay, erasure*> of messages between the endpoints, using <*list of protocols and measures*>.

Value

The composite product developer can use the secure communication channels.

The composite evaluator/certifier knows that if the product developer uses the functionality, it is secure in the above manner.

The user has to rely on the composite product developer and composite certifier to know if the secure channels are actually used.

Considerations

The variable parts of this SFR should be completed as follows:

- *list of endpoints*: the list of endpoints to be protected using the protocols and measures listed hereafter. Endpoints can be identified by their quality, by the interface to which they are connected, or by a more general category, *e.g.*, all local endpoints. If different protocols and measures are used for different endpoints, defining different secure communication channels, then the SFR must be iterated.
- *list of attacks*: the list of security issues to avoid, typically including disclosure, modification, replay and erasure.
- *list of protocols and measures*: the list of the protocols and other measures used to protect the communication channel, e.g., TLS 1.2 with TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, or IPSEC with … . It is important to provide enough information to properly characterize the quality of the algorithms used.

If multiple different secure communication channels are provided, this SFR should be iterated.

This SFR could be sensitive to misleading claims, the fully authorized SESIP CB and evaluators shall ensure this is not misleading.

Traditional CC

Traditional CC has multiple ways to encode communication channels, depending on what type of entity the TOE is communicating with (other portions of the TOE, human, trusted IT entity, etc). These methods include FPT_ITT Internal TOE TSF data transfer, FTP_ITC Inter-TSF Trust Channel, FTP_TRP Trusted Path, and are sometimes supported by specification of algorithms and key sizes with FCS_COP.

2.3.2 Secure communication enforcement

The platform ensures the application can only communicate with <*list of endpoints*> over the secure communication channel(s) supported by the platform.

Value

The user and composite evaluator/certifier can trust the product is securely communicating (typically with cloud services), if the platform is attested in the secure state.

Considerations

This SFR requires that one or more iterations of the "Secure communication support" SFR is claimed. The variable part of this SFR, *list of endpoints*, identifies the secure channels to which the SFR applies, as they have been defined in the corresponding "Secure communication support" SFRs.

This SFR implies that the platform does all the unsecured communication layers under the secure communication channel (TCP/IP, DHCP, DNS, BT, …), and that the application does not have direct access to the communication layers underlying that secure channel.

This SFR could be sensitive to misleading claims, the fully authorized SESIP CB and evaluators shall ensure this is not misleading.

Traditional CC

Traditional CC uses FPT_ITT Internal TOE TSF data transfer to encode communication channels with local endpoints and FTP_ITC Inter-TSF Trust Channel for communication channels with remote endpoints.

2.4 Extra attacker resistance

In our regular attacker model, attackers have only logical (network) access to the product, application and platform during the exploitation phase of the attack, and therefore:

- do not have physical access to the specific product instance attacked, and
- are unable to run their own hostile code on the platform[1].

When this attacker model is not valid, adding security functional requirements from this section allows expression of resistance against physical and software attackers.

Note that during the identification phase, the attacker is assumed to have physical access to his platform instance when preparing his attack.

2.4.1 Limited physical attacker resistance

The platform detects or prevents attacks by an attacker with physical access before the attacker compromises <*list of security functional requirements*>.

Value

For situations where a physical attacker is in scope for a limited set of functionality such as valuable assets warranting a physical attack, but not for all functionality. The attacker is not limited in his physical attack, the attacker is limited in the target of his attack (the limited set of SFRs).

Considerations

All attacks enabled by physical access within the attack potential need to be considered.

With this SFR, local non-networked interfaces such as an USB port and MicroSD card reader must now also be considered in the vulnerability analysis as accessible to the attacker (such as an "evil maid").

Attackers (mis-)using production and debug functionality such as JTAG and ISD functionality would typically be countered by disabling this functionality prior to delivery to the customer. Invasive attacks such as physical tampering or perturbation would typically be countered by detection and decommissioning the device before the detected attack succeeds. Non-invasive attacks such as side channel attacks would typically be countered by not leaking secret information via side channels such as timing, power and EM emissions.

The threat of replacement of the product is not covered by this SFR; it can be countered with SFRs "Identification of individual platform" together with "Genuine platform instantiation", allowing an application or user to detect replacement.

Traditional CC

Traditional CC has similar security requirements in the FPT_PHP family, which can be used to specify detection (FPT_PHP.1), notification (FPT_PHP.2) and response (FPT_PHP.3) to physical tampering. Only the traditional FPT_PHP.3 requirement has the capability to limit what attacks to which the TOE should respond.

2.4.2 Physical attacker resistance

The platform detects or prevents attacks by an attacker with physical access before the attacker compromises any of the other functional requirements, ensuring that the other functional requirements are not compromised.

Value

[1] That is: an attacker would have to "break" the application or platform first in order to install his code: there are no regular ways to do this.

For situations where a physical attacker is in scope, such as products that are typically used outside a secured area, or when the products store highly valuable assets warranting a physical attack.

Considerations

All attacks enabled by physical access within the attack potential need to be considered.

With this SFR, local non-networked interfaces such as an USB port and MicroSD card reader must now also be considered in the vulnerability analysis as accessible to the attacker (such as an "evil maid").

Attackers (mis-)using production and debug functionality such as JTAG and ISD functionality would typically be countered by disabling this functionality prior to delivery to the customer. Invasive attacks such as physical tampering or perturbation would typically be countered by detection and decommissioning the device before the detected attack succeeds. Non-invasive attacks such as side channel attacks would typically be countered by not leaking secret information via side channels such as timing, power and EM emissions.

The threat of replacement of the product is not covered by this SFR; it can be countered with SFRs "Identification of individual platform" together with "Genuine platform instantiation", allowing an application or user to detect replacement.

Traditional CC

Traditional CC has similar security requirements in the FPT_PHP family, which can be used to specify detection (FPT_PHP.1), notification (FPT_PHP.2) and response (FPT_PHP.3) to physical tampering.

2.4.3 Software attacker resistance: isolation of platform

The platform provides isolation between the application and itself, such that an attacker able to run code as an application on the platform cannot compromise the other functional requirements.

Value

For situations where an attacker may be able to load his own code on the platform, or the attacker might subvert any part of the application.

Considerations

This typically would require at least an OS with kernel-user mode separation.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this, although the notion of domain separation is considered under ADV_ARC.1.

2.4.4 Software attacker resistance: isolation of application parts

The platform provides isolation between parts of the application, such that an attacker able to run code as one of the <*list of application parts*> cannot compromise the integrity and confidentiality of the other application parts.

Value

For situations where the product developer wants to separate the critical assets in its own application part (process/executable/…), and thus safeguard them from compromises by other parts of the application code that are too complex to be shown to be secure.

Considerations

The variable part, *list of application parts*, of this SFR must list the part types that may compose the application (modules, processes, applets), and between which some isolation is to be provided.

This typically would require an OS with application-application memory separation or an interpreter-based platform with similar access rules. Java Card for example fulfils this.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Although the notion of domain separation is considered under ADV_ARC.1, the isolation between parts of application is not specifically addressed in the traditional CC.

2.4.5 Software attacker resistance: isolation of platform parts

The platform provides isolation between platform parts, such that an attacker able to run code in <*list of vulnerable platform parts*> can compromise neither the integrity and confidentiality of <*list of protected platform parts*> nor the provision of any other security functional requirements.

Value

For situations where an attacker may be able to load his own code on inner parts of the platform.

The platform developer can separate the critical assets in different parts of the platform, and thus safeguard them from compromises of other parts of the platform.

Considerations

The variable parts of this SFR need to be completed:

- The *list of vulnerable platform parts* defines the parts of the platform that are considered to be potentially compromised by an attacker and used to attack other parts of the platform.
- The *list of protected platform parts* defines the parts of the platform that are considered to be potentially targeted by attackers running code in other compromised parts, and that are protected against such attacks.

A platform offering this functionality would typically also claim “Software attacker resistance: isolation of platform”.

This would typically require a micro-kernel or PSA/TrustZone-type technology.

Traditional CC

Traditional CC does not have an equivalent security functional requirement to encode this. Although the notion of domain separation is considered under ADV_ARC.1, the isolation between parts of application is not specifically addressed in the traditional CC.

2.5 Cryptographic functionality

These are common cryptographic functions that a platform can provide that are useful to a product developer, but typically not directly visible to the product user.

Standard CC interpretation applies for references to standards: all of the claimed parts of the specification need to be fully implemented, so precise references are encouraged. Evaluators and certifiers must verify all aspects of the parts of standards referenced.

2.5.1 Cryptographic operation

The platform provides the application with <*list of cryptographic operations*> functionality with <*list of algorithms*> as specified in <*specification*> for key lengths <*list of key lengths*> and modes <*list of modes*>.

Value

Evaluators and developers of composites can be ensured of standard-compliant cryptographic functions.

