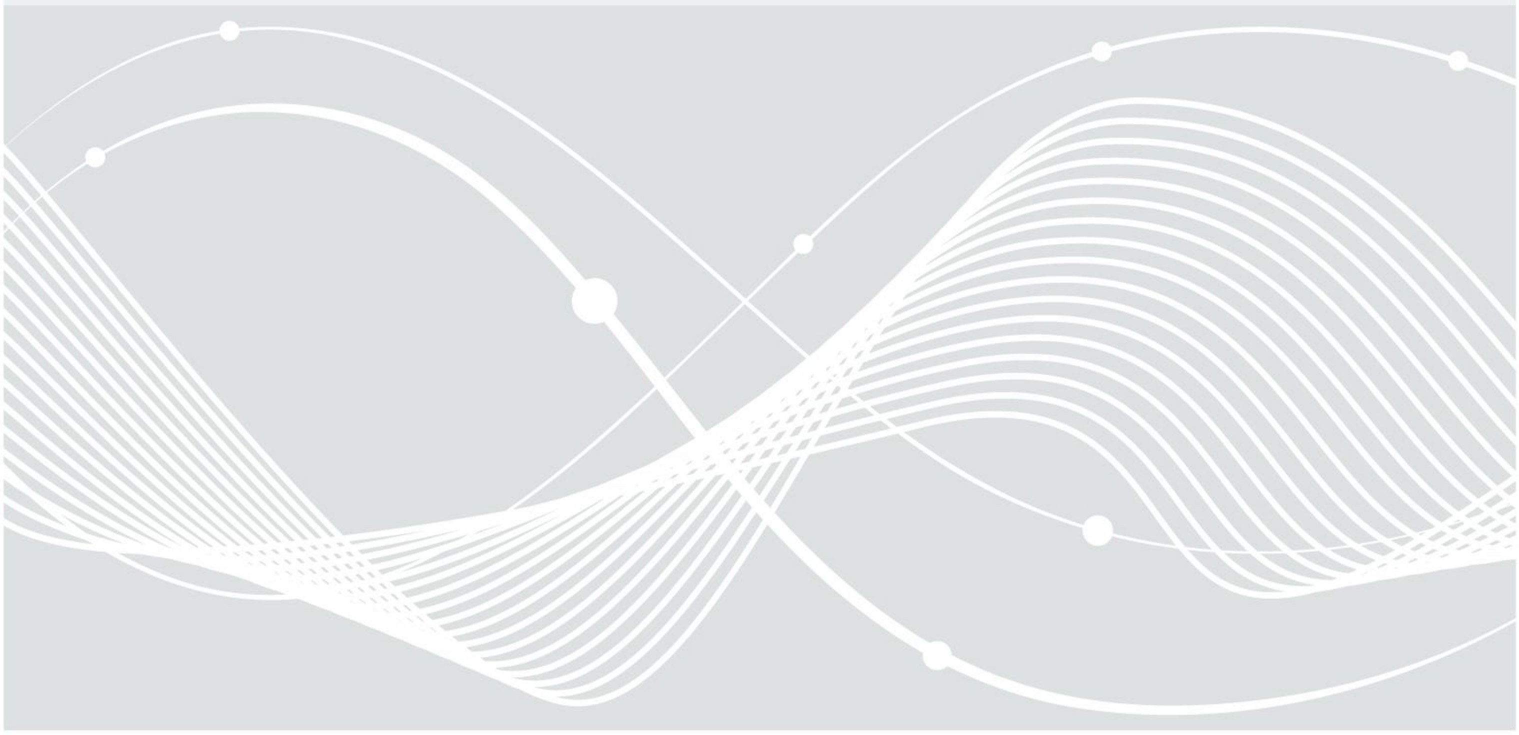
Federal Office
for Information Security

Common Criteria Protection Profile

FIDO Universal Second Factor (U2F) Authenticator

BSI-PP-CC-0096-2017



Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn

Internet: https://www.bsi.bund.de

Contents

1 PP Introduction

1.1 PP Reference

Title
FIDO Universal Second Factor (U2F) Authenticator

CC Version
3.1 (Revision 4)

Assurance Level
Minimum assurance level for this PP is EAL4 augmented.

General Status
in development

Version Number
Version 1.0 (Jun 26th, 2017)

Registration
BSI-CC-PP-0096-2017

Keywords
FIDO, U2F

1.2 TOE Overview

1.2.1 TOE Definition and Operational Use

The TOE addressed by the current protection profile (PP) is a FIDO Authenticator intended for FIDO Universal Second Factor (U2F) authentication [FIDOSpec]. In this PP a specific implementation of the U2F token is considered. The authenticator is physically implemented as a security chip with an application and is used to securely access online services. The authenticator, also referred to as a „U2F token“ (or just „token“), communicates with an external server controlled by a relying party (RP) that supports the standardised FIDO U2F protocol.

The main goal of U2F based authentication is to provide a strong second-factor authentication mechanism for web-users while preserving the user's privacy. The second factor is the U2F authenticator, carried by the user. A U2F authenticator thus augments the security of the commonly used user name and password mechanism.

To authenticate himself towards the RP, the user has to prove his presence by interacting with the authenticator (e.g. by pressing some button or placing the token into the proximity range of an NFC-enabled device) or by entering a PIN code. The TOE then uses cryptographic material, namely a private key from an asymmetric key pair that is generated and used by the token, to log in. Key pairs are unique for each tuple of relying party, user account and U2F authenticator, thus preventing to trace the user over different relying parties and accounts.

Two main steps have to be performed, the registration process and the authentication process. In the following, we assume the existence of a *seed* and a *secret key for MAC computation (MACKey)* on the token, whereupon seed and MACKey have to be different. The MACKey and the seed are generated in the initialisation process during the first usage of the authenticator by the end-user and stored securely on the device. The user can also afterwards reset the device and thus destruct all existing key material. After that, the initialisation process has to be repeated before the next usage.

During the registration process, a site/application/account-specific asymmetric key pair is generated. First, the relying party submits the AppID (a 32 byte value) to the authenticator. The authenticator then generates a random nonce. Then a private key is generated using a key derivation function (KDF). The input to the KDF is the seed, the AppID as well as the previously generated fresh nonce. The public asymmetric key pair is generated from the private key using the underlying asymmetric cryptographic algorithm. The authenticator proceeds to generate a *key handle* that allows the authenticator later on to recover the generated key pair in the authentication phase. The key handle consists of the generated nonce, as well as a message authentication code (MAC) over the AppID and nonce using the MACKey. The key handle together with the public key is then transmitted to and stored at the relying party.

During authentication, the authenticator receives (via the user device) the AppID, the previously registered key handle as well as a challenge. The authenticator verifies the message authentication code and thus checks whether the supplied key handle contains a nonce generated by the authenticator that fits to the supplied AppID. After successful verification, the KDF is activated with the seed together with the AppID and nonce to re-generate the private key. Then, the challenge as well as the AppID are signed with the private key, and the challenge together with the signature is sent to the relying party. After successful verification of the signature with the public key of the authenticator (which was stored during registration at the relying party), the relying party can conclude that indeed the user authenticated himself to the service.

The four steps that can be handled by the authenticator (initialisation, reset, registration and authentication) are visualised in figure 1 to 4. These pseudo-code descriptions are purely informative. Relevant details for the technical implementation can be found in the security requirements chapter 5.1.

