Federal Office
for Information Security

Developer Documentation ARC - Security Architecture

de.fac2 - FIDO U2F Authenticator Applet, v1.34

1.1 (3. Apr. 2019)

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Internet: https://www.bsi.bund.de

Table of Contents

Figures

Tables

1 Introduction

This document contains the security architecture description for the de.fac2 applet as required by the CC part 3, chapter 12.1 as family ADV_ARC.

The TOE is a FIDO Authenticator intended for FIDO Universal Second Factor (U2F) authentication [FIDOSpec]. The authenticator is physically implemented as a security chip with an application written in JavaCard code and is used to securely access online services.

2 Security Domains

The basic but highly efficient domain separation is done implicitly by the underlying Java Card Platform. Applet isolation is achieved through the applet firewall mechanism. This mechanism is already certified and generally described in the document provided to the underlying platform [ST SmartCafe] (see O.FIREWALL and SF.APLLET).

There is no additional domain separation inside the applet as the applet hasn't implemented privileged processes and methods which need to be protected.

The final product only contains the de.fac2 applet on the SmartCafe platform. No other applets will be allowed to be installed on the same card. This is ensured by using secret keys for the platforms Global Platform Manager which are known only to the Composite Product Integrator which installs the applet to the platform. Installing applets without these keys is not possible.

Furthermore, the applet itself only creates unshareable objects, so that they will be kept separate from other resources by the underlying platform.

All keys will be stored in secure key objects which are provided by the platform. See O.KEY-MNGT in [ST SmartCafe].

3 Secure Initialisation

After the de.fac2 applet was uploaded to the card and got instantiated, the applets install() method will be called. This method is invoked only once during the applet lifetime and calls the constructor of the applet and initializes all resources that are needed for using the applet in operational state. The objects created in the constructor are preserved in non-volatile memory, making it available across sessions (power up/down and reset).

The applet uses a byte array for storing temporary data needed at runtime. This byte array is transient and its content will be lost at power downs or deselection of the applet.

When power is removed from the card, any data contained in RAM is lost, but any state stored in non-volatile memory is retained. When power is reapplied, the JavaCard Virtual Machine (VM) becomes active again, at which time the states of the VM and of objects are restored, and execution resumes waiting for further input.

The de.fac2 applet will be installed via the underlying JavaCards GlobalPlatform Manager in a secured environment. Once the applet is installed on the platform, it remains in uninitialized state until the FIDO attestation certificate is uploaded into the applet. The upload of the certificate will also be done in a secured environment. On completion of the certificate upload, the applet will generate the internal secret keys (seed, MACKey) and switches into the operational state. The seed is a AES-CMAC key for the KDF to deviate the Fido private key used in the Fido U2F protocol. The MACKey s is a AES-CMAC key for the MAC calculation over the Fido protocol data. Once the applet is in operational state, no changes on the attestation certificate or any install parameter will be possible anymore. The applet will accept configuration parameter for disabling the user presence check only during the installation process of the cap file as additional installation parameter. Once the applet was successful installed on the platform, this parameter can not be changed anymore. The attestation certificate will be set with a proprietary command (see TSFI SET_ATTESTATION_CERT in [de.fac2 FSP]). Once the upload of the certificate is completed, the applet switches into operational state and doesn't accept the command anymore.

4 Self-Protection

The interface the applet interacts with the APDU interface of the TOE's JavaCard platform subsystem. All incoming and outgoing data is described by the TSFIs in the ADV_FSP documentation. The applet will check all APDUs and only will react to correct and allowed commandos and only return allowed data. The TOE implements a flow control to ensure the correct behavior of the applet all time.

It is possible that an attacker might use other channels to attack and get information which shouldn't get known. To avoid successful attacks, the TOE implements different techniques for detection and prevention. The following actions are done from the applet itself:

- Since some method uses boolean values, the applet replace simple boolean values by 16 bit constant values. An attacker doesn't know which 16 bit value represents a true or false value. If the code doesn't use the given constants, errors will arise. This detection will be done at security relevant places in the code.
- If parameters are modified by an attack resulting in random changes, this is also detectable with high probability as the applet uses MAC over data that comes as input parameter into the applet.
- The applet uses the internal state to allow or prevent different actions. This state is also represented by constant values which are unknown for an attacker. If the current state value isn't a valid value, errors will be detected. Most values are invalid, so the probability of detection is high.
- The applet also uses transactions which will prevent partly executions of code parts. This will avoid incomplete operations.
- Keys which are used at runtime in the applet will be deleted as early as possible, so that there is no chance of misuse by manipulation.
- The input and output data of AES operations will be blinded [redacted]

The TOE uses the following security functions of the underlying platform:

- SF.CRYPTO.2 for clearing keys,
- SF.CRYPTO.3 for encryption and signing functions
- SF.CRYPTO.4 for hashing messages (used for blinding operation)
- SF.CRYPTO.5 for using AES_CMAC as KDF
- SF.CRYPTO.6 as input for random key values
- SF.INTEGRITY to ensure the integrity of the keys and certificates
- SF.TRANSACTION for atomic operations

5 Non-Bypassability

The APDU interface is the only possibility to communicate with the applet and to use the provided functions. Even if physical attacks are possible, these would be defended by the TOE's self-protection functions (see chapter 4). The APDUs will be protected against bypassability by

- Protection of TSFI: All incoming APDUs will be check against misuse. The TOE uses a strict processing. Only known and accepted commands will be processed and lead to a valid answer.
- Encapsulation of Crypto: All crypto functions will be executed in a separate class to reduce the chance of implementation errors.
- Parameter checking: All parameters which contains information for processing a specific incoming APDUs will be checked. If invalid or unknown parameters occur the TOE will abort the processing. In the case the TOE doesn't expect parameters in a specific APDU, any set parameters will be ignored and will not be processed in any way.

6 Design Compliance Justification

The TOE fulfills all requirements of the underlying platform described in [Guide SmartCafe] chapter 3. The following chapters provide rationales for the design compliance and how the requirements are fulfilled.

6.1 Handling of keys

A. All cryptographic operations and storage of keys uses API functions provided by the platform.

B. Not applicable since the TOE doesn't use user authentication

C. Not applicable since the TOE doesn't use DES at all

D. Not applicable since the TOE doesn't use RSA at all

E. Not applicable since the TOE doesn't use authentication

F. All generated keys have a length appropriate for the required level of security. TOE uses AES keys with 128 bit and ECDSA keys with 256 bit as needed.

G. TOE clears temporary keys as soon as they aren't need anymore.

H. The end user (Cardholder) can not use an authentication since the TOE doesn't provide authentication. The TOE doesn't use the SecureChannel interface.

6.2 Handling of User Authentication (PIN verification)

A. Not relevant and not applicable since the TOE doesn't use user authentication.

B. Not relevant and not applicable since the TOE doesn't use user authentication.

6.3 Handling of Application Code

A. Bytecode verification is part of the development process and will be done after each completion of the applet code.

6.4 Changing security application state

A. All updates of the security application state consists only of a single primitive value and therefore doesn't need to be encapsulated in transactions.

6.5 Maintaining the isolation property of the platform

A. For version and revision management and defect reporting git is used.

B. The externalAccess parameter is set to **false** for all instance of cipher or signature objects.

C. See chapter 6.4

D. All implemented security functions in the TOE are specified in international standards and national security bodies.

E. Eclipse is used as software development tool to check program correctness.

6.6 DES / AES cryptographic operations

The applet doesn't use DES operations but AES operations. All input data which will be processed in the AES engine will be preprocessed to avoid that an attacker knows any plaintext block which will be processed by the AES engine. All output data will postprocessed with a hash algorithm to avoid that an attacker knows any ciphertext which was processed by the AES engine. [redacted]

6.7 RSA cryptographic encryption operation

The TOE doesn't use any RSA operations.

6.8 Reading or writing sensitive data to the NVM

The TOE uses the platforms API for reading and write sensitive data. The TOE stores AES and EC keys in objects provided by the platform. The platform itself protects this sensitive data. Therefore there is no need to use additional measures for this key objects.

6.9 Using security relevant API functions with return type 'Boolean'

The TOE uses API functions for checking the MAC which returns type 'Boolean'. Due security reasons the TOE use additional checks by using the platforms special API for (re)checking the last result and add some random delays.

6.10 Generation of symmetric keys

The AES keys used in the TOE are generated by the applet itself by instantiate AES Key objects and filling them with the platforms secure random number generator. No other symmetric keys will be generated off-card and loaded into the smartcard.

6.11 Weekly reset of TOE

The user guidance directs the end user to remove the TOE from the reader after each usage. After each security relevant operation the TOE must be removed before the TOE performs further operations. By removing and returning the TOE to the reader, the TOE performs are reset/power cycle.

6.12 SHA-function

The TOE only uses the high secure SHA-256 function provided by the platform via the interface GenericCryptoFactory.getMessageDigestInstance().

Glossary

U2F Universal Second Factor

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

Reference Documentation

FIDOSpec	FIDO Alliance: Fido Alliance Universal 2nd Factor 1.1 Specifications
ST SmartCafe	G&D: Security Target Lite V2.9 - Sm@rtCafé® Expert 7.0 C3
de.fac2 FSP	Bundesamt für Sicherheit in der Informationstechnik: de.fac2 Developer Documentation FSP - Functional Specification
Guide SmartCafe	G&D: Operational User Guidance Sm@rtCafé Expert 7.0 C3, Version 5.2