Considerations

The variable parts in the SFR should be completed as follows:

- The *list of cryptographic operations* defines the operations that can be considered, typically selected among encryption, decryption, hashing, signing, signature verification.
- The *list of algorithms* provides a reference to the cryptographic algorithms used.
- The *specification* references the standard in which the operation is defined (including a section or similar information if relevant).
- The *list of key lengths* defines the key lengths that are supported for that cryptographic operation.
- The *list of modes* defines the operational modes that are supported for that cryptographic operation.

A typical example of a fully defined SFR would be: The platform provides the application with encryption and decryption functionality as specified in NIST FIPS 197 (AES) with 256-bit keys in GCM mode.

This SFR should be iterated if more than one algorithm is provided. For the sake of clarity, it may be preferable to use a table to define the supported parameters:

Operations	**Algorithm**	**Specification**	**Key lengths**	**Modes**
Encryption, decryption	AES	NIST FIPS 197	256	GCM

The cryptographic operation must keep the confidentiality of the secret keys against the attacker. Even with the standard attacker model, timing and padding oracle attacks must be considered if within the attack potential.

Note that this SFR is for cryptographic functionality available to the application. This SFR should not be used to only restate the crypto functionality claimed in other SFRs (such as “Secure encrypted storage”, “Secure communication support”, “Secure communication enforcement”) unless that functionality is separately made available to the application.

Traditional CC

Traditional CC has the requirement FCS_COP.1 for the specification of cryptographic operations.

2.5.2 Cryptographic key generation

The platform provides the application with a way to generate cryptographic keys for use in <*list of cryptographic algorithms*> as specified in <*specification*> for key lengths <*list of key lengths*>.

Value

Evaluators and developers of composites can be ensured of standard-compliant cryptographic key generation.

Considerations

The variable parts in the SFR should be completed as follows:

- The *list of algorithms* provides a reference to the algorithms used
- The *specification* references the standard in which the operation is defined (including a section or similar information if relevant).
- The *list of key lengths* defines the key lengths that are supported for that cryptographic operation.

This SFR should be iterated if keys are generated for more than one algorithm. For the sake of clarity, it may be preferable to use a table to define the supported parameters.

The specification of the key generation algorithm can be useful for use in specific product evaluation schemes. Regardless of the algorithm used, every attack within the attack potential applies: attacks such as ROCA are always to be checked against.

Traditional CC

Traditional CC has the requirement FCS_CKM.2 for the specification of cryptographic key generation, with FCS_COP.3 for the specification of cryptographic key import.

2.5.3 Cryptographic keystore

The platform provides the application with a way to store <*list of assets, such as cryptographic keys and passwords*> such that not even the application can compromise the <*authenticity, integrity, confidentiality*> of this data. This data can be used for the cryptographic operations <*list of operations*>.

Value

Evaluators and developers of composites can be ensured that keys cannot be disclosed accidentally, provided that the keys are stored only in the keystore.

Considerations

The variable parts of the SFR should be completed as follows:

- The *list of assets* defines the types of assets to be protected by the keystore.
- The properties *authenticity, integrity, confidentiality* defines the protection afforded.
- The *list of operations* defines the operations that an application can perform on the assets stored in the keystore without having to access the assets' values.

A software keystore in the platform typically will require either "Software attacker resistance: isolation of platform" in the platform or code review or automated code verification of the product.

Traditional CC

Traditional CC does not have a singular way in which the secure key storage is specified. Typically, it is defined through a combination of user data protection (access control policy FDP_ACC.1 and access control functions FDP_ACF.1) together with requirement FCS_COP.1 for cryptographic operations.

2.5.4 Cryptographic random number generation

The platform provides the application with a way based on <*list of entropy sources* > to generate random numbers to as specified in <*specification*>.

Value

Evaluators and developers of composites can be ensured of standard-compliant cryptographic key generation.

Considerations

The variable parts of the SFR should be completed as follows:

- The *list of entropy sources* defines how the randomness is generated from physical and computational resources.
- The *specification* references the standard in which the operation is defined (including a section or similar information if relevant).

The specification of the number generation algorithm can be useful for use in specific product evaluation schemes. Regardless of the algorithm used, every attack within the attack potential applies: weak entropy and predictability must always be checked against.

Traditional CC

Traditional CC does not have an equivalent security requirement. [PP-0084] has the SFR FCS_RNG.1 with open operations allowing encoding as such:

FCS_RNG.1.1 The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic] random number generator that implements: [assignment: list of security capabilities].

FCS_RNG.1.2 The TSF shall provide [selection: bits, octets of bits, numbers [assignment: format of the numbers]] that meet [assignment: a defined quality metric].

2.6 Compliance functionality

These are commonly required properties from various product domains and schemes.

These features are typically not easily visible to either the application developer or the product user. Features that are typically visible, such as "Secure communication", are listed elsewhere.

2.6.1 Secure encrypted storage

The platform ensures that all data stored by the application, with the exception of <*list of data stored in plaintext*>, is encrypted as specified in <*specification*> with a platform instance unique key of key length <*key length*>

Value

Evaluators and developers of composites can be ensured that (implicitly) stored data is encrypted, without further activities.

Considerations

The variable parts of the SFR should be completed as follows:

- The *list of data stored as plaintext* lists the categories that are excluded from the secure encrypted storage, which should be used for all other data.
- The *specification* references the standard in which the encryption mechanism is defined (including a section or similar information if relevant).
- The *key length* is the size of the key used for the encryption.

The encryption mechanism used must protect both the integrity and confidentiality of stored data.

The key used to encrypt the data must be generated such that compromise of the key of one product-instance does not allow easier compromise of another product-instance.

This may be implemented using the "Cryptographic operation" functionality, but that SFR should only be claimed if the functionality of that SFR is directly available to the application.

Note that if the storage is not encrypted but otherwise protected against read out by physical attackers, "Physical attacker resistance" should be used.

Traditional CC

Traditional CC does not have a singular way in which the secure encrypted storage is specified. Typically, it is defined through a combination of user data protection (access control policy FDP_ACC.1 and access control functions FDP_ACF.1) together with requirement FCS_COP.1 for cryptographic operations.

2.6.2 Residual information purging

The platform ensures that <*list of data*>, with the exception of <*list of exceptions of data that is not erased automatically*>, is erased using the method specified in <*specification*> before the memory is (re)used by the platform or application again and before an attacker can access it.

Value

Evaluators and developers of composites can be ensured that (implicitly) stored and copied data is erased automatically, without further activities.

Considerations

The variable parts of the SFR should be completed as follows:

- The *list of data* defines the categories of data that are to be cleared automatically, which is expected to be very wide (see below).
- The *list of exceptions* defines the categories that are not expected to be cleared automatically.
- The *specification* references the standard in which the encryption mechanism is defined (including a section or similar information if relevant)

Typically, *list of data* would be "all data of the application no longer used by the application", allowing for lazy erasure happening only at re-use of the memory, not immediately at release of the memory to match common software implementations.

Attacks such as coldboot need to be considered.

Traditional CC

Traditional CC has a similar SFR of FDP_RIP for residual information protection.

2.6.3 Audit log generation and storage

The platform generates and maintains an audit log of <*list of significant security events*> and allows access and analysis of these logs following a specific <*access control policy*>.

Value

Evaluators, developers and users of composites can detect attack attempts to the platform.

Considerations

The variable parts of the SFR should be completed as follows:

- The *list of significant security events* defines the events in the log, which are expected to provide a wide coverage of the platform's possible events.
- The *access control policy* defines the conditions under which the logs may be inspected, typically by specific privileged users after authentication.

"Software attacker resistance: isolation of platform" might also be claimed to support this claim.

Traditional CC

Traditional CC has a similar SFR of FAU_GEN.1 for generation of audit logs.

2.6.4 Reliable time

The platform provides the application with a measure of time with an accuracy of <*resolution and maximum drift*>.

Value

Evaluators and developers of composites can be ensured that there is a time source, without further activities.

Considerations

-

Traditional CC

Traditional CC has a similar SFR of FPT_STM.1 for provision of reliable time stamp for use by the TOE.

2.6.5 Secure debugging

The platform only provides <*list of endpoints*> authenticated as specified in <*specification*> with debug functionality.

The platform ensures that all data stored by the application, with the exception of <*exceptions*>, is made unavailable.

Value

Developers of composites can debug their applications without compromising the security of their users' data.

Considerations

The variable parts of the SFR can be completed as follows:

- The *list of endpoints* lists the endpoints (potentially with their physical connection and the authentication data needed) allowed to set the platform in debug mode.
- The *specification* references the standard in which the authentication mechanism is defined (including a section or similar information if relevant).
- The *exceptions* lists the data that is excluded from the protection during debugging. This should not include application data that has a reasonable expectation of containing the user's personally identifiable information.

This may be implemented using "Secure communication support" functionality.