This description omits some parts from the specification, e.g. the user-device in between the relying party and the token, as well as some implementation-specific details. For a full specification of the FIDO standard we refer to [FIDOSpec].

```
                        Initialisation Process
U2F Authenticator                                Relying Party

if (user_presence == true) and (security_state == delivery_state) {
   seed = RNG()
   MACKey = RNG()
   security_state = ready_for_use
}
manage_user_presence()
```

Figure 1: Process description of the initialisation step

```
                        Reset Process
U2F Authenticator                                Relying Party

if (user_presence == true) and (security_state == ready_for_use) {
   security_state = delivery_state
   secure_erase(seed)
   secure_erase(MACKey)
}
manage_user_presence()
```

Figure 2: Process description of the reset step

```
                        Registration Process
U2F Authenticator                                Relying Party
                          <-------- App ID  ---------
if (user_presence == true) and (security_state == ready_for_use){
   nonce = RNG()
   PrivK = KDF(seed, AppID, nonce)
   PubK  = genKeyPair(PrivK)
   secure_erase(PrivK)
   KeyHandle =  nonce | MAC(MACKey, AppID | nonce)
                        ------ Key Handle, PubK ----->
   delete(PubK)
}
manage_user_presence()
```

Figure 3: Process description of the registration step

```
                        Authentication Process
U2F Authenticator                                     Relying Party
                         <-------- App ID  ---------
                  <----- KeyHandle (nonce | MAC) -------
                  <------------ challenge --------------
if (security_state == ready_for_use){
   if VerifyMac(MACKey, MAC) = false {
      error
   } else {
      if (user_presence == false){
          error
      } else {
         PrivK = KDF(seed, AppID, nonce)
         sig = sign(PrivK, AppID | challenge)
         secure_erase(PrivK)
                       ---------- sig ---------->
                                         verify(PubK,AppID | challenge,sig)
   }
}
manage_user_presence()
```

Figure 4: Process description of the authentication step

1.2.2 TOE Major Security Features

The following security goals are met by a certified device and thus describe the essential security features of the TOE:

- **Strong Authentication:** The TOE authenticates a user and/or a device to a relying party with high cryptographic strength. The protocol also protects against typical attacks during authentication, such as man-in-the-middle or phishing attacks.
- **Unlinkability:** The generated asymmetric key pair is unique for each relying party and account. All other information, occurring within the U2F protocol, that could be potentially used to link two accounts to the same user are protected by the TOE. Thus linking two accounts (e.g. by using public keys or other protocol information) is impossible, even if these two accounts are with the same relying party.
- **Privacy:** The authenticator does not store or require to associate any personal information with the identity of the user.
- **User Presence:** The U2F device has a physical test of user presence to ensure that the relying party can trust in that the authentication process is actively triggered by the user himself.

Note: Unlinkability and Privacy are only ensured, if not any other application are operated on the same authenticator.

1.2.3 TOE Type

The TOE is a contactless or contact-based secure chip including all IC dedicated software, embedded in an arbitrary housing with embedded software including the operating system and an application for FIDO U2F authentication.

The TOE requires a client application (e.g. a web-browser) to interact with the relying party. Neither the client application nor the relying party are part of the TOE.

1.2.4 TOE Life Cycle

This protection profile describes security objectives and requirements for an authenticator supporting the FIDO U2F protocol. The life-cycle and its order is viewed from a logical perspective on product development. In the likely case of a composite evaluation, the exhaustive guidance of [ICPP] should be taken into account by the ST-Writer. Some life-cycle phases may then take a different order, are split into separate, or combined into a single phase.

Stage 1: Development

Step 1 - IC Development (ICPP Phase2)

The IC developer develops the TOE in phase 1. This includes the IC design, the IC dedicated software and the guidance documentation associated with these TOE components.

Step 2 - Security IC Embedded Software Development (ICPP Phase 1)

The software developer uses the guidance documentation for the integrated circuit and the guidance documentation for relevant parts of the IC dedicated software, and develops the IC embedded software (operating system), the FIDO U2F application and the guidance documentation associated with these TOE components.

The manufacturing documentation of the IC including the IC dedicated software and the embedded software to be stored in the non-volatile non-programmable memories is securely delivered to the IC manufacturer. The IC embedded software to be stored in the non-volatile programmable memories, the application(s), and the guidance documentation is securely delivered to the chip manufacturer

Stage 2: Manufacturing

Step 3 – IC Manufacturing (ICPP Phase 3)

This step includes the integration and the IC production.

Step 4 – IC Packaging (ICPP Phase 4)

The IC is packaged and combined with hardware for the contact based or contactless interface.

Step 5 - Composite product integration (ICPP Phase 5)

The package is IC is combined into a composite product by the composite product integrator

Step 6 – Personalisation (ICPP Phase 6)

Creation of the application implies the creation of the master file (MF), dedicated files (DFs), and elementary files (EFs) according to [ISO7816-4] by the personalizer. How this process is handled internally depends on the IC and IC embedded software.

This is the finishing process and the personaliser delivers the TOE for operational use.

Stage 3: Operational Use

Step 7 – Operational Use (ICPP Phase 7)

The TOE is now in delivery state. The end-user has to command the token to (re-)seed and to generate the MACKey that is stored in the TOE during the initialisation process to switch the FIDO U2F authenticators security_state to state ready_for_use. For each registration a new nonce has to be generated. The nonce is then used to compute the private key:

nonce = RNG()
PrivK = KDF(seed, AppID, nonce)

If the end-user resets the authenticator the TOE is in delivery_state (Step 7). Subsequently new cryptographic key material has to be generated or the token cannot be used anymore. So all previously-done registrations are permanently unusable for future authentication. A reset can be performed at any time by the end-user.

Akin to [ICPP] , „TOE Delivery“ indicates delivery after Step 3 (or before Step 4) if the TOE is delivered in form of wafers or sawn wafers (dice) or after Step 4 (or before Step 5) if the TOE is delivered in form of packaged products.

The personaliser (party responsible for personalisation, Step 6) will be regarded as the **FIDO U2F token manufacturer, or simply manufacturer** in subsequent chapters. This party can have additional other ICPP roles, such as IC manufacturer, product/composite integrator, personaliser etc., but must at least comprise the role „Personaliser“. This should be clarified by the ST writer.

2 Conformance Claims

2.1 CC Conformance Claim

This protection profile claims conformance to

Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model; CCMB-2012-09-001, Version 3.1, Revision 4, September 2012 [CC1]

Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components; CCMB-2012-09-002, Version 3.1, Revision 4, September 2012 [CC2]

Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components; CCMB-2012-09-003, Version 3.1, Revision 4, September 2012 [CC3]

as follows

Part 2 extended,

Part 3 conformant.

The

Common Methodology for Information Technology Security Evaluation, Evaluation methodology; CCMB-2012-09-004, Version 3.1, Revision 4, September 2012 [CCEM]

has to be taken into account.

2.2 PP Claim

This PP does not claim conformance to any other protection profile.

2.3 Package Claim

The current PP is conformant to the following packages:

Assurance package EAL4 augmented with AVA_VAN.5 as defined in [CC3].

2.4 Conformance Rationale

This PP does not claim conformance to any other PP.

2.5 Conformance Statement

This PP requires strict conformance of any ST or PP claiming conformance to it.

3 Security Problem Definition

3.1 Assets, subjects and threat agents

3.1.1 Assets

Object No.	Asset	Definition
1	seed	The seed is once generated by the TOEs RNG. For this purpose the end-user has to execute a command to start the generation during the initialisation process. The seed is used to compute the private key and must never leave the TOE. In case of a reset, initiated by the end-user, the seed will be deleted. A new seed is generated by the RNG of the token if the end-user starts the initialisation process again.
2	MACKey	The MACKey is once generated by the TOEs RNG. For this purpose the end-user has to execute a command to start the generation during the initialisation process. The MACKey is used for MAC generation; the MAC itself is part of the key handle. The key must never leave the TOE. In case of a reset, initiated by the end-user, the MACKey will be deleted. A new MACKey is generated by the RNG of the token if the end-user starts the initialisation process again.
3	private keys	For each combination of relying party and account, one asymmetric key pair exists. A private key from such a key pair is generated temporarily on the TOE during the authentication and registration process for that account. Private keys must never leave the TOE.

Table 1: Assets to be protected by the TOE.

3.1.2 Subjects

This protection profile considers the following external entities and subjects:

Authenticator: FIDO Authenticator intended for FIDO Universal Second Factor (U2F) authentication. The authenticator, also referred to as a „U2F token“ (or just „token“), communicates with an external server controlled by a relying party (RP) that supports the standardised FIDO U2F protocol.

User: An *end-user* is the owner of the authenticator (TOE) who is legitimated to use the token.

Manufacturer: The *manufacturer* refers to the FIDO token manufacturer (cf. Life-Cycle)

Relying Party: A web site or other entity that uses a FIDO protocol to directly authenticate users (i.e., performs peer-entity authentication).

User Agent: A client application running on the user device that is acting on behalf of a user in a client-server system. Examples of user agents include web browsers and mobile applications.

