AN13266

A5000 - User Guidelines

Rev. 1.3 — 18 November 2022 Application note

Document information

Information	Content
Keywords	Secure Authenticator, A5000, User Guidelines, Plug & Trust, EdgeLock A5000
Abstract	This document provides the guidelines for the usability of A5000 and the security recommendations for using the authenticator.

Revision history

Revision history A5000

Rev	Date	Description
1.3	2022-11-18	• Update Figure 13 • Update Figure 12
1.2	2022-07-22	• Update Section 7.2.5
1.1	2022-05-25	• Update Section 4.1
1.0	2022-03-25	Initial version

1 Introduction

This document provides functional and security recommendations for the EdgeLock A5000 Secure Authenticator to system integrators and application developers. The recommendations are meant for developing a secure solution when the products is used according to those and under normal conditions.

This document refers to the EdgeLock Ready configuration of A5000, these recommendations also apply to EdgeLock A5000 Custom configurations (see Section 6.2).

A difference will be made between single-tenant use and multi-tenant use:

- **Single-tenant** means the A5000 does not protect credentials separately for different users. No separation of users on A5000 is needed to use any of the credentials.
- **Multi-tenant** means the A5000 separates access to credentials based on a secret (= authentication object). Multi-tenant can be multiple (physical) users, but also multiple different applications or even multiple threads in the same application.

The guidelines in this document for single-tenant are always applicable, both for single-tenant and for multi-tenant use of the A5000. Guidelines specific to multi-tenant use do not apply to single-tenant use.

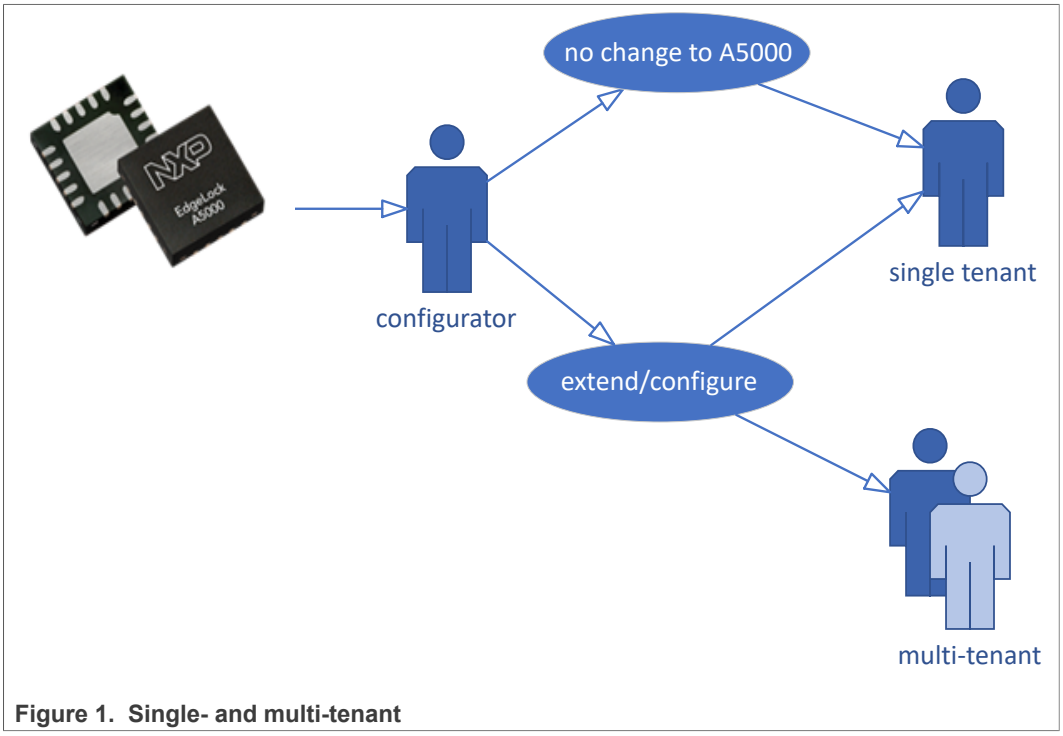

Figure 1. Single- and multi-tenant

The different chapters describe the different possible usages of the A5000:

- **A5000 basics**
  - Describes the basics of the A5000: what are Secure Objects and how they can be used.
- **A5000 Plug and Trust: Usage out of the box** (= use as-is as single-tenant)
  - describes how to use the A5000 Secure Authenticator straight out of the box, with one single entity using the A5000. It provides general guidelines on the use cases which are feasible with the product variants off the shelf.