Traditional CC

Traditional CC does not have a singular way in which the secure debug is specified. Typically, it is partly defined through a combination of FPT_ITT Internal TOE TSF data transfer, FTP_ITC Inter-TSF Trust Channel, FTP_TRP Trusted Path, and are sometimes supported by specification of algorithms and key sizes with FCS_COP.

3 Security Assurance Requirements

This document contains five hierarchical sets of Common Criteria assurance packages that are suitable to evaluate IoT platforms or parts thereof.

The sets are named **SESIP1, SESIP1+, SESIP3, SESIP4 and SESIP5** and are hierarchical:

SESIP1	**SESIP Assurance Level 1 (SESIP1)** is a self-assessment-based level: the developer has to provide a simplified Security Target, describing the security claims of his product, together with a rationale why he believes these claims are met. Only minimal evaluator effort is needed: checks on consistency and clarity of the self-assessments are performed. There is no independent check by the evaluators the platform actually implements the SFRs. SESIP1 provides a *basic level* of assurance.
SESIP2	**SESIP Assurance Level 2 (SESIP2)** is a black-box penetration testing level: the evaluation is structured around a time-limited penetration testing effort. No design or source code is required to be available besides a full functional specification. This is the highest level that can be applied to a closed-source platform without cooperation by the developer. SESIP2 provides a *moderate level* of assurance.
SESIP3	**SESIP Assurance Level 3 (SESIP3)** is a traditional white-box vulnerability analysis: the evaluation is structured around a time-limited source code analysis combined with a time-limited penetration testing effort. Other assurance components have only been included to support this approach to save as much effort as possible. SESIP3 provides a *substantial level* of assurance.
SESIP4	**SESIP Assurance Level 4 (SESIP4)** is an extended white-box vulnerability analysis with secure development process: the evaluation is structured to show the resistance against the attackers based on thorough source code analysis combined with a verification that the development process supports secure software development. The attack potential is increased to be commensurate with that of AVA_VAN.4 specified in traditional CC evaluations. SESIP4 provides an *extended-substantial level* of assurance.
SESIP5	**SESIP Assurance Level 5 (SESIP5)** is the same as the traditional full CC evaluation against an EAL4+ALC_DVS.2+AVA_VAN.5 level that is used for smartcards, secure elements, e-passports etc. SESIP5 provides a *high level* of assurance. During the trial this level is exclusively for re-use of SOG-IS certified platforms by licensed labs, allowing those platforms to utilize the mappings from SESIP to specific commercial product domains immediately.

New assurance requirements (ASE_REQ.3, ADV_IMP.3) are marked in **bold** in the overview tables. Assurance requirements with a significant refinement are marked in *italic* in the overview tables. Vulnerability analysis (AVA) shall use the rating tables described in "Attack potential rating".

Note that earlier [SESIP] versions used the term "IoT Platform Assurance Level x" (or "ITPx") for the same levels or SESIP ratings SESIP1, SESIP1+, SESIP2, SESIP2+ and SESIP3. For clarity of communication, only these names have been changed to "SESIP Assurance Level x" or "SESIPx", incrementing the SESIP numerical value by one between each level. The only content change between these assurance levels is in SESIP1, which now includes a vulnerability scan by the developer (to effectively include AVA_VAN.1). The five "ITPx" labels and SESIP1, SESIP1+, SESIP2, SESIP2+ and SESIP3 labels can be mapped directly onto the SESIP1-SESIP5 levels.

3.1 SESIP Assurance Level 1 (SESIP1)

SESIP1 provides a basic level of assurance for IoT platforms. SESIP1 is based on self-assessment of the developer, with only minimal verification by an evaluator. There is no independent check by the evaluators that the platform actually implements the SFRs.

At this assurance level, the developer is expected to provide:

- A simplified Security Target with self-assessment rationale. ST templates are available on the SESIP website. The self-assessment rationale includes:
    - a (self-)check against publicly known vulnerabilities against the platform,
    - references to guidance documents describing the objectives for the environment, and
    - a reference to the public facing procedures describing how flaws are reported, tracked and corrected, and the resulting updates communicated.
- The referred (self-)check against publicly known vulnerabilities and the referred guidance documents. The public facing procedures by definition need to be available to the public (and hence the evaluators) already.

3.1.1 Objectives

SESIP1 provides assurance by using a simplified Security Target, augmented by a self-assessment rationale. The self-assessment is considered to be part of the Security Target, even if it is delivered separately (for example in the form of a completed questionnaire). The self-assessment includes testing of the TOE on the basis of a survey of public domain sources for potential vulnerabilities.

The evaluator assesses the Security Target for clarity and consistency, but existence or effectiveness of the security functionality in the actual platform is not independently assessed by the evaluator or the certifier.

3.1.2 Assurance components

Assurance Class	**Assurance Families**
ASE: Security Target evaluation	ASE_INT.1 ST Introduction *ASE_OBJ.1 Security requirements for the operational environment* **ASE_REQ.3 Listed Security requirements** *ASE_TSS.1 TOE Summary Specification*
ALC: Life-cycle support	ALC_FLR.2 Flaw reporting procedures
AVA: Vulnerability Assessment	AVA_VAN.1 Vulnerability survey

3.1.3 Security Target Requirements

ASE_INT.1 ST Introduction

As per CC part 3.

Refinement: The ST should also list the SESIP version.

ASE_OBJ.1 Security objectives for the operational environment

Dependencies: No dependencies.

Developer action elements:

ASE_OBJ.1.1D The developer shall provide a statement of security objectives.

Content and presentation elements:

ASE_OBJ.1.1C The statement of security objectives shall describe the security objectives for the operational environment.

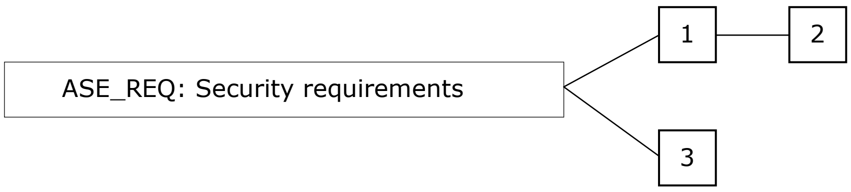
Refinement: All security objectives concerning the operational environment shall be listed with references to the place in the guidance documents where these objectives are addressed.

Evaluator action elements:

ASE_OBJ.1.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ASE_REQ.3 Listed security requirements

Amended hierarchy of components in ASE_REQ family:



Dependencies: No dependencies.

Developer action elements:

ASE_REQ.3.1D The developer shall provide a statement of security requirements.

Content and presentation elements:

ASE_REQ.3.1C The statement of security requirements shall describe the SFRs and the SARs.

ASE_REQ.3.2C All SFRs shall be drawn from the list of allowed Security Functional Requirements.

ASE_REQ.3.3C The SFR "Identification of platform type" must be included. The SFR "Secure update of platform" must be included or under the ALC_FLR.2 it must be argued why updates are not applicable.

ASE_REQ.3.4C The SARs shall be an exact SESIP assurance level. No augmentation is allowed.

ASE_REQ.3.4C If multiple SESIP assurance levels are claimed, it shall be clear to the reader of the ST what SFRs covered by what SESIP assurance level.

Evaluator action elements:

ASE_REQ.3.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Guidance:

If multiple instances of a claim for security functionality are required, the body of an SFRs may be iterated. For example, if multiple secure communication channels are supported the "Secure communication support" may be iterated as follows:

The secure communication channel:

1. authenticates **servers** and protects against **disclosure, modification** of messages between the endpoints, using **TLS 1.2 with TLS_RSA_WITH_AES_128_CBC_SHA** ciphersuite
2. authenticates **servers and clients** and protects against **disclosure, modification** of messages between the endpoints, using **SSH 1.2 with aes256-cbc using hmac-sha2-256 for SSH transport, rsa-sha2-256 SSH public-key based authentication and diffie-hellman-group14-sha256 key exchange**.

If an SFR makes references to an external standard, the reference must be precise. By including a reference to a standard, everything that could reasonably be considered part of the reference implicitly forms part of the SFR claim, and must therefore be verified. For example, a reference to FIPS-140-2 requires FIPS-140-2 certification. Whereas a reference to FIPS-140-2 section 4.9.1.does not require FIPS certification but does require power-up tests to be performed by the TOE, including cryptographic algorithm tests, software/firmware integrity test and critical functions tests.

The evaluator determines that statement of security assurance requirements matches one of the SESIP assurance levels defined in chapter 3 of this document. If the SARs are listed or copied in the ST, the evaluator determines that the set of claimed SARs exactly matches one of the SESIP assurance levels defined in SESIP (this document). It is not permitted to extend or augment a SESIP assurance level.

The evaluator determines that it is clear to the reader of the ST what SFRs are covered by what SARs. Multiple claims of SESIP assurance levels are allowed (as described in "Additive composition"), provided the claims are clear.

ASE_TSS.1 TOE summary specification

As per CC Part 3

The dependency on ASE_REQ.1 is considered to be fulfilled by ASE_REQ.3.