Threat Agents: The attacker is a human or a process acting on his behalf located outside the TOE. The main goal of the attacker is to access the token in such a way that allows him to circumvent the authentication mechanism. He does so by e.g. trying to gain knowledge of secret information stored on the U2F authenticator. The attacker has high attack potential.

3.2 Threats

This section describes the threats to be averted by the TOE independently or in collaboration with its IT environment. These threats result from the assets protected by the TOE and the method of the TOE's use in the operational environment.

All attacks are executed through the threat agent described in the section above.

If the seed or private keys are compromised, an attacker is able to violate all security goals of FIDO. The attacker could impersonate the user with a cloned authenticator and has unauthorized access to the relying party. Similar, the compromise of the MACKey can lead to similar violations of the security goals of FIDO.

In the following the threats are listed in detail. For reference, identifiers in brackets link to threats from the "FIDO Security Reference" document [FIDOSpec].

T.InformationLeakage

Adverse action: An attacker may exploit information leaking from the TOE during its usage in order to disclose confidential keys stored on the TOE token or/and exchanged between the TOE and the relying party. The information leakage may be inherent in the normal operation or caused by the attacker.

Information leakage can occur through covered channels (side channels). Typical side channel attacks include measuring the power consumption (differential power/electromagnetic analysis) during operational use or to actively enforce leakage by fault injection (differential fault analysis).

Asset: seed, private key, MACKey

Threat agent: Attacker

T.PhysTamper

Adverse action: An attacker may perform physical probing of the TOE in order to disclose/reconstruct the keys. An attacker may physically modify the authenticator in order to alter its security functionality (hardware and software part).

Physical tampering may be focused directly on the disclosure or manipulation of key material Techniques commonly employed in IC failure analysis and IC reverse engineering efforts may be used. Before that, hardware security mechanisms and layout characteristics need to be identified. The modification may result in the deactivation of a security function. Changes of circuitry or data can be permanent or temporary.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.KeyCompromise (T-1.4.2)

Adverse action: An attacker succeeds in extracting a user's MACKey, private signing keys, or the seed.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.UserVerificationBy-Pass (T-1.4.3)

Adverse action: An attacker uses an authenticator or a private signing key either with or without being noticed by the legitimate user.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.SignatureAlgorithmAttack (T-1.4.8)

Adverse action: A cryptographic attack is discovered against the public key cryptography system used to sign data by the FIDO authenticator.

Asset: seed, private key

Threat agent: Attacker

T.AbuseFunctionality (T-1.4.9)

Adverse action: It might be possible for an attacker to abuse the authenticator functionality by sending commands with invalid parameters or invalid commands to the authenticator

Asset: seed, private key, MACKey

Threat agent: Attacker

T.RandomNumberPrediction (T-1.4.10)

Adverse action: It might be possible for an attacker to get access to information allowing the prediction of RNG data.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.FirmwareRollback (T-1.4.11)

Adverse action: An attacker might be able to install a previous and potentially buggy version of the firmware.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.Forgery (SG-11)

Adverse action: An attacker may attempt to modify intercepted communications in order to masquerade as the legitimate user and login to the relying party.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.Clone

Adverse action: An attacker clones a U2F authenticator and uses the cloned authenticator to login at the relying party as the legitimate user.

Asset: seed, private key, MACKey

Threat agent: Attacker

T.PrivacyViolation

Adverse action: An attacker is able to trace logins to two different accounts by information exchanged between the token and (one or more) relying parties to the same FIDO U2F authenticator, thus violating the user's privacy.

Asset: seed, private key, MACKey

Threat agent: Attacker

3.3 Organizational Security Policies

The TOE shall comply with the following Organizational Security Policies (OSP) as security rules, procedures, practices, or guidelines imposed by an organization upon its operations (see CC part 1, sec. 3.2 [CC1]).

P.UserConsent (SG-7)

The user is notified before a relationship to a new relying party is being established (requiring explicit consent).

3.4 Assumptions

The assumptions describe the security aspects of the environment in which the TOE will be used or is intended to be used

A.SeparationMechanism (SA-2)

Operating system privilege separation mechanisms relied up on by the software modules involved in a FIDO operation on the user device perform as advertised, e.g. boundaries between user and kernel mode, between user accounts, and between applications (where applicable) are securely enforced and security principals can be mutually, securely identifiable.

A.TrustworthAppServerChannel (SA-3)

Applications on user devices are able to establish secure channels that provide trustworthy server authentication, and confidentiality and integrity for messages (TLS).

A.TrustworthCE (SA-5)

The computing environment (CE) on the FIDO user device and the applications involved in a FIDO operation act as trustworthy agents of the user.

A.TrustworthRP (SA-7)

The computing resources at the relying party (RP) involved in processing a FIDO operation act as trustworthy agents of the relying party

3.5 Security Objectives for the TOE

This chapter describes the security objectives for the TOE and for the TOE environment.

OT.LeakageResistance

The TOE must provide protection against disclosure of TSF-data stored by the token (passive side channel attacks), including disclosure

- by measurement and analysis of the shape and amplitude of signals or the time between events found by measuring signals on the electromagnetic field, power consumption, clock, or I/O lines
- by physical manipulation of the TOE
- by exploiting software bugs and vulnerabilities

OT.TamperResistance

The TOE must provide protection of confidentiality and integrity of secret keys, TSF and the TOEs embedded software active or semi-active side channel attacks). The TOE must in particular implement counter-measures that prevent information extraction by

- power and emission analysis using the TOE's contact-based, contactless or other interfaces combined with active probing of the IC
- other types of side channel attacks, including those using tools employed in solid-state physics research and IC failure analysis,
- manipulation of the hardware and its security functionality, e.g. fault analysis
- reverse-engineering to understand the design and its properties and functionality.

OT.Prot_Abuse-Func (T-1.4.9)

The TOE must prevent that functions of the TOE, which may not be used in the TOE's operational phase, can be abused in order to manipulate or disclose TSF-data stored in the TOE or to manipulate (bypass, deactivate or modify) soft-coded security functionality of the TOE.

OT.HighLevelOfAssurance (SG-1)

Contribute to a FIDO authentication with a high level of assurance by employing a cryptographic protocol with high (cryptographic) strength.

OT.Unlinkability (SG-4)

Protect the protocol conversation such that any two relying parties cannot link the separate conversation to one user (i.e. be unlinkable).

OT.TrustworthyData

The FIDO Authenticator only accepts and processes sensitive data that are generated by itself and are linkable to the AppID of the relying party.

OT.AuthenticatorLeakResilience (SG-6)

Be resilient to leaks from other FIDO Authenticators. I.e., nothing that a particular FIDO Authenticator could possibly leak can help an attacker to impersonate any other user to any relying party.

OT.LimitedPII (SG-8)

Limit the amount of personal identifiable information (PII) exposed to the relying party to the absolute minimum.

OT.UserConsentForAllProcesses

For all processes the user invokes with the TOE, successful execution requires an immediately-prior demonstration of user presence.

3.6 Security Objectives for the Operational Environment

OE.RespectSecurityBoundaries (SG-15)

Ensure that registrations and key material as a shared system resource is appropriately protected according to the operating environment privilege boundaries in place on the FIDO user agent.

3.7 Security Objective Rationale

The following table 2 provides an overview for security objectives coverage (TOE and its environment). It shows that all threats and OSPs are addressed by the security objectives. It also shows that all assumptions are addressed by the security objectives for the TOE environment.

3.8 Rationale for Objectives for the TOE and the Operational Environment

	OT.LeakageResistance	OT.TamperResistance	OT.Prot_Abuse-Func	OT.HighLevelOfAssurance	OT.Unlinkability	OT.TrustworthyData	OT.AuthenticatorLeakResilince	OT.LimitedPII	OT.UserConsentForAllProcesss	OE.RespectSecurityBoundaries
T.InformationLeakage	x	x	x	x						
T.PhysTampering	x	x	x	x						
T.KeyCompromise	x	x		x			x			
T.UserVerificationByPass	x	x		x			x			
T.SignatureAlgorithmAttack				x						
T.AbuseFunctionality			x				x			
T.RandomNumberPrediction				x	x		x			
T.FirmwareRollback			x							
T.Forgery				x		x	x			
T.Clone	x	x	x				x			
T.PrivacyViolation					x			x		
P.UserConsent									x	x
A.TrustworthCE										x
A.SeparationMechanism										x
A.TrustworthAppServerChannel										x
A.TrustworthRP										x

Table 2: Security Objective Rationale

3.9 Security objective sufficiency

Countering of threats by security objectives:

The threats **T.InformationLeakage** and **T.PhysTampering** are protected by the directly related security objectives *OT.LeakageResistance* and *OT.TamperResistance* as well as *OT.Prot_Abuse-Func* and *OT.HighLevelOfAssurance*.