  - **Ease of Use Configuration** describes the generic A5000 variants which will be available in the market ready to plug and trust. These devices come with a specific set of credentials being trust-provisioned by NXP.
  - **Single-tenant Use Cases** provides an explanation of the most simple use cases applicable to the single-tenant usage of A5000. In this context single-tenant means that the A5000 is operated in a single instance.
- **A5000 Plug and Trust: extendibility** (= configure the A5000)
  - provides support to those users or system integrators who extend the Ease of Use Configuration with new provisioned credentials. This can apply for single-tenant use (e.g. adding additional credentials besides the Ease Of Use Configuration) or multi-tenant use (e.g. configure the A5000 for use by 2 different end users).
- **A5000 Plug and Trust: multi-tenant usage**
  - describes how to use the A5000 Secure Authenticator by multiple entities. It provides general guidelines on the use cases which are feasible with the turnkey product variants.
  - **Module Description Advanced**
    - Secure objects Advanced
    - Policies
    - Object Deletion
  - **Trust Provisioning**
  - **Multi-tenant System**
    - Authenticated Key Creation
    - Multi level SCP
  - **Security Recommendations**
  - **Functional Recommendations**

2 A5000 basics

This chapter explains some basics about the A5000 so users can start using the A5000. It repeats definitions and concepts in a short form as described in the APDU Specification, see: [1].

2.1 Product Information

The A5000 product identification can be obtained out by sending a dedicated command to the secure authenticator.

The Plug & Trust Middleware (nxp.com) includes a utility called ‘Auth_GetInfo’ to retrieve detailed product information from the connected A5000 derivative. It is available as a Windows binary (binaries\ex\VCOM-Auth_GetInfo.exe) and in source code. The html documentation included with the Plug & Trust Middleware package (section ‘Demo & Examples’ > ‘Auth Get Info example’) provides additional information on using and compiling the utility. To setup the additional hardware required to execute the The Plug & Trust Middleware follow one of the EdgeLock™ Quick start guides, for example [12].

The information retrieved by Auth_GetInfo is a superset of what is required to determine whether an entry in the errata sheet is applicable to the product.

The exact product identification is covered by two parameters:

- The product OS configuration (Platform build ID) in the format JXXXXXXXXXXXXXXX. Example below : `J3R351029B411100`

- The version of the application is indicated as Applet version in the format xx.xx.xx (major.minor.patch). Example below: 7.2.0