Refinement[2]:

The TSS shall include a self-assessment by the developer how the TOE correctly implements the Security Functional Requirements.

This self-assessment may be provided as a separate document (such as a filled-in questionnaire), however is considered part of the public ST.

The self-assessment must consider each implementation of the Security Functional Requirements, describing how the developer has supported his assessment, based, for instance on:

- *Testing, either by the developer or by a third party.*
- *Conformance to another standard by the TOE or part of the TOE.*
- *Reliance on the statements of another party on the TOE or a part of the TOE.*

This includes an assessment against the AVA_VAN.1 requirements, such as a scan for publicly known vulnerabilities.

The self-assessment must indicate what procedures cover ALC_FLR.2 if "Secure update of platform" is included in the Security Target.

[2] This was explicitly stated as the scope of TSS (ASE_TSS.1.5C) in the CC v2.3, unfortunately lost in transition to CC v3.1.

3.1.4 Life-cycle Support Requirements

ALC_FLR.2 Flaw Reporting Procedures

As per CC Part 3.

Refinement

It is recognized that many IoT platforms will require a method to update in the field to defend against new threats or against vulnerabilities that were found later on. On the other hand, there may also be IoT platforms for which this would be impractical or overkill (e.g. a very simple low-power sensor that communicates only one way).

To enable both TOE types, this assurance requirement can be met in two ways:

- *For platforms that can be updated, as per CC Part 3. Note that these platforms shall have the SFR "Secure update of platform" included in the Security Target.*
- *For platforms that cannot be updated, the Security Target shall contain a rationale why it is acceptable that this platform cannot be updated in the field. It is acceptable that a hardware root of trust cannot be updated in the field.*

In both cases, the flaw reporting procedure must describe how flaws are to be reported to the developer, and how updates to the platform or guidance (or retraction of the product as certified) are communicated to the users of the platform.

At SESIP1 and SESIP2 level, flaw reporting procedure steps not visible to the users may be omitted from the description as these are not verifiable by the evaluators. The public facing procedures must be described.

3.1.5 Vulnerability Assessment Requirements

AVA_VAN.1 Vulnerability Survey

Dependencies:

AGD_OPE.1 and AGD_PRE.1 are met by ASE_TSS.1 and the availability of the guidance documents.

ADV_FSP.1 is met by ASE_TSS.1.

Note

As per CC Part 3

Refinement

The vulnerability survey may be performed by the developer or evaluator.

The evaluator is to determine whether the testing evidence submitted by the developer considers all applicable public domain sources to identify potential vulnerabilities in the TOE. This testing may utilise appropriate penetration test tools available in the public domain to test for publicised potential vulnerabilities.

3.2 SESIP Assurance Level 2 (SESIP2)

SESIP2 is a moderate level of assurance for (parts of) IoT platforms.

SESIP2 provides significantly more assurance than SESIP1 by requiring a vulnerability analysis and actual penetration testing on the platform by an evaluator, but will provide less assurance than the higher assurance provided by white-box SESIP3.

At this assurance level, the developer is expected to provide:

- A simplified Security Target with self-assessment rationale. ST templates are available on the SESIP website. The self-assessment rationale includes:
    - a (self-)check against publicly known vulnerabilities against the platform,
    - references to guidance documents describing the objectives for the environment, and
    - a reference to the public facing procedures describing how flaws are reported, tracked and corrected, and the resulting updates communicated.
- Guidance documents. This typically consists of the existing user manuals and data sheets.
- A complete functional specification. This typically consists of existing programmer's manuals, data books or API specifications, and a mapping showing what interfaces implement the SFRs declared in the Security Target.
- Proof of functional conformance testing.

3.2.1 Objectives

SESIP2 provides assurance by an analysis of the SFRs in the simplified Security Target, using a full functional specification, guidance documentation, and the platform being tested, to understand the security behaviour.

The analysis is supported by independent testing of the platform, and a vulnerability analysis demonstrating resistance to penetration attackers with a Basic attack potential. The vulnerability analysis is based upon mapping of SFRs to interfaces and guidance evidence provided, and the description of all interfaces as provided by the functional specification.

SESIP2 also provides assurance through the assessment of the developer's procedures to produce and distribute updates to IoT Platforms in the field (ALC_FLR.2).

3.2.2 Assurance components

Assurance Class	**Assurance Families**
ASE: Security Target evaluation	ASE_INT.1 ST Introduction *ASE_OBJ.1 Security requirements for the operational environment* **ASE_REQ.3 Listed Security requirements** *ASE_TSS.1 TOE Summary Specification*
ADV: Development	ADV_FSP.4 Complete functional specification
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_FLR.2 Flaw reporting procedures

ATE: Tests	ATE_IND.1 Independent testing: conformance
AVA: Vulnerability Assessment	AVA_VAN.2 Vulnerability analysis

3.2.3 Security Target Requirements

As per section 3.1.3.

3.2.4 Development Requirements

ADV_FSP.4 Complete functional specification

Dependencies:

ADV_TDS.1 Basic design is considered to be sufficiently fulfilled by *ASE_TSS.1 for the black-box platform.*

For the rest as per CC Part 3.

3.2.5 Guidance Documents Requirements

AGD_OPE.1 Operational User Guidance

As per CC Part 3.

AGD_PRE.1 Preparative Procedures

As per CC Part 3.

3.2.6 Life-cycle Support Requirements

ALC_FLR.2 Flaw Reporting Procedures

As per section 3.1.4.

3.2.7 Tests Requirements

ATE_IND.1 Independent testing: conformance

As per CC Part 3.

Note

Re-use of industry standard testing by the developer or third parties is encouraged, provided this is accepted by the CB.

3.2.8 Vulnerability Analysis Requirements

AVA_VAN.2 Vulnerability analysis

Dependencies:

AGD_OPE.1 and AGD_PRE.1 are met.

ADV_ARC.1 and ADV_TDS.1 are considered to be met by ASE_TSS.1 in combination with ADV_FSP.4.

ADV_FSP.2 is met by ADV_FSP.4.

Note

The effort spent on AVA_VAN.2 (vulnerability analysis and penetration testing) is expected to be equivalent to 20 man day for a typical platform to reach Basic attack potential.

Definition of the "equivalent effort" and "typical platform" is at the discretion of the implementing SESIP scheme in alignment with the SESIP community.

Note that in case of a composed platform, the evaluator must verify that all objectives for the environment of the platform parts fulfilled by another platform part are accurately fulfilled. All guidance of one platform part for another platform part should be considered as part of the vulnerability analysis.

3.3 SESIP Assurance Level 3 (SESIP3)

SESIP3 is a substantial level of assurance for (parts of) IoT platforms.

It provides significantly more assurance than SESIP2 by requiring significant source code analysis by an evaluator as input to the vulnerability analysis. Use of the source code analysis will increase the assurance gained from the vulnerability analysis and penetration testing, but will provide less assurance than the extended-substantial assurance provided by SESIP4.

At this assurance level, the developer is expected to provide:

- A simplified Security Target with self-assessment rationale. ST templates are available on the SESIP website. The self-assessment rationale includes:
  - a (self-)check against publicly known vulnerabilities against the platform,
  - references to guidance documents describing the objectives for the environment, and
  - a reference to the internal and public facing procedures describing how flaws are reported, tracked and corrected, and the resulting updates communicated.
- Guidance documents. This typically consists of the existing user manuals and data sheets.
- A complete functional specification. This typically consists of existing programmer's manuals, data books or API specifications, and a mapping showing what interfaces implement the SFRs declared in the Security Target.
- The full source code, and a mapping showing where in the source code the SFRs are implemented. This mapping typically is a minor extension of the mapping for the functional specification.
- Proof of functional conformance testing.
- A short description showing how the platform's version number is maintained to be uniquely identifying the platform in its version, and showing that the whole platform (source code and all that is described above) is maintained in a standard version management system (such as CVS, GIT or SVN).
- Internal Flaw Reporting procedures describing internal (as well as the public facing) procedures describing how flaws are reported, tracked and corrected, and the resulting updates communicated.

3.3.1 Objectives

SESIP3 provides assurance by a simplified security target, and an analysis of the SFRs in that ST, using a full functional specification, guidance documentation, and the implementation representation, to understand the security behaviour.

The analysis is supported by independent testing of the TSF, and a vulnerability analysis (based upon the functional specification, implementation representation and guidance evidence provided) demonstrating resistance to penetration attackers with an Enhanced-Basic attack potential.

SESIP3 also provides assurance through the use of development environment controls and additional TOE configuration management including automation, and evidence of secure delivery procedures.