T.KeyCompromise via publicly known data produced by the TOE could lead to impersonating the user with a cloned authenticator. *OT.HighLevelOfAssurance* counters this threat by implementing cryptographically secure generation of the key pair. *OT.LeakageResistance* and *OT.TamperResistance* prevent leakage of the key. *OT.AuthenticatorLeakResilience* prevents that the authenticator himself leaks

T.UserVerificationByPass: Impersonating the user is prevented by *OT.HighLevelOfAssurance*. Readout of the MACKey and/or private keys is prevented by *OT.LeakageResistance* and *OT.TamperResistance*. Leakage of the MACkey and/or private keys from the authenticator is prevented by *OT.AuthenticatorLeakResilience.*

T.SignatureAlgorithmAttack: *OT.HighLevelOfAssurance* aims for the use of proper cryptographic security and prevents the use of insecure signature algorithms.

T.AbuseFunctionality: The security objective *OT.Prot_Abuse-Func* and *OT.AuthenticatorLeakResilience* ensures that the usage of functions having not to be used in the operational phase is effectively prevented.

T.RandomNumberPrediction is covered by *OT.HighLevelOfAssurance. OT.AuthenticatorLeakResilience, and OT.Unlinkability* prevent to gain any information w.r.t. generated random numbers.

T.FirmwareRollback is directly addressed by *OT.Prot_Abuse-Func.*

T.Forgery is prevented by secure cryptography (*OT. HighLevelOfAssurance),* leak resistance (*OT.AuthenticatorLeakResilience)* and is also covered by *OT.TrustworthyData.*

T.Clone: To prevent this threat any kind of leakage w.r.t information stored on the TOE must be minimized. *OT.LeakageResistance, OT.TamperResistance, OT.AuthenticatorLeakResilience,* and *OT.Prot_Abuse-Func* covers this threat.

T.PrivacyViolation is prevented by *OT.Unlinkability* and *OT.LimitedPII.*

Enforcement of Organizational Security Policies by security objectives:

P.UserContent *OE.RespectSecurityBoundaries and OT. UserConsentForAllProcesses* provides for the guarantee that the user is informed before establishing a relationship between user and relying party. *OE.RespectSecurityBoundaries e*nsures that key material (e.g. TLS keys) and registrations handled by the operating environment cannot be easily manipulated. If the privileged boundaries of the operating environment cannot be maintained a user could be fooled into registering at a new relying party while thinking she is merely providing user presence to authenticate to a known relying party. *OT.UserConsentForAllProcesses* ensures that the authenticator only act if user presence has been verified prior to execution.

Upholding assumptions by environment objectives:

A.TrustworthCE, A.SepartionMechanism, A.TrustworthAppServerChannel and **A.Trustworth RP** are covered by *OE.RespectSecurityBoundaries.*

4 Extended Components Definition

This protection profile uses components defined as extensions to CC part 2 [CC2].

4.1 Definition of the Family FCS_RNG

To define the IT security functional requirements of the TOE a family (FCS_RNG) of the class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes. The component FCS_RNG.1 is not limited to generation of cryptographic keys unlike the component FCS_CKM.1. The similar component FIA_SOS.2 is intended for non-cryptographic use.

The family 'Generation of random numbers (FCS_RNG)' is specified as follows:

FCS_RNG Generation of random numbers

Family behaviour

This family defines quality requirements for the generation of random numbers intended to be used for cryptographic purposes.

Component levelling:

FCS_RNG.1 Generation of random numbers requires that the random number generator implements defined security capabilities and that the random numbers meet a defined quality metric.

Management:	FCS_RNG.1
	There are no management activities foreseen.
Audit:	FCS_RNG.1
	There are no actions defined to be auditable.

FCS_RNG.1 Random number generation

Hierarchical to:	No other components.
Dependencies:	No dependencies
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 random numbers that meet [assignment: *a defined quality metric*].

4.2 Definition of the Family FPT_EMS

The family FPT_EMS (TOE Emanation) of the class FPT (Protection of the TSF) is defined here to describe the IT security functional requirements of the TOE. The TOE shall prevent attacks against secret data stored in and used by the TOE where the attack is based on external observable physical phenomena of the TOE. Examples of such attacks are evaluation of TOE's electromagnetic radiation, simple power analysis (SPA), differential power analysis (DPA), timing attacks, etc. This family describes the functional requirements for the limitation of intelligible emanations being not directly addressed by any other component of CC part 2 [CC2].

The family 'TOE Emanation (FPT_EMS)' is specified as follows:

FPT_EMS **TOE emanation**

Family behaviour

This family defines requirements to mitigate intelligible emanations.

Component levelling:

FPT_EMS: TOE emanation
1

FPT_EMS.1 TOE emanation has two constituents:

FPT_EMS.1.1 Limit of Emissions requires to not emit intelligible emissions enabling access to TSF data or user data.

FPT_EMS.1.2 Interface Emanation requires to not emit interface emanation enabling access to TSF data or user data.

Management: FPT_EMS.1

There are no management activities foreseen.

Audit: FPT_EMS.1

There are no actions defined to be auditable.

FPT_EMS.1 **TOE Emanation**

Hierarchical to: No other components.

Dependencies: No dependencies

FPT_EMS.1.1 The TOE shall not emit [assignment: *types of emissions*] in excess of [assignment: specified limits] enabling access to [assignment: *list of types of TSF data*] and [assignment: *list of types of user data*].

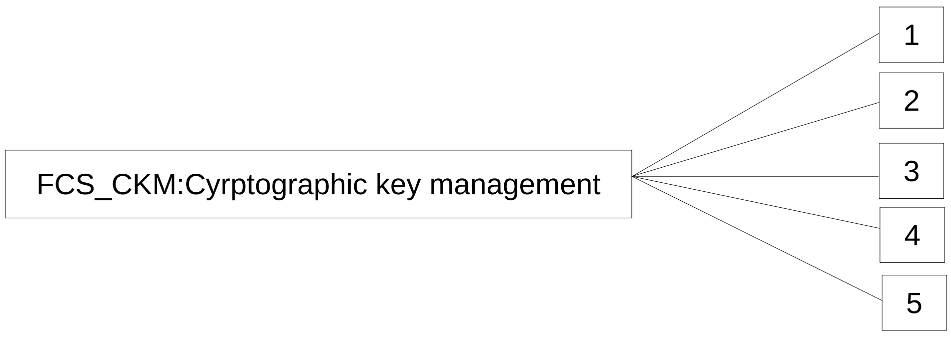
FPT_EMS.1.2 The TSF shall ensure [assignment: *type of users*] are unable to use the following interface [assignment: *type of connection*] to gain access to [assignment: list of types of TSF data] and [assignment: *list of types of user data*].

4.3 Definition of the Component FCS_CKM.5

This chapter describes functional requirements for key derivation as process by which one or more keys are calculated from either a pre-shared key or a shared secret and other information. The component is part of the family FCS_CKM of the class FCS. The component FCS_CKM.5 has been specified as follows:

FCS_CKM **Cryptographic key management**

Component levelling:



Management:	FCS_CKM.5
	There are no management activities foreseen.
Audit:	FCS_CKM.5
	There are no actions defined to be auditable.

FCS_CKM.5 **Cryptographic key derivation**

Hierarchical to:	No other components.
Dependencies:	[FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation] FCS_CKM.4 Cryptographic key destruction
FCS_CKM.5.1	The TSF shall derive cryptographic keys [assignment: *key type*] from [assignment: *input parameters*] in accordance with a specified cryptographic key derivation algorithm [assignment: *cryptographic key derivation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

5 Security Requirements

This part defines detailed security requirements that shall be satisfied by the TOE. The statement of TOE security requirements shall define the functional and assurance security requirements that the TOE must satisfy in order to meet the security objectives for the TOE.