For information on the available features and configuration of the A5000 please refer to [2].

```
App   :INFO :PlugAndTrust_v04.01.00_20211214
sss   :INFO :atr (Len=35)
  01 A0 00 00  03 96 04 03  E8 00 FE 02  0B 03 E8 00
  01 00 00 00  00 64 13 88  0A 00 65 53  45 30 35 31
  00 00 00
App
  :WARN :#####################################################
App   :INFO :uid (Len=18)
  04 00 50 01    71 7C FD B5    E6 FF 7D 04    31 02 D1 B1
  00 00
sss   :INFO :atr (Len=35)
  01 A0 00 00  03 96 04 03  E8 00 FE 02  0B 03 E8 00
  01 00 00 00  00 64 13 88  0A 00 65 53  45 30 35 31
  00 00 00
App
  :WARN :#####################################################
App   :INFO :Applet Major = 7
App   :INFO :Applet Minor = 2
App   :INFO :Applet patch = 0
App   :INFO :AppletConfig = 1592
App   :INFO :With    ECDSA_ECDH_ECDHE
App   :INFO :WithOut EDDSA
App   :INFO :WithOut DH_MONT
App   :INFO :With    HMAC
App   :INFO :WithOut RSA_PLAIN
App   :INFO :WithOut RSA_CRT
App   :INFO :With    AES
App   :INFO :With    DES
App   :INFO :WithOut PBKDF
App   :INFO :With    TLS
App   :INFO :WithOut MIFARE
App   :INFO :WithOut I2CM
App   :INFO :Internal = FFFF
App
  :WARN :#####################################################
App   :INFO :Tag value - proprietary data 0xFE = 0xFE
App   :INFO :Length of following data 0x45 = 0x45
App   :INFO :Tag card identification data (Len=2)
  DF 28
App   :INFO :Length of card identification data = 0x42
App   :INFO :Tag configuration ID (Must be 0x01) = 0x01
App   :INFO :Configuration ID (Len=12)
  00 04 A7 36    32 1F 01 21    C7 76 19 D8
App   :INFO :OEF ID (Len=2)
  A7 36
App   :INFO :Tag patch ID (Must be 0x02) = 0x02
App   :INFO :Patch ID (Len=8)
  00 00 00 00  00 00 00 01
App   :INFO :Tag platform build ID1 (Must be 0x03) = 0x03
App   :INFO :Platform build ID (Len=24)
  4A 33 52 33  35 31 30 32  39 42 34 31  31 31 30 30
  1A 08 FA 50  67 B5 F2 56
App   :INFO :JCOP Platform ID = J3R351029B411100
App   :INFO :Tag FIPS mode (Must be 0x05) = 0x05
```

```
App   :INFO :FIPS mode var = 0x00
App   :INFO :Tag pre-perso state (Must be 0x07) = 0x07
App   :INFO :Bit mask of pre-perso state var = 0x00
App   :INFO :Tag ROM ID (Must be 0x08) = 0x08
App   :INFO :ROM ID (Len=8)
  2E 5A D8 84  09 C9 BA DB
App   :INFO :Status Word (SW) (Len=2)
  90 00
App   :INFO :Auth_GetInfoPlainApplet Example Success !!!...
App
  :WARN :#####################################################
App   :INFO :cplc_data.IC_fabricator (Len=2)
  47 90
App   :INFO :cplc_data.IC_type1 (Len=2)
  D3 21
App   :INFO :cplc_data.Operating_system_identifier (Len=2)
  47 00
App   :INFO :cplc_data.Operating_system_release_date (Len=2)
  00 00
App   :INFO :cplc_data.Operating_system_release_level (Len=2)
  00 00
App   :INFO :cplc_data.IC_fabrication_date (Len=2)
  13 22
App   :INFO :cplc_data.IC_Serial_number (Len=4)
  00 00 01 95
App   :INFO :cplc_data.IC_Batch_identifier (Len=2)
  58 53
App   :INFO :cplc_data.IC_module_fabricator (Len=2)
  00 00
App   :INFO :cplc_data.IC_module_packaging_date (Len=2)
  00 00
App   :INFO :cplc_data.ICC_manufacturer (Len=2)
  00 00
App   :INFO :cplc_data.IC_embedding_date (Len=2)
  00 00
App   :INFO :cplc_data.IC_OS_initializer (Len=2)
  01 33
App   :INFO :cplc_data.IC_OS_initialization_date (Len=2)
  02 30
App   :INFO :cplc_data.IC_OS_initialization_equipment (Len=4)
  30 30 30 31
App   :INFO :cplc_data.IC_personalizer (Len=2)
  00 00
App   :INFO :cplc_data.IC_personalization_date (Len=2)
  00 00
App   :INFO :cplc_data.IC_personalization_equipment_ID (Len=4)
  00 00 00 00
App   :INFO :cplc_data.SW (Len=2)
  90 00
App   :INFO :ex_sss Finished
```

2.2 Unauthenticated user

For any single-tenant use case, the user can use the A5000 functionality without authentication when both conditions are met:

- there is no interaction between (multiple) users
- no access control is needed to protect credentials against other users.