3.3.2 Assurance components

Assurance Class	**Assurance Families**
ASE: Security Target evaluation	ASE_INT.1 ST Introduction *ASE_OBJ.1 Security requirements for the*

	operational environment **ASE_REQ.3 Listed Security requirements** *ASE_TSS.1 TOE Summary Specification*
ADV: Development	ADV_FSP.4 Complete functional specification **ADV_IMP.3 Complete mapping of the implementation representation of the TSF to the SFRs**
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE ALC_CMS.1 TOE CM Coverage ALC_FLR.2 Flaw reporting procedures
ATE: Tests	ATE_IND.1 Independent testing: conformance
AVA: Vulnerability Assessment	AVA_VAN.3 Focused Vulnerability analysis

3.3.3 Security Target Requirements

As per section 3.1.3.

3.3.4 Development Requirements

ADV_FSP.4 Complete functional specification

Dependencies: ADV_TDS.1 Basic design

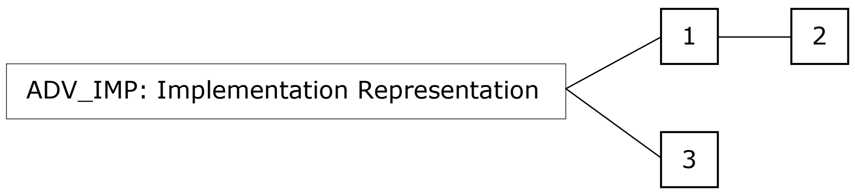
This dependency does not need to be fulfilled, as ADV_IMP.3.3D covers this.

For the rest as per CC Part 3.

ADV_IMP.3 Complete mapping of the implementation representation of the TSF to the SFRs

Dependencies: ASE_REQ.3 Listed security requirements

Amended hierarchy of components in ADV_IMP family



Developer action elements:

ADV_IMP.3.1D The developer shall make available the implementation representation for the entire TSF.

ADV_IMP.3.2D The developer shall provide a mapping between the SFRs and the entire implementation representation.

ADV_IMP.3.3D The developer shall provide further information on the structure and meaning of the implementation representation, when so required by the evaluator.

Content and presentation elements:

ADV_IMP.3.1C The implementation representation shall define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representation shall be in the form used by the development personnel.

Evaluator action elements:

ADV_IMP.3.1E The evaluator ***shall confirm*** that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator ***shall determine*** that the implementation representation is an accurate and complete instantiation of the SFRs.

See also the refinement for "Vulnerability Analysis Requirements*"*.

3.3.5 Guidance Documents Requirements

AGD_OPE.1 Operational User Guidance

As per CC Part 3.

AGD_PRE.1 Preparative Procedures

As per CC Part 3.

3.3.6 Lifecycle Support Requirements

ALC_CMC.1 Labelling of the TOE

As per CC Part 3.

ALC_CMS.1 TOE CM Coverage

As per CC Part 3.

ALC_FLR.2 Flaw Reporting Procedures

As per section 3.1.4.

Note that at SESIP3 level and higher, the entire flaw reporting procedure is to be described; not just the externally visible steps required for SESIP1 and SESIP2 levels.

3.3.7 Tests Requirements

ATE_IND.1 Independent testing: conformance

As per section 3.2.7.

3.3.8 Vulnerability Analysis Requirements

AVA_VAN.3 Focused vulnerability analysis

Dependencies:

AGD_OPE.1 and AGD_PRE.1 are met.

ADV_FSP.2 is met by ADV_FSP.4.

ADV_ARC.1 and ADV_TDS.3 are considered to be met by ADV_IMP.3 in combination with ASE_TSS.1.

ADV_IMP.1 is met by ADV_IMP.3

ATE_DPT.1 is met through a combination of ATE_IND.1 and ADV_IMP.3 informing the selection of independent test cases to ensure all major parts (subsystems) of the TOE have been tested.

Note

The effort spent on ADV_IMP.3 is expected to be equivalent of at least 5 man days for a typical platform.

The effort spent on AVA_VAN.3 (vulnerability analysis and penetration testing) is expected to be equivalent to 25 man day for a typical platform to reach Enhanced-Basic attack potential.

Definition of the "equivalent effort" and "typical platform" is at the discretion of the implementing SESIP scheme in alignment with the SESIP community.

Note that in case of a composed platform, the evaluator must verify that all objectives for the environment of the platform parts fulfilled by another platform part are accurately fulfilled. All guidance of one platform part for another platform part should be considered as part of the vulnerability analysis.

3.4 SESIP Assurance Level 4 (SESIP4)

SESIP4 is an extended-substantial level of assurance for (parts of) IoT platforms.

It provides significantly more assurance than SESIP3 by not limiting the time applied but requiring sufficient time and effort be applied by the evaluator to the source code analysis and the subsequent vulnerability analysis to address the higher attack potential. More assurance is also provided through the inclusion of requirements for additional lifecycle details and delivery of developer test evidence.

At this assurance level, the developer is expected to provide:

- A simplified Security Target with self-assessment rationale. ST templates are available on the SESIP website. The self-assessment rationale includes:
    - a (self-)check against publicly known vulnerabilities against the platform,
    - references to guidance documents describing the objectives for the environment, and
    - a reference to the internal and public facing procedures describing how flaws are reported, tracked and corrected, and the resulting updates communicated.
- Guidance documents. This typically consists of the existing user manuals and data sheets.
- A complete functional specification. This typically consists of existing programmer's manuals, data books or API specifications, and a mapping showing what interfaces implement the SFRs declared in the Security Target.
- The full source code, and a mapping showing where in the source code the SFRs are implemented. This mapping typically is a minor extension of the mapping for the functional specification.
- Proof of functional conformance testing.
- A short description showing how the platform's version number is maintained to be uniquely identifying the platform in its version, and showing that the whole platform (source code and all that is described above) is maintained in a standard version management system (such as CVS, GIT or SVN).
- Internal Flaw Reporting procedures describing internal (as well as the public facing) procedures describing how flaws are reported, their security impact is assessed, tracked and corrected, and the resulting updates communicated.
- Internal procedures describing the delivery of the platform to the integrator
- Internal procedures describing the physical, procedural, personnel, and other security measures that are applied at development and manufacturing sites to protect the confidentiality and integrity of the TOE design and implementation.
- List of tools and associated versions of all tools used in the development of the TOE. Identify the documentation providing the definition of the tools and, where applicable, identify the documented use of the tools.
- Developer testing evidence demonstrating the developer approach to demonstrate the security functionality that can be exercised through the external interfaces of the platform.

3.4.1 Objectives

SESIP4 provides assurance by a simplified security target, and an analysis of the SFRs in that ST, using a full functional specification, guidance documentation, and the implementation representation, to understand the security behaviour.

The analysis is supported by independent testing of the TSF, selective independent confirmation of the developer test results and a vulnerability analysis (based upon the functional specification, implementation representation and guidance evidence provided) demonstrating resistance to penetration attackers with a moderate attack potential.

SESIP4 also provides assurance through the use of development environment controls and additional TOE configuration management including automation and definition of tools used, evidence of secure delivery procedures and site security procedures.

3.4.2 Assurance components

Assurance Class	**Assurance Families**
ASE: Security Target evaluation	ASE_INT.1 ST Introduction *ASE_OBJ.1 Security requirements for the operational environment* **ASE_REQ.3 Listed Security requirements** *ASE_TSS.1 TOE Summary Specification*
ADV: Development	ADV_ARC.1 Security architecture description ADV_FSP.4 Complete functional specification **ADV_IMP.3 Complete mapping of the implementation representation of the TSF to the SFRs**
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE ALC_CMS.1 TOE CM Coverage ALC_FLR.2 Flaw reporting procedures ALC_DEL.1 Delivery procedures ALC_DVS.1 Identification of security measures ALC_TAT.1 Well-defined development tools
ATE: Tests	ATE_COV.1 Evidence of coverage ATE_FUN.1 Functional testing ATE_IND.1 Independent testing: conformance
AVA: Vulnerability Assessment	AVA_VAN.4 Methodical Vulnerability analysis

3.4.3 Security Target Requirements

As per section 3.1.3.

3.4.4 Development Requirements

ADV_ARC.1 Security Architecture Definition

Dependencies: ADV_FSP.1 Basic functional specification, ADV_TDS.1 Basic design

ADV_FSP.1 is met by ADV_FSP.4.

ADV_TDS.1 dependency is met by ADV_IMP.3.3D.

For the rest as per CC Part 3.

ADV_FSP.4 Complete functional specification

Dependencies: ADV_TDS.1 Basic design

This dependency is considered to be fulfilled with ADV_IMP.3.3D.

For the rest as per CC Part 3.

ADV_IMP.3 Complete mapping of the implementation representation of the TSF to the SFRs

Dependencies: ASE_REQ.3 Listed security requirements

For the rest as per the definition of ADV_IMP.3 in Development Requirements.

3.4.5 Guidance Documents Requirements

AGD_OPE.1 Operational User Guidance

As per CC Part 3.

AGD_PRE.1 Preparative Procedures

As per CC Part 3.