Common Criteria allows several operations to be performed on security requirements on the component level: *refinement, selection, assignment* and *iteration*, cf. sec. 8.1 of [CC1]. Each of these operations is used in this PP.

The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinements of security requirements are denoted in such a way that added words are in **bold text** and removed words are ~~crossed out~~.

The **selection** operation is used to select one or more options provided by CC in stating a requirement. Selections that have been made by the PP author are denoted as underlined text. Selections to be filled in by the ST author appear in square brackets with an indication that a selection has to be made, [*selection: choose from these options*], and are italicised.

The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password. Assignments that have been made by the PP author are denoted as underlined text. Assignments to be filled in by the ST author appear in square brackets with an indication that an assignment has to be made [*assignment: choose your assignment*], and are italicized. In some cases the assignment made by the PP authors defines a selection to be performed by the ST author. Then this text is *underlined and italic*.

The **iteration** operation is used when a component is repeated with varying operations. Iteration is denoted by showing a slash "/", and the iteration indicator after the component identifier. For the sake of better readability, the iteration operation may also be applied to a non-repeated single component in order to indicate that such component belongs to a certain functional cluster. In such a case, the iteration operation is applied to only one single component.

5.1 Security Functional Requirements

5.1.1 Class FCS Cryptographic Support

FCS_CKM.1 **Cryptographic key generation**

Hierarchical to:	No other components.
Dependencies:	[FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation]: Fulfilled by FCS_COP.1 FCS_CKM.4 Cryptographic key destruction: Fulfilled by FCS_CKM.4
FCS_CKM.1.1	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment: *cryptographic key generation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards compliant to [FIDOCrypt]*][1].

Application note 1: The SFR above applies to all cryptographic key generation methods implemented on the TOE.

FCS_CKM.4 **Cryptographic key destruction**

1 [assignment: *list of standards*]

Hierarchical to:	No other components.
Dependencies:	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]: Fulfilled by FCS_CKM.1
FCS_CKM.4.1	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [assignment: *cryptographic key destruction method*] that meets the following: [assignment: *list of standards*].

FCS_COP.1 Cryptographic operation

Hierarchical to:	No other components.
Dependencies:	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]: Fulfilled by FCS_CKM.1 FCS_CKM.4 Cryptographic key destruction: : Fulfilled by FCS_CKM.4
FCS_COP.1.1	The TSF shall perform [assignment: *list of cryptographic operations*] in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards compliant to* [FIDOCrypt]][2].

Application note 2: The above SFR applies to all cryptographic operations implemented on the TOE.

FCS_COP.1/MAC Cryptographic operation – MAC

Hierarchical to:	No other components.
Dependencies:	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]: Fulfilled by FCS_CKM.1 FCS_CKM.4 Cryptographic key destruction: Fulfilled by FCS_CKM.4
FCS_COP.1.1	The TSF shall perform message authentication code[3] in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards compliant to* [FIDOCrypt]][4].

FCS_RNG.1/nonce Random number generation – RNG for seed/nonce/MACKey generation

Hierarchical to:	No other components.
Dependencies:	No dependencies.
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 random numbers that meet [assignment: *a defined quality metric*].

Application note 3: The above SFR requires the TOE to generate random numbers used for seed, nonce and MACKey generation.

FCS_CKM.5/U2F Cryptographic key derivation for U2F private keys

2 [assignment: *list of standards*]
3 [assignment: *list of cryptographic operations*]
4 [assignment: *list of standards*]

Hierarchical to:	No other components.
Dependencies:	[FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation]: Fulfilled by FCS_COP.1
	FCS_CKM.4 Cryptographic key destruction: Fulfilled by FCS_CKM.4
FCS_CKM.5.1	The TSF shall derive cryptographic keys U2F private keys[5] from seed as the confidential key, and FIDO AppID and nonce[6] in accordance with a specified cryptographic key derivation [assignment: c*ryptographic key derivation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards compliant to* [FIDOCrypt]][7].

Application note 4: The SFR implements the process of generating random bits in the FIDO authentication process. The generated random bits of the FIDO compliant KDF are used to derive FIDO private keys from the global seed.

5.1.2 Class FDP User data protection

FDP_IFC.1/Initialisation Subset information flow control - Initialisation

Hierarchical to:	No other components.
Dependencies:	FDP_IFF.1 Simple security attributes: Fulfilled by FDP_IFF.1/ Initialisation
FDP_IFC.1.1	The TSF shall enforce the information flow control FDP_IFF.1/Initialisation SFP[8] on [assignment: *list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

FDP_IFC.1/Reset Subset information flow control - Reset

Hierarchical to:	No other components.
Dependencies:	FDP_IFF.1 Simple security attributes: Fulfilled by FDP_IFF.1/Reset
FDP_IFC.1.1	The TSF shall enforce the information flow control FDP_IFF.1/Reset SFP[9] on [assignment: *list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

FDP_IFC.1/Registration Subset information flow control - Registration

Hierarchical to:	No other components.
Dependencies:	FDP_IFF.1 Simple security attributes: Fulfilled by FDP_IFF.1/Registration
FDP_IFC.1.1	The TSF shall enforce the information flow control FDP_IFF.1/Registration SFP[10] on [assignment: *list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

FDP_IFC.1/Authentication Subset information flow control - Authentication

5 [assignment: key type]
6 [assignment: input parameters]
7 [assignment: list of standards]
8 [assignment: *information flow control SFP*]
9 [assignment: *information flow control SFP*]
10 [assignment: *information flow control SFP*]

Hierarchical to:	No other components.
Dependencies:	FDP_IFF.1 Simple security attributes: Fulfilled by FDP_IFF.1/Authentication
FDP_IFC.1.1	The TSF shall enforce the information flow control FDP_IFF.1/Authentication SFP[11] on [assignment: *list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

FDP_IFF.1/Initialisation Simple Security Attributes - Initialisation

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control: Fulfilled by FDP_IFC.1/Inititialisation

FMT_MSA.3 Static attribute initialisation : Fulfilled by FMT_MSA.3/Initialisation

FDP_IFF.1.1 The TSF shall enforce the initialisation process SFP[12] based on the following types of subject and information security attributes:

1) Subjects:
   a) end-user
   b) authenticator
2) information:
   a) seed
   b) MACKey
3) security attributes:
   a) user_presence: true, false
   b) security_state of the authenticator: delivery_state, ready_for_use[13].

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

1) The end-user initialises the authenticator.
2) After initialisation the authenticator indicates its security status to the end-user[14].

FDP_IFF.1.3 The TSF shall enforce the following rules in their specific order:

```
1) user_presence = check_user_presence()
   a) user_presence == false → continue with 6)
   b) user_presence == true → continue with 2)
2) security_state = check_security_state()
   a) security_state == ready_for_use → continue with 6)
   b) security_state == delivery_state → continue with 3)
3) seed = RNG() (cf. FCS_RNG.1/nonce)
4) MACKey = RNG() (cf. FCS_RNG.1/nonce)
5) set security_state = ready_for_use
```

11 [assignment: *information flow control SFP*]
12 [assignment: *information flow control SFP*]
13 [assignment: *list of subjects and information controlled under the indicated SFP, and for each, the security attributes*]
14 [assignment: *for each operation, the security attribute-based relationship that must hold between subject and information security attributes*]

6) manage_user_presence() [cf. FIA_UAU.2 resp. FIA_UAU.6][15]

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: none[16].

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules:

1) The MACKey never leaves the TOE

2) The seed never leaves the TOE[17].

Application note 5: The function manage_user_presence() (re)sets the user presence value according to the requirements defined in FIA_UAU.2 and FIA_UAU.6

Application note 6: FDP_IFF.1.4 is left open here and in subsequent SFRs FDP_IFF.* and is intended to be used when there is a need to cater to specific implementational details. Assignments here must not circumvent the explicit rules stated in FDP_IFF.1.3 and must not weaken the security requirements. The ST-Writer should assign "none" here if no such specific implementational detail has to be considered.