2.3 Platform SCP

By default, any delivered A5000 device has a SCP03 base key set that contains the same keys for each device-type per OEF (non-die-individual, keys for device types specified in: [2]). For the intended usage and details of the cryptographic keys used with Platform SCP please refer to [9].

Users who want to protect the communication between a host processor and the secure authenticator can use SCP03 on platform level. This secure channel including the key management to update the base keys can be fully managed by GlobalPlatform commands and does not need any A5000 specific command.

2.4 Unbound user

Regardless of the authentication on platform level, the user will not apply any authentication to the application. This is referred to as an ***unbound*** user. Section "Sessions" will detail *bound* users.

2.5 Secure Objects

Anything that is stored or generated inside the A5000 is a Secure Object.

2.5.1 Secure Object types

Supported Secure Object types are:

- **Keys**
  - ECKey = asymmetric key on any of the supported elliptic curves
  - AESKey = symmetric key of 128, 192 or 256 bit; used for AES cipher operations
  - DESKey = symmetric key of 8, 16 or 24 byte; used for DES operations
  - HMACKey = symmetric key of 1 byte up to 256 byte length; used for HMAC and HKDF operations.
- **Files**
  - BinaryFile = a byte array (i.e. general purpose storage)
  - Counter = a monotonic counter
  - PCR = a hash value that can be extended with extra data
  - UserID = user identification string with the length of 4 to 16 bytes that can be used to group secure objects and allow their usage in an associated session (intended for use cases where a trusted operating system on a host MCU/MPU is isolating their applications based e.g. on their application ID).

See the [1] for more information.

2.5.2 Secure Object Attributes

Secure Object attributes are linked to any Secure Object. The attributes are:

- *Object identifier* = a unique identifier of the Secure Object
- *Type* = Secure Object type
- *Policy* = Access control applicable to the secure object
- *Origin* = Origin of the data, either external, internal or provisioned
- Additional attributes (only applies to multi-tenant; see Section 5)
  - *Authentication attribute*

- *Object counter*
- *Authentication object identifier*
- *Maximum authentication attempts*
- *Minimum output length*
- *Minimum tag length for AEAD operations*

2.5.2.1 Object identifier

The object identifier cannot be modified during the object lifetime, so it remains the same until the object is deleted.

Object identifiers are always defined externally, the A5000 will not (automatically) assign object identifiers to objects. However, there is a set of reserved identifiers that are assigned to serve specific use cases. For more information, refer to [1] and [2].

All pre-provisioned credentials being trust-provisioned by NXP as part of the Ease Of Use Configuration have an identifier from the range “application Reserved Area” or “NXP reserved region” in Table 1. Customers that will create their own Secure Objects are advised to use an identifier from the range “In field usage”.

Table 1. Identifier for application reserved area or NXP reserved region

Address Range	IDs
0x00000000-0x7BFFFFFF	In field usage
0x7C000000-0x7CFFFFFF	Android Key Master area
0x7D000000-0x7DFFFFFF	Demo area
0x7FFF0000-0x7FFFFFFF	application reserved area
0x80000000-0xFFFFFFFF	NXP reserved area

2.5.2.2 Type

See Secure object types.

2.5.2.3 Policy

A Secure Object policy defines the access control to a Secure Object by specifying the operations that each user can perform.

For access control the user must set the policy according to the use case of their system (see section Secure Object policy). For any Secure Object the minimum access control policy for each user has to be set up.

If no explicit policy is passed to a Secure Object at object creation, a default policy will apply as specified in [1] and accroding to the features available on the A5000 variant as described in [2].

Secure Object policies are assigned at object creation and cannot change afterwards, so they remain constant over the lifetime of a Secure Object.

2.5.2.4 Origin

The origin attribute indicates the origin of the content of a Secure Object: either externally generated, internally generated or if trust provisioned by NXP.

3 A5000 Plug and Trust: Usage out of the box

3.1 Ease of Use Configuration

All generic A5000 variants will be offered pre-provisioned according to a profile.

For more details, refer to [2].

3.2 Single-tenant protection

The configurator is responsible to bring the A5000 in a usable state for the end user's application.

The configurator and end user might be the same entity, e.g. a developer for prototyping purpose.