3.4.6 Lifecycle Support Requirements

ALC_CMC.1 Labelling of the TOE

As per CC Part 3.

ALC_CMS.1 TOE CM Coverage

As per CC Part 3.

ALC_FLR.2 Flaw Reporting Procedures

As per section 3.1.4.

Note that at SESIP3 level and higher, the entire flaw reporting procedure is to be described; not just the externally visible steps required for SESIP1 and SESIP2 levels. Note also that the flaw reporting procedure must include a determination of security impact starting at SESIP4 level.

ALC_DEL.1 Delivery procedures

As per CC Part 3.

ALC_DVS.1 Identification of security measures

As per CC Part 3.

ALC_TAT.1 Well-defined development tools

As per CC Part 3.

3.4.7 Tests Requirements

ATE_COV.1 Evidence of coverage

As per CC Part 3.

ATE_FUN.1 Functional testing

As per CC Part 3.

ATE_IND.1 Independent testing: conformance

As per section 3.2.7.

3.4.8 Vulnerability Analysis Requirements

AVA_VAN.4 Methodical vulnerability analysis

Dependencies:

AGD_OPE.1 and AGD_PRE.1 are met.

ADV_FSP.2 is met by ADV_FSP.4.

ADV_ARC.1 is met

ADV_TDS.3 is considered to be met by ADV_IMP.3 in combination with ASE_TSS.1.

ADV_IMP.1 is met by ADV_IMP.3

ATE_DPT.1 is met through a combination of ATE_IND.1 and ADV_IMP.3 informing the selection of independent test cases to ensure all major parts (subsystems) of the TOE have been tested.

Note

The necessary time and effort must be spent performing source code analysis (for ADV_IMP.3) and performing vulnerability analysis and penetration testing (for AVA_VAN.4) to demonstrate that the platform defends against attackers with Moderate attack potential. There is no time or effort limit, only an assurance goal.

3.5 SESIP Assurance Level 5 (SESIP5)

SESIP5 is the same as the standard high assurance level currently being used for smartcards, secure elements, E-passports etc[3][4]. It provides a very robust defence against very advanced threats.

During the trial this assurance level is exclusively for re-use of SOG-IS certified platforms by licensed labs, allowing those platforms to utilize the mappings from SESIP to specific commercial product domains immediately.

3.5.1 Objectives

SESIP5 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and complete interface specification, guidance documentation, a description of the basic modular design of the TOE, and the entire implementation, to understand the security behaviour.

The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification and TOE design, selective independent confirmation of the developer test results, and a vulnerability analysis (based upon the functional specification, TOE design, implementation representation, security architecture description and guidance evidence provided) demonstrating resistance to penetration attackers with a High attack potential.

SESIP5 also provides assurance through the use of development environment controls and additional TOE configuration management including automation, evidence of secure delivery procedures, and, where possible procedures for updating the TOE in the field.

3.5.2 Assurance components

Assurance Class	**Assurance Families**
ASE: Security Target evaluation	ASE_INT.1 ST Introduction ASE_CCL.1 Conformance claims ASE_ECD.1 Extended components definition ASE_OBJ.2 Security objectives **ASE_REQ.3 Listed Security requirements** ASE_SPD.1 Security problem definition *ASE_TSS.1 TOE Summary Specification*
ADV: Development	ADV_ARC.1 Security architecture description ADV_FSP.4 Complete functional specification ADV_TDS.3 Basic modular design ADV_IMP.2 Complete mapping of the implementation representation of the TSF
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.4 Production support, acceptance

[3] See for instance the Security IC Platform Protection Profile BSI-PP-0084-2014.
[4] With the exception of the addition of ALC_FLR.2, as updating of flaws is considered a very important functionality for IoT platforms.

	procedures and automation ALC_CMS.4 Problem tracking CM coverage ALC_DEL.1 Delivery procedures ALC_DVS.2 Sufficiency of security measures ALC_FLR.2 Flaw reporting procedures ALC_LCD.1 Developer defined life-cycle model ALC_TAT.1 Well-defined development tools
ATE: Tests	ATE_COV.2 Analysis of coverage ATE_DPT.1 Testing: basic design ATE_FUN.1 Functional testing ATE_IND.2 Independent testing - sample
AVA: Vulnerability Assessment	AVA_VAN.5 Advanced methodical vulnerability analysis

All but ASE_REQ.3 are as per CC part 3. Optimized methodology may be available with a scheme.

4 Guidance: Vulnerability analysis and attack potential rating

SESIP defines a methodology for evaluation as a variant to the Common Criteria standard. The definition of scheme-optimized attack potential and attack rating methodology is an essential part of any certification scheme based on SESIP. This appendix contains the standard methodology such a scheme-optimized methodology should be compatible with.

4.1 Attack rating calculation in the sum of the identification and the exploitation phases

The vulnerability analysis is performed in accordance with the Common Criteria [CC]. The rating is performed in accordance with the [AM] or [CC] v2.x methodology: by distinguishing in an "identification" and "exploitation" phase.

In the identification phase, the attacker has remote and physical (local) access to the platform to identify the vulnerabilities and prepare the exploitation of that vulnerability to break one of the SFRs. In the exploitation phase, the attacker has at least remote access to the platform.

This base threat model covers scalable attacks over "the internet". For example, an attacker in the base threat model of SESIP might read out the firmware of the platform using an open debug interface, analyse it, and find a buffer overflow exploitable remotely on another instance of the platform.

4.2 Physical (local) attacks and remote attacks

"Physical (local) attack" is any attack that require physical access to the platform to exploit the vulnerability. For example: the (ab)use of a debug interface like JTAG, the reading out of an external memory like a flash memory, many of the fault injection and side channel attacks, physical modifications like a FIB. To address contactless fault injection and side channel attacks, physical proximity of up to 5cm is considered to be 'physical access'.

"Remote attack" is any attack that does not require physical access to the platform to exploit the vulnerability. Note that attacks from a local network are considered to be remote attacks: SESIP does not distinguish between a trusted local network and the hostile internet, any connectivity made available is considered to be accessible to even the basic attacker.

4.3 Attack potential rating

The attack potential should be calculated with the following rating tables, replacing the general rating table of the CC. The minimal sum of the identification and exploitation costs of a vulnerability is the total attack rating.

Note that these rating tables are consistent with the rating used in smart card devices as described in [AM].

Factors	Identification	Exploitation	Notes
Elapsed time			
< one hour	0	0	-
< one day	1	3	
< one week	2	4	
< one month	3	6	
> one month	5	8	
Not practical	*	*	

Expertise			
Layman	0	0	-
Proficient	2	2	
Expert	5	4	
Multiple Expert	7	6	
Knowledge of the TOE			
Public	0	0	Critical or higher can only be claimed if all sites with access to that information are included in the scope of the evaluation at ALC_DVS.2 level (i.e. SESIP5)[5].
Restricted	2	2	
Sensitive	4	3	
Critical	6	5	
Very critical hardware design	9	NA	
Access to TOE			
< 10 samples	0	0	It is unlikely that evaluations under SESIP4 will rate more than 10 samples in the attack.
< 100 samples	2	4	
> 100 samples	3	6	
Not practical	*	*	
Equipment			
None	0	0	-
Standard	1	2	
Specialized	3	4	
Bespoke	5	6	
Multiple Bespoke	7	8	
Open samples			
Public	0	NA	Sensitive or higher can only be claimed if all sites with access to such open samples are included in the scope of the evaluation at ALC_DVS.2 level (i.e. SESIP5)[6].
Restricted	2	NA	
Sensitive	4	NA	
Critical	6	NA	

The table below shows what the total rating of identification + exploitation needs to be to meet the attack potentials.

[5] Licensed evaluation labs and the CB should be considered to meet this requirement.
[6] Licensed evaluation labs should be considered to meet this requirement.

SESIP				Standard CC rating (in [CEM]) for comparison[7]		
	Rating of at least	(Range of values)	**Meets attack potential of**	Rating of at least	(Range of values)[8]	Meets attack potential of
SESIP Assurance Level 1 (SESIP1)	**16**[9]	0-15	**Basic (AVA_VAN.1)**	14	14-19	Enhanced basic (AVA_VAN.3)
SESIP Assurance Level 2 (SESIP2)	**16**	0-15	**Basic (AVA_VAN.2)**	14	14-19	Enhanced basic (AVA_VAN.3)
SESIP Assurance Level 3 (SESIP3)	**21**	16-20	**Enhanced-Basic (AVA_VAN.3)**	20	20-24	Moderate (AVA_VAN.4)
SESIP Assurance Level 4 (SESIP4)	**25**	25-30	**Moderate (AVA_VAN.4)**	25	≥25	High (AVA_VAN.5)
SESIP Assurance Level 5 (SESIP5)	**31**	31 and above	**High (AVA_VAN.5)**	Does not exist		

[7] This shows that the rating of SESIP and [AM] at AVA_VAN.x is effectively exceeding the standard CC AVA_VAN.x+1 level, due to the higher minimal ratings required in

[8] The failing values are indicated in ranges in the [AM] and [CEM] documents, the minimal passing value is listed in the “Rating of at least” column.