FDP_IFF.1/Reset Simple Security Attributes - Reset

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control: Fulfilled by FDP_IFC.1/Reset

FMT_MSA.3 Static attribute initialisation : Fulfilled by FMT_MSA.3/Reset

FDP_IFF.1.1 The TSF shall enforce the reset process SFP[18] based on the following types of subject and information security attributes:

1) Subjects:
  a) end-user
  b) authenticator

2) information:
  a) seed
  b) MACKey

3) security attributes:
  a) user_presence: true, false
  b) security_state: delivery_state, ready_for_use[19].

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

1) The end-user resets the authenticator

2) After the reset process the authenticator indicates its security_state to the end-user[20].

FDP_IFF.1.3 The TSF shall enforce the following rules in their specific order:

15 [assignment: *additional information flow control SFP rules*]
16 [assignment: *rules, based on security attributes, that explicitly authorise information flows*]
17 [assignment: *rules, based on security attributes, that explicitly deny information flows*]
18 [assignment: *information flow control SFP*]
19 [assignment: *list of subjects and information controlled under the indicated SFP, and for each, the security attributes*]
20 [assignment: *for each operation, the security attribute-based relationship that must hold between subject and information security attributes]*

1) user_presence = check_user_presence()

   a) user_presence == false → continue with 5)

   b) user_presence == true → continue with 2)

2) security_state = check_security_state()

   a) security_state = delivery_state → continue with 5)

   a) security_state = ready_for_use → continue with 3)

2) set security_state = delivery_state

3) secure_erase(seed)

4) secure_erase(MACKey)

5) manage_user_presence() [cf. FIA_UAU.2 resp. FIA_UAU.6][21].

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: none[22].

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules:

1) The MACKey never leaves the TOE

2) The seed never leaves the TOE[23].

FDP_IFF.1/Registration Simple Security Attributes - Registration

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control: Fulfilled by FDP_IFC.1/Registration

FMT_MSA.3 Static attribute initialisation : Fulfilled by FMT_MSA.3/Registration

FDP_IFF.1.1 The TSF shall enforce the registration process SFP[24] based on the following types of subject and information security attributes:

1) Subjects:

   a) authenticator

   b) relying party

   c) end-user

2) information:

   a) AppID

   b) nonce

   c) MACKey

   d) MAC(MACKey, [nonce|AppID])

   e) private key

   f) public key

   g) seed

3) security attributes:

21 [assignment: *additional information flow control SFP rules*]
22 [assignment: *rules, based on security attributes, that explicitly authorise information flows*]
23 [assignment: *rules, based on security attributes, that explicitly deny information flows*]
24 [assignment: *information flow control SFP*]

a) user_presence: true, false[25].

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

1) The end-user starts to register the authenticator at the relying party

2) The authenticator requests AppID from RP

3) The authenticator sends nonce, MAC(MACKey, [nonce|AppID]) and public key[26].

FDP_IFF.1.3 The TSF shall enforce the following rules in their specific order:

1) relying party supplies AppID

2) user_presence = check_user_presence()
   a) user_presence == false → continue with 8)
   b) user_presence == true → continue with 3)

3) security_state = check_security_state()
   a) security_state = delivery_state → continue with 8)
   a) security_state = ready_for_use → continue with 4)

4) nonce = RNG() (cf. FCS_RNG.1/nonce)

5) PrivK = KDF(seed, AppID, nonce) (cf. FCS_CKM.5/U2F)

6) PubK = genKeyPair(PrivK)

7) delete(PubK) and secure_erase(PrivK)

8) manage_user_presence() [cf. FIA_UAU.2 resp. FIA_UAU.6][27].

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: none[28].

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules:

1) The MACKey never leaves the TOE

2) The seed never leaves the TOE

3) Private keys never leave the TOE[29].

FDP_IFF.1/Authentication Simple Security Attributes - Authentication

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control: Fulfilled by FDP_IFC.1/Authentication

FMT_MSA.3 Static attribute initialisation : Fulfilled by FMT_MSA.3/Authentication

FDP_IFF.1.1 The TSF shall enforce the authentication process SFP[30] based on the following types of subject and information security attributes:

25 [assignment: *list of subjects and information controlled under the indicated SFP, and for each, the security attributes*]

26 [assignment: *for each operation, the security attribute-based relationship that must hold between subject and information security attributes]*

27 [assignment: *additional information flow control SFP rules*]

28 [assignment: *rules, based on security attributes, that explicitly authorise information flows*]

29 [assignment: *rules, based on security attributes, that explicitly deny information flows*]

30 [assignment: *information flow control SFP*]

1) Subjects:
   a) authenticator
   b) relying party
   c) end-user
2) information:
   a) AppID
   b) nonce
   c) MACKey
   d) MAC(MACKey, [nonce|AppID])
   e) private key
   f) seed
3) security attributes:
   a) user_presence: true, false
   b) MAC verification status: true, false[31].

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

1) The end-user starts authentication at the RP
2) The authenticator requests AppID, nonce, MAC from RP
3) The authenticator sends the signature to the RP[32].

FDP_IFF.1.3 The TSF shall enforce the following rules in their specific order:

1) security_state = check_security_state()
   a) security_state = delivery_state → continue with 7)
   a) security_state = ready_for_use → continue with 2)
2) verify MAC
   a) verify(MAC) == false → continue with 7)
   b) verify(MAC) == true → continue with 3)
3) user_presence = check_user_presence()
   a) user_presence == false → continue with 7)
   b) user_presence == true → continue with 4)
4) PrivK = KDF(seed, AppID, nonce) (cf. FCS_CKM.5/U2F)
5) sig = sign(PrivK, AppID | challenge)
6) secure_erase(PrivK)
7) manage_user_presence() [cf. FIA_UAU.2 resp. FIA_UAU.6][33].

31 [assignment: *list of subjects and information controlled under the indicated SFP, and for each, the security attributes*]
32 [assignment: *for each operation, the security attribute-based relationship that must hold between subject and information security attributes*]
33 [assignment: *additional information flow control SFP rules*]

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: [assignment: *rules, based on security attributes, that explicitly authorise information flows that model the "check-only" command 0x07* [FIDOSpec]].

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules:

1) The MACKey never leaves the TOE

2) The seed never leaves the TOE

3) Private keys never leave the TOE[34].

FDP_SDI.1 Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1 The TSF shall monitor user data stored in containers controlled by the TSF for [assignment: *integrity errors*] on all objects, based on the following attributes: [assignment: *user data attributes*].

Application note 7: Assignment in the above SFR should be filled by the ST-writer and very much depend on the specific implementation. For example, integrity errors could be manipulated errors in RAM during execution.

5.2 Class FIA Identification and Authentication

FIA_UAU.2 User authentication before any action

Hierarchical to: FIA_UAU.1 Timing of authentication

Dependencies: FIA_UID.1 Timing of identification: not fulfilled, but justified: Identification is not needed for user presence

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

Application note 8: Proof of presence means e.g. pressing some button or placing the token into the proximity range of an NFC-enabled device.

FIA_UAU.6 Re-authenticating

Hierarchical to: No other components.

Dependencies: No dependencies.

FIA_UAU.6.1 The TSF shall **require to** re-authenticate the user **by the user demonstrating proof of presence again** under the conditions [assignment: list of conditions under which re-authentication is required].

34 [assignment: *rules, based on security attributes, that explicitly deny information flows*]

Application note 9: Possible conditions are: 1) the elapsed time since the previous proof of presence demonstration has surpassed a limit (specified in the condition) 2) an event (specified in the condition) has occurred. The list of conditions must not be empty.

5.2.1 Class FMT Security Management

FMT_SMF.1 Security management functions

Hierarchical to:	No other components.
Dependencies:	No dependencies.
FMT_SMF.1.1	The TSF shall be capable of performing the following management functions: (1) <u>creation and destruction of the MACKey/seed</u>, (2) <u>reset the authenticator</u>, (3) <u>[assignment: *list of other security management functions to be provided by the TSF*]</u>[35]

Application note 10: The list of other security management functions to be provided by the TSF may be empty (i.e. assignment "none").

FMT_SMR.1 Security roles

Hierarchical to:	No other components.
Dependencies:	FIA_UID.1 Timing of identification: not fulfilled, but justified: No user is identified, only user presence is checked
FMT_SMR.1.1	The TSF shall maintain the roles [assignment: *the authorised identified roles*].
FMT_SMR.1.2	The TSF shall be able to associate users with roles.

FMT_MTD.1 Management of TSF data

Hierarchical to:	No other components.
Dependencies:	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
	FMT_SMF.1 Specification of Management Functions: Fulfilled by FMT_SMF.1
FMT_MTD.1.1	The TSF shall restrict the ability to [selection: *change_default, query, modify, delete, clear*, [assignment: *other operations*]] the [assignment: *list of TSF data*] to the [assignment: *the authorised identified roles*].

FMT_MSA.1/Initialisation Management of security attributes - Initialisation

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]: Fulfilled by FDP_IFC.1/Initialisation
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1

35 [assignment: *list of security management functions to be provided by the TSF*]

	FMT_SMF.1 Specification of Management Functions: Fulfilled by FMT_SMF.1
FMT_MSA.1.1	The TSF shall enforce the information flow control FDP_IFC.1/Initialisation SFP[36] to restrict the ability to query[37] the security attribute security_state[38] to [assignment: *the authorised identified roles*].

FMT_MSA.3/Initialisation Static attribute initialisation - Initialisation

Hierarchical to:	No other components.
Dependencies:	FMT_MSA.1 Management of security attributes: Fulfilled by FMT_MSA.1/Initialisation
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
FMT_MSA.3.1	The TSF shall enforce the information flow control FDP_IFF.1/Initialisation SFP[39] to provide restrictive[40] default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2	The TSF shall allow the [assignment: *the authorised identified roles*] to specify alternative initial values to override the default values when an object or information is created.

*Application note 11: R*estrictive means: user_presence = false and security_state = delivery_state

FMT_MSA.1/Reset Management of security attributes - Reset

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]: fulfilled by FDP_IFC.1/Reset
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
	FMT_SMF.1 Specification of Management Functions: Fulfilled by FMT_SMF.1
FMT_MSA.1.1	The TSF shall enforce the information flow control FDP_IFC.1/Reset SFP[41] to restrict the ability to query[42] the security attribute user_presence, security_state[43] to [assignment: *the authorised identified roles*].

FMT_MSA.3/Reset Static attribute initialisation - Reset

Hierarchical to:	No other components.
Dependencies:	FMT_MSA.1 Management of security attributes: Fulfilled by FMT_MSA.1/Reset
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
FMT_MSA.3.1	The TSF shall enforce the information flow control FDP_IFF.1/Reset SFP[44] to provide

36 [assignment: *access control SFP(s), information flow control SFP(s)*]
37 [selection: *change_default, query, modify, delete*, [assignment: *other operations*]]
38 [assignment: *list of security attributes*]
39 [assignment: *access control SFP, information flow control SFP*]
40 [selection, choose one of: *restrictive, permissive,* [assignment: *other property*]]
41 [assignment: *access control SFP(s), information flow control SFP(s)*]
42 [selection: *change_default, query, modify, delete*, [assignment: *other operations*]]
43 [assignment: *list of security attributes*]

restrictive[45] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the [assignment: *the authorised identified roles*] to specify alternative initial values to override the default values when an object or information is created.

*Application note 12: R*estrictive means: user_presence = false and security_state = ready_for_use

FMT_MSA.1/Registration Management of security attributes - Registration

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]: fulfilled by FDP_IFC.1/Registration
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
	FMT_SMF.1 Specification of Management Functions: Fulfilled by FMT_SMF.1
FMT_MSA.1.1	The TSF shall enforce the information flow control FDP_IFC.1/Registration SFP[46] to restrict the ability to query[47] the security attribute user_presence/security_state[48] to [assignment: *the authorised identified roles*].

FMT_MSA.3/Registration Static attribute initialisation - Registration

Hierarchical to:	No other components.
Dependencies:	FMT_MSA.1 Management of security attributes: Fulfilled by FMT_MSA.1/Registration
	FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1
FMT_MSA.3.1	The TSF shall enforce the information flow control FDP_IFF.1/Registration SFP[49] to provide restrictive[50] default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2	The TSF shall allow the [assignment: *the authorised identified roles*] to specify alternative initial values to override the default values when an object or information is created.

*Application note 13: R*estrictive means: user_presence = false

FMT_MSA.1/Authentication Management of security attributes - Authentication

Hierarchical to:	No other components.
Dependencies:	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]:

44 [assignment: *access control SFP, information flow control SFP*]
45 [selection, choose one of: *restrictive, permissive,* [assignment: *other property*]]
46 [assignment: *access control SFP(s), information flow control SFP(s)*]
47 [selection: *change_default, query, modify, delete*, [assignment: *other operations*]]
48 [assignment: *list of security attributes*]
49 [assignment: *access control SFP, information flow control SFP*]
50 [selection, choose one of: *restrictive, permissive,* [assignment: *other property*]]

fulfilled by FDP_IFC.1/Authentication

FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1

FMT_SMF.1 Specification of Management Functions: Fulfilled by FMT_SMF.1

FMT_MSA.1.1 The TSF shall enforce the information flow control FDP_IFC.1/Authentication SFP[51] to restrict the ability to query[52] the security attribute user_presence/MAC verification status[53] to [assignment: *the authorised identified roles*].

FMT_MSA.3/ Authentication Static attribute initialisation - Authentication

Hierarchical to: No other components.

Dependencies: FMT_MSA.1 Management of security attributes: Fulfilled by FMT_MSA.1/Authentication

FMT_SMR.1 Security roles: Fulfilled by FMT_SMR.1

FMT_MSA.3.1 The TSF shall enforce the information flow control FDP_IFF.1/Authentication SFP[54] to provide restrictive[55] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the [assignment: *the authorised identified roles*] to specify alternative initial values to override the default values when an object or information is created.

*Application note 14: R*estrictive means: user_presence = false and MAC verification status = false

5.2.2 Class FPR Privacy

FPR_ANO.1 Anonymity

Hierarchical to: No other components.

Dependencies: No dependencies.

FPR_ANO.1.1 The TSF shall ensure that [assignment: *set of users and/or subjects*] are unable to determine the real user name bound to [assignment: *list of subjects and/or operations and/or objects*].

5.2.3 Class FPT Protection of the TSF

FPT_EMS.1 TOE Emanation

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_EMS.1.1 The TOE shall not emit [assignment: *types of emissions*] in excess of [assignment:

51 [assignment: *access control SFP(s), information flow control SFP(s)*]
52 [selection: *change_default, query, modify, delete*, [assignment: *other operations*]]
53 [assignment: *list of security attributes*]
54 [assignment: *access control SFP, information flow control SFP*]
55 [selection, choose one of: *restrictive, permissive*, [assignment: *other property*]]

	specified limits] enabling access to 1. the seed 2. the MACKey 3. private keys[56] and [assignment: *list of types of user data*].
FPT_EMS.1.2	The TSF shall ensure any users[57] are unable to use the following interface authenticator's contactless/contact-based interface and circuit contacts[58] to gain access to 1. the seed 2. the MACKey 3. private keys[59] and [assignment: *list of types of user data*].

FPT_PHP.3 Resistance to physical attack

Hierarchical to:	No other components.
Dependencies:	No dependencies.
FPT_PHP.3.1	The TSF shall resist [assignment: *physical tampering scenarios*] to the [assignment: *list of TSF devices/elements*] by responding automatically such that the SFRs are always enforced.

FPT_TST.1 TSF testing

Hierarchical to:	No other components.
Dependencies:	No dependencies.
FPT_TST.1.1	The TSF shall run a suite of self tests [selection: *during initial start-up, periodically during normal operation, at the request of the authorised user, at the conditions* [assignment: *conditions under which self test should occur*]] to demonstrate the correct operation of [selection: [assignment: *parts of TSF*], *the TSF*].
FPT_TST.1.2	The TSF shall provide authorised users with the capability to verify the integrity of [selection: [assignment: *parts of TSF data]*, *TSF data*].
FPT_TST.1.3	The TSF shall provide authorised users with the capability to verify the integrity of [selection: [assignment: *parts of TSF*], *TSF*].

Application note 15: Authorised user means user has demonstrated his proof of presence.

56 [assignment: *list of types of* TSF *data*]
57 [assignment: *type of users*]
58 [assignment: *type of connection*]
59 [assignment: *list of types of TSF data*]

5.3 Security Assurance Requirements

The assurance requirements for the evaluation of the TOE, its development and operating environment are to choose as the predefined assurance package EAL4 augmented by the following components:

- AVA_VAN.5 (Advanced methodical vulnerability analysis).

5.4 Security Requirements Rationale

5.4.1 Security Functional Requirements Rationale

The following table 3 provides an overview for the coverage of the security functional requirements, and also gives evidence for sufficiency and necessity of the chosen SFRs.

OT.LeakageResistance: The SFR's FPT_TST.1, FPT_EMS.1, FMT_SMF.1 and FMT_MTD.1 supports OT.LeakageResistance.

OT.TamperResistance: FPT_PHP.3, FPT_TST.1 supports OT.TamperResistance. The SFR FDP_SDI.1 is used to detect integrity errors, so this SFR also supports the objective OT.TamperResistance.

OT.HighLevelOfAssurance: FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FCS_COP.1/MAC, FCS_RNG.1/nonce, FCS_CKM.5/U2F are concerned with cryptographic operations and key generation. They support the objective OT.HighLevelOfAssurance as well as the managment SFR's FMT_MTD.1, FMT_SMF.1.

OT.LimitedPII, OT.Unlinkability: Because for each relying party and AppID a new nonce and key pair is generated it is not possible to link between two relying parties or user accounts and limit the amount of of personal identifiable information. OT.LimitedPII and OT.Unlinkability is covered by FCS_COP.1/MAC, FCS_RNG.1/nonce, FCS_CKM.5/U2F and FPR_ANO.1. The management SFR's FMT_SMF.1, FMT_MTD.1 supports OT.Unlinkability.

OT.TrustworthyData: FCS_COP.1/MAC, FCS_RNG.1/nonce, FCS_CKM.5/U2F supports OT.TrustworthyData.

OT.AuthenticatorLeakResilience: FCS_RNG.1/nonce, all instances of FDP_IFC.1, FDP_IFF.1, the SFRs FIA_UAU.2, FIA_UAU.6, FMT_SMR.1, FMT_SMF.1, all instances of FMT_MSA.1, all instances of FMT_MSA.3, and FMT_MTD.1 supports OT.AuthenticatorLeakResilience.

OT.Prot_Abuse-Func: The TSF self test SFR FPT_TST.1 supports OT.Prot_Abuse-Func.

	OT.LeakageResistance	OT.TamperResistance	OT.HighLevelOfAssurance	OT.Unlinkability	OT.TrustworthyData	OT.AuthenticatorLeakResilience	OT.Prot_Abuse-Func	OT.LimitedPII
Class FCS								
FCS_CKM.1			x					
FCS_CKM.4			x					
FCS_COP.1			x					
FCS_COP.1/MAC			x	x	x			x
FCS_RNG.1/nonce			x	x	x	x		x
FCS_CKM.5/U2F			x	x	x			x
Class FDP								
FDP_IFC.1/Initialisation						x		
FDP_IFC.1/Reset						x		
FDP_IFC.1/Registration						x		
FDP_IFC.1/Authentication						x		
FDP_IFF.1/Initialisation						x		
FDP_IFF.1/Reset						x		
FDP_IFF.1/Registration						x		
FDP_IFF.1/Authentication						x		
FDP_SDI.1		x						
Class FIA								
FIA_UAU.2						x		
FIA_UAU.6						x		
Class FMT								
FMT_SMR.1						x		
FMT_SMF.1	x		x	x		x		
FMT_MSA.1/Initialisation						x		
FMT_MSA.1/Reset						x		

	OT.LeakageResistance	OT.TamperResistance	OT.HighLevelOfAssurance	OT.Unlinkability	OT.TrustworthyData	OT.AuthenticatorLeakResilience	OT.Prot_Abuse-Func	OT.LimitedPII
FMT_MSA.1/Registration						x		
FMT_MSA.1/Authentication						x		
FMT_MSA.3/Initialisation						x		
FMT_MSA.3/Reset						x		
FMT_MSA.3/Registration						x		
FMT_MSA.3/Authentication						x		
FMT_MTD.1	x		x	x		x		
Class FPR								
FPR_ANO.1				x				x
Class FPT								
FPT_EMS.1	x							
FPT_PHP.3		x						
FPT_TST.1	x	x					x	

Table 3: Security Functional Requirements Rationale

5.4.2 Rationale for SFR's Dependencies

The dependency analysis for the security functional requirements shows that the basis for mutual support and internal consistency between all defined functional requirements is satisfied. All dependencies between the chosen functional components are analyzed, and non-dissolved dependencies are appropriately explained.

The dependency analysis has directly been made within the description of each SFR in the sections above. All dependencies being expected are either fulfilled, or their non-fulfillment is justified.

5.4.3 Security Assurance Requirements Rationale

The current assurance package was chosen based on the predefined assurance package EAL4. This package permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level, at which it is likely to retrofit to an existing product line in an economically feasible way. EAL4 is applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security specific engineering costs.

The selection of the component AVA_VAN.5 provides a higher assurance than the predefined EAL4 package, namely requiring a vulnerability analysis to assess the resistance to penetration attacks performed by an attacker possessing a high attack potential.

The set of assurance requirements being part of EAL4 fulfills all dependencies a priori. The augmentation of EAL4 chosen comprises the following assurance components: AVA_VAN.5.

For these additional assurance component, all dependencies are met or exceeded in the EAL4 assurance package. Below we list only those assurance requirements that are additional to EAL4.

AVA_VAN.5	
Dependencies:	ADV_ARC.1, ADV_FSP.4, ADV_TDS.3, ADV_IMP.1, AGD_OPE.1, AGD_PRE.1, ATE_DPT.1
fulfilled by	ADV_ARC.1, ADV_FSP.4, ADV_TDS.3, ADV_IMP.1, AGD_OPE.1, AGD_PRE.1, ATE_DPT.2

5.4.4 Security Requirements – Internal Consistency

The following part of the security requirements rationale shows that the set of security requirements for the TOE consisting of the security functional requirements (SFRs) and the security assurance requirements (SARs) are internally consistent. The analysis of the TOE´s security requirements with regard to their mutual support and internal consistency demonstrates:

The dependency analysis above for the security functional requirements shows that the basis for internal consistency between all defined functional requirements is satisfied. All dependencies between the chosen functional components are analyzed and non-satisfied dependencies are appropriately justified.

All subjects and objects addressed by more than one SFR are also treated in a consistent way: the SFRs impacting them do not require any contradictory property or behavior of these ‘shared’ items.

The assurance package EAL4 is a predefined set of internally consistent assurance requirements. The dependency analysis for the sensitive assurance components in Section 5.3.3 shows that the assurance requirements are internally consistent as all (additional) dependencies are satisfied and no inconsistency appears.

Inconsistency between functional and assurance requirements can only arise due to functional-assurance dependencies not being met. As shown above, the chosen assurance components are adequate for the functionality of the TOE. Hence, there are no inconsistencies between the goals of these two groups of security requirements.

Glossary

FIDO User Device The computing device where the FIDO Client operates, and from which the user initiates an action that utilizes FIDO.

Computing Environment The user's computer that is interacting with the FIDO Token.

For further FIDO related terms see also the „FIDO Technical Glossary“ of [FIDOSpec].

References

FIDOSpec: FIDO Alliance, Fido Alliance Universal 2nd Factor 1.1 Specifications, 15. September 2016, https://fidoalliance.org/specifications/download/

ICPP: Inside Secure, Infineon Technologies AG, NXP Semiconductors Germany GmbH, STMicroelectronics, Common Criteria Protection Profile - Security IC Platform Protection Profile with Augmentation Packages, BSI-CC-PP-0084-2014, 13.01.2014, v1.0

CC1: Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model; CCMB-2012-09-001, September 2012, Version 3.1, Revision 4

CC2: Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components; CCMB-2012-09-002, September 2012, Version 3.1, Revision 4

CC3: Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components; CCMB-2012-09-003, September 2012, Version 3.1, Revision 4

CCEM: Common Methodology for Information Technology Security Evaluation, Evaluation methodology; CCMB-2012-09-004, September 2012, Version 3.1, Revision 4

FIDOCrypt: FIDO Alliance, FIDO Authenticator Allowed Cryptography List, 24. May 2017, https://fidoalliance.org/wp-content/uploads/fido-authenticator-allowed-cryptography-list-2.pdf