[9] There is **no** vulnerability analysis or penetration testing at this level, only self-assessment.

5 Example use cases

5.1 Generic examples

5.1.1 IoT cloud connectivity platform

The first example is an IoT cloud connectivity platform, which includes both hardware (a PCB with a microprocessor, some memory, and some peripherals, including at least a network connection) and software (an operating system, and a connectivity layer that manages all communications with the IoT cloud, and provides a high-level API to applications).

This is a complete IoT platform, which is likely to support most of the SFRs.

For instance, if the platform supports platform- and application-level attestations, it would include the following SFRs:

SFR	*Rationale for inclusion in ST*
Identification of platform type: The platform provides a unique identification of the platform type, including all its parts and their versions.	*This identification is mandatory for all SESIP platforms*
Identification of individual platform: The platform provides a unique identification of that specific instantiation of the platform, including all its parts and their versions.	*An individual ID of each instance is needed to support attestations*
Genuine platform instantiation: The platform provides an attestation of the "Identification of platform type" and "Identification of individual platform", in a way that cannot be cloned or changed without detection.	*The platform can generate a proof of its identity, typically by using an instance-specific secret*
Secure initialization of platform: The platform ensures its authenticity and integrity during the platform initialization. If the platform authenticity or integrity cannot be ensured, the platform will go to a secure state.	*This is the secure boot of the platform*
Attested secure state of platform: The platform provides an attestation of the state of the platform, such that it can be determined that the platform is in a secure state.	*An attestation can be generated to provide some assurance about the platform's authenticity and current state*
Genuine application: The platform provides an attestation of the application, in a way that cannot be cloned or changed without detection.	*This is an extension of the platform's identity proof and secure boot to its application*
Attested state of application: The platform provides an attestation of state of the application.	*The attestation provides assurance on the application state as seen by the platform.*

Such an IoT platform would most likely also cover the full range of lifecycle management and update SFRs, not shown here, as well as the secure communications SFRs below:

Secure communication support: The secure communication channel authenticates the IoT Cloud endpoint and protects against disclosure, modification, replay, erasure of messages between the endpoints, using TLS1.2 with TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite.	*The first step is to define what protocols are used to protect a given connection. Here, the connection to the cloud is protected by TLS1.2.*

Secure communication enforcement: The platform ensures the application can only communicate with the IoT Cloud endpoint over the secure communication channel(s) supported by the platform.	*The second step makes that secure protocol the only way to communicate with the IoT Cloud.*

Next comes the attack resistance. If we assume that the PP will be used to target platforms used in exposed and sensitive devices, some extra resistance is required. Note that some of these SFRs may be instantiated more than once:

Physical attacker resistance: The platform detects or prevents attacks by an attacker with physical access before the attacker compromises any of the other functional requirements, ensuring that the other functional requirements are not compromised.	*Physical attacks are fully covered, using any means available at the expected assurance level, targeting any part of the platform.*
Software attacker resistance: isolation of platform: The platform provides isolation between the application and itself, such that an attacker able to run code as an application on the platform cannot compromise the other functional requirements.	*This level requires a separation between the platform and application, mostly through user/system distinctions expected on a complex chip.*
Software attacker resistance: isolation of platform parts: The platform provides isolation between platform parts, such that an attacker able to run code in the main CPU can compromise neither the integrity and confidentiality of the integrated Secure Element nor the provision of any other security functional requirements.	*This is the isolation of a dedicated security subsystem and its critical assets (keys, in particular).*
Software attacker resistance: isolation of platform parts: The platform provides isolation between platform parts, such that an attacker able to run code in the main CPU's non-secure environment, including the application, can compromise neither the integrity and confidentiality of the main CPU's secure environment nor the provision of any other security functional requirements.	*This is the isolation provided by a TEE between a secure and a non-secure world.*

Finally, an application must provide services, which, for basic IoT applications, could be limited to high-level functions:

Secure encrypted storage: The platform ensures that all data stored by the application, with the exception of data stored in containers tagged as PLAIN_TEXT by the application, is encrypted as specified in AES (NIST FIPS 197) with a platform instance unique key of keylength 256 bits.	*The platform offers two types of containers to the application, one of them being protected by encryption.*
Residual information purging: The platform ensures that all memory allocated through the platform, with no exception, is erased using the method specified in NIST SP-800-88 Rev. 1 (zeroization) before the memory is (re)used by the platform or application again and before an attacker can access it.	*All memory is cleared before reused (including RAM, as long as it is managed by the platform).*

5.1.2 Root-of-Trust based on a microcontroller

As security hardware becomes increasingly complex, the Root-of-Trust becomes a significant abstraction layer, to model the security functions provided by chip vendors. Arm has defined such an abstraction layer as part of their Platform Security Architecture (PSA) initiative, and they also have defined as part of the related PSA Certified initiative a Protection Profile for the certification of PSA-compliant roots-of-trust.

This Protection Profile can be mapped to SESIP, and it would lead to the following selection of SFRs:

SFR	*Rationale for inclusion in ST*
Identification of platform type: The platform provides a unique identification of the platform type, including all its parts and their versions.	*This SFR is mandatory for all SESIP platform (parts).*
Identification of individual platform: The platform provides a unique identification of that specific instantiation of the platform, including all its parts and their versions.	*Dependency from "Genuine platform instantiation" that provides a unique identifier*
Genuine platform instantiation: The platform provides an attestation of the "Identification of platform type" and "Identification of individual platform", in a way that cannot be cloned or changed without detection.	*Corresponds to F.ATTESTATION for the platform's identity*
Secure initialization of platform: The platform ensures its authenticity and integrity during the platform initialization. If the platform authenticity or integrity cannot be ensured, the platform will go to a secure state.	*Corresponds to F.INITIALIZATION and F.SECURE_STATE, providing secure boot features*
Attested secure state of platform: The platform provides an attestation of the state of the platform, such that it can be determined that the platform is in a secure state.	*Corresponds to F.ATTESTATION and F.SECURE_STATE to provide an externally verifiable platform-level attestation, essential in PSA*
Genuine application: The platform provides an attestation of the application, in a way that cannot be cloned or changed without detection.	*Dependency from "Attested state of application", a basis for the application-level attestation*
Attested state of application: The platform provides an attestation of state of the application.	*Corresponds to F.ATTESTATION, here extended to include the application and its state (to be detailed in ST)*
Secure update of platform: The platform can be updated to a newer version in the field such that the integrity, authenticity and confidentiality of the platform is maintained.	*Corresponds to F.FIRMWARE_UPDATE for the platform part*
Secure update of application: The application can be updated to a newer version in the field such that the integrity, authenticity and confidentiality of the application is maintained.	*Corresponds to F.FIRMWARE_UPDATE for the application part*
Software attacker resistance: isolation of platform parts: The platform provides isolation between platform parts, such that an attacker able to run code in **the NSPE** can compromise neither the integrity and confidentiality of **the SPE** nor the provision of any other security functional requirements.	*Corresponds to F.SOFTWARE_ISOLATION for the isolation between SPE and NSPE (both parts of a full IoT platform) at isolation Level 2*
Software attacker resistance: isolation of platform parts: The platform provides isolation between platform parts, such that an attacker able to run code in **the NSPE's Application Roots-of-Trust** can compromise neither the integrity and confidentiality of **the PSA Root-of-Trust** nor the provision of any other security functional requirements.	*Corresponds to F.SOFTWARE_ISOLATION for the isolation the PSA RoT from Application RoTs (both parts of the NSPE) at isolation Level 3*
Cryptographic operation: The platform provides the application with <*list of cryptographic operations*> functionality with <*list of algorithms*> as specified in <*specification*> for key lengths <*list of key lengths*> and modes <*list of modes*>.	*Corresponds to F.CRYPTO, to be iterated as much as needed*

Cryptographic key generation: The platform provides the application with a way to generate cryptographic keys for use in <*list of cryptographic algorithms*> as specified in <*specification*> for key lengths <*list of key lengths*>.	*Corresponds to F.CRYPTO, to be iterated as much as needed*
Cryptographic keystore: The platform provides the application with a way to store <*list of assets, such as cryptographic keys and passwords*> such that not even the application can compromise the <*authenticity, integrity, confidentiality*> of this data. This data can be used for the cryptographic operations <*list of operations*>.	*Corresponds to F.SECURE_STORAGE, which does not differentiate between crypto assets and general assets*
Cryptographic random number generation: The platform provides the application with a way based on <*list of entropy sources* > to generate random numbers to as specified in <*specification*>.	*Not explicitly mentioned in F.CRYPTO, but a likely implicit requirement*
Secure encrypted storage: The platform ensures that all data stored by the application, with the exception of <*list of data stored in plaintext*>, is encrypted as specified in <*specification*> with a platform instance unique key of key length <*key length*>	*Corresponds to F.SECURE_STORAGE for application assets*
Audit log generation and storage: The platform generates and maintains an audit log of <*list of significant security events*> and allows access and analysis of these logs following a specific <*access control policy*>.	*Corresponds to F.AUDIT*
Secure debugging: The platform only provides <*list of endpoints*> authenticated as specified in <*specification*> with debug functionality. The platform ensures that all data stored by the application, with the exception of <*exceptions*>, is made unavailable.	*Corresponds to F.DEBUG with a few additional details about data protection*

5.2 Examples for specific use cases

A scheme based on the SESIP standard should provide a way for platform developers (hardware and software) to show that certain security functional requirements are met, such that product developers can build a secure product on top and with it.

5.2.1 Secure update of a device (OTA)

If the platform developer wants to facilitate some secure update of the whole product (for example over the air update of critical part of a car), this would be the set of requirements for the platform:

- Identification of platform type:
  The platform provides a unique identification of the platform type, including all its parts and their versions.
- Secure update of platform
  The platform can be updated in the field such that the integrity, authenticity and confidentiality of the platform is maintained.
- Secure update of application:
  The product can be updated in the field such that the integrity, authenticity and confidentiality of the product is maintained.

A more extensive platform supporting checks that the product is genuine and operating correctly, would add:

- Genuine application:
  The platform provides an attestation of the application, in a way that cannot be cloned or changed without detection.
- Attested state of application:
  The platform provides an attestation of state of the application.

5.2.2 A blood glucose measurement device (DTSec)

A blood glucose measurement device uses a platform for a secure channel over Bluetooth LE 4.1 with a mobile device, and cryptographic operations to make signatures and verify them.

This product will need to fulfill DTSec requirements.

A suitable ST for the platform built from hardware and software could contain the following security functional requirements:

- Identification of platform type:
  The platform provides a unique identification of the platform type, including all its parts and their versions.
- Secure communication support:
  The platform provides the product with the secure communication channels it can optionally use.
  The channels authenticate **the platform and any external party to each other**, protect against **disclosure and modification** of messages between these endpoints, using **Bluetooth LE 4.1 with options x,y,z always enabled**.
- Cryptographic operation:
  The platform provides the product with **hashing, signing, and signature verification** functionality as specified in **<DTSec compliant references here>** for keylengths of **1024-2048 bit** and modes **PKCS#11**.
- Cryptographic random number generation:
  The platform provides the product with a way based on **combined physical noise and cryptographic computation** to generate random numbers to as specified in <**DTSec compliant references here, for example FIPS-something section x.y>**.

If the platform developer wants to facilitate the product developer to meet the DTSec scheme specific requirements easily, a hardened platform could implement parts of the requirements already in a way not even the product can circumvent. The DTSec evaluation would then be limited to verifying the "Genuine application" and "Attested state of application" return the right values, and that the signature setting and verification is implemented in the way intended:

- Identification of platform type:
  The platform provides a unique identification of the platform. This is uniquely identifies all parts of the platform (hardware and software parts) and their versions.
- Genuine application:
  The platform provides an attestation of the application, in a way that cannot be cloned or changed without detection.
- Attested state of application:
  The platform provides an attestation of state of the application.
- Secure update of platform:
  The platform can be updated in the field such that the integrity, authenticity and confidentiality of the platform is maintained.
- Software attacker resistance: isolation of platform:
  The platform provides isolation between the application and itself, such that an attacker able to run code as an application on the platform cannot compromise the other functional requirements.

- Secure communication support:
  The secure communication channel authenticates **any external party** and protects against **disclosure and modification** of messages between the endpoints, using **Bluetooth LE 4.1 with options x,y,z always enabled**.
- Secure communication enforcement:
  The platform ensures the application can only communicate with **any external party** over the secure communication channel(s) supported by the platform.
- Secure encrypted storage:
  The platform ensures that all data stored by the product, with the exception of **data stored in attached SD cards**, is encrypted as specified in <**DTSEC compliant references here, for example AES FIPS-something section x.y>** with a product unique key of keylength of **128 bits.**

The product will still need to implement (and be evaluated) to properly sign/verify data exchanged, so the platform needs to provide:

- Cryptographic operation:
  The platform provides the product with **hashing, signing, and signature verification** functionality as specified in **<DTSEC compliant references here>** for keylengths of **1024-2048 bit** and modes **PKCS#11**.
- Cryptographic random number generation:
  The platform provides the product with a way based on **combined physical noise and cryptographic computation** to generate random numbers to as specified in <**DTSEC compliant references here, for example FIPS-something section x.y>**.

6 Guidance for schemes implemented using SESIP

The SESIP standard has been designed for efficient high-quality evaluations and certifications. As such, there are assumptions on the evaluation labs and certification bodies to be able to apply this standard. Evaluation labs and certification bodies must comply to these assumptions for their results to be meaningful and SESIP compliant.

6.1 Evaluation lab

As SESIP builds on Common Criteria, understanding of at least the CC assurance requirements is needed for the evaluation lab to perform their tasks: the evaluation lab must be ISO 17025 accredited with the scope including as methodology at least the underlying Common Criteria ([CC] and [CEM]), and should [10]also include the SESIP methodology. The scope should also include the relevant technical capabilities, especially for analysing and testing the "Extra attacker resistance" SFRs.

If the evaluation lab performs evaluations against SESIP3 or higher on platforms claiming "Limited physical attacker resistance" or "Physical attacker resistance" SFRs, the lab should be accredited by the CB using [ITSEF] requirements. At SESIP4 or higher, the lab should be accredited by the CB as capable to use [JHAS] rating methodology.

6.2 Certification body

The CB verifies that the assurance requirements are met and can make exceptions, the CB must be experienced in the assurance requirements. SESIP expresses these assurance requirements in terms of CC assurance requirements, hence a CB must be experienced in CC assurance requirements: the certification body must be ISO 17065 accredited with the scope including at least the underlying Common Criteria ([CC] and [CEM]), preferably also the SESIP methodology.

For certification of platforms claiming "Limited physical attacker resistance" or "Physical attacker resistance" SFRs at SESIP3 and higher, the CB must be expert in assessing the evaluation lab using [ITSEF] requirements. At SESIP4 or higher, the CB must be expert in using [JHAS] rating methodology.

The CB can only authorize deviations from the current SESIP standard for trial use, and only if the CB has at least a year experience with SESIP and seeks alignment on this deviation with the community.

[10] Under some accreditation bodies, one can only apply for SESIP in the scope if SESIP is already in use in a scheme recognized by that accreditation body. Currently, the only SESIP scheme is the one managed by TrustCB, so by making SESIP optional this "should" opens SESIP up for other schemes and CBs.

7 References

7.1 Terminology

CC	Common Criteria
CEM	CC Evaluation Methodology
ICD	In-Circuit Debugger, allows for debugging of a live platform.
JHAS	JIL Hardware Attacks Subgroup, workgroup under Eurosmart and JIL, maintaining and further developing the industry-standard attack rating for integrated ICs and similar devices.
JTAG	Standardised interface for debugging, testing and programming devices.
SmartCC	Project code name for the Smart application of the CC, such as the MIFARE v3.0 scheme and the SESIP standard

7.2 Bibliography

[AM] Application of Attack Potential to Smartcards, Joint Interpretation Library, version 2.9, dated January 2013

[CC] Common Criteria for Information Technology Security Evaluation, Parts I, II and III, Version 3.1 Revision 5, April 2017

[CEM] Common Methodology for Information Technology Security Evaluation, Version 3.1 Revision 5, April 2017

[ITSEF] Minimum ITSEF Requirements for Security Evaluations of Smart cards and similar devices, version 2.0, January 2017, https://www.sogis.eu/uk/supporting_doc_en.html

Minimum ITSEF Requirements for Security Evaluations of Hardware Devices with Security Boxes, version 1.0, January 2018, https://www.sogis.eu/uk/supporting_doc_en.html

[JHAS] Attack Methods for Smartcards and Similar Devices

[PP-0084] Security IC Platform Protection Profile with Augmentation Packages, version 1.0, reference BSI-CC-PP-0084-2014

7.3 Revision history

Version	Primary author(s)	Changes
1.0	Wouter Slegers (TrustCB), Dirk-Jan Out (Brightsight)	Original version
1.1	Wouter Slegers (TrustCB)	Renamed ITPx to SESIPx. Clarified (but did not change) the rating methodology.
1.2	Wouter Slegers (TrustCB)	Added SFRs. Creation of SESIP logos. Added explicit description of theexpected developer deliverables.
1.3	Wouter Slegers (TrustCB)	Added references to traditional CC. Added SFRs. Defined SESIP4. Renamed SESIP 1/1+/2/2+/3 to SESIP1/2/3/4/5 and aligned these with VAN.1/2/3/4/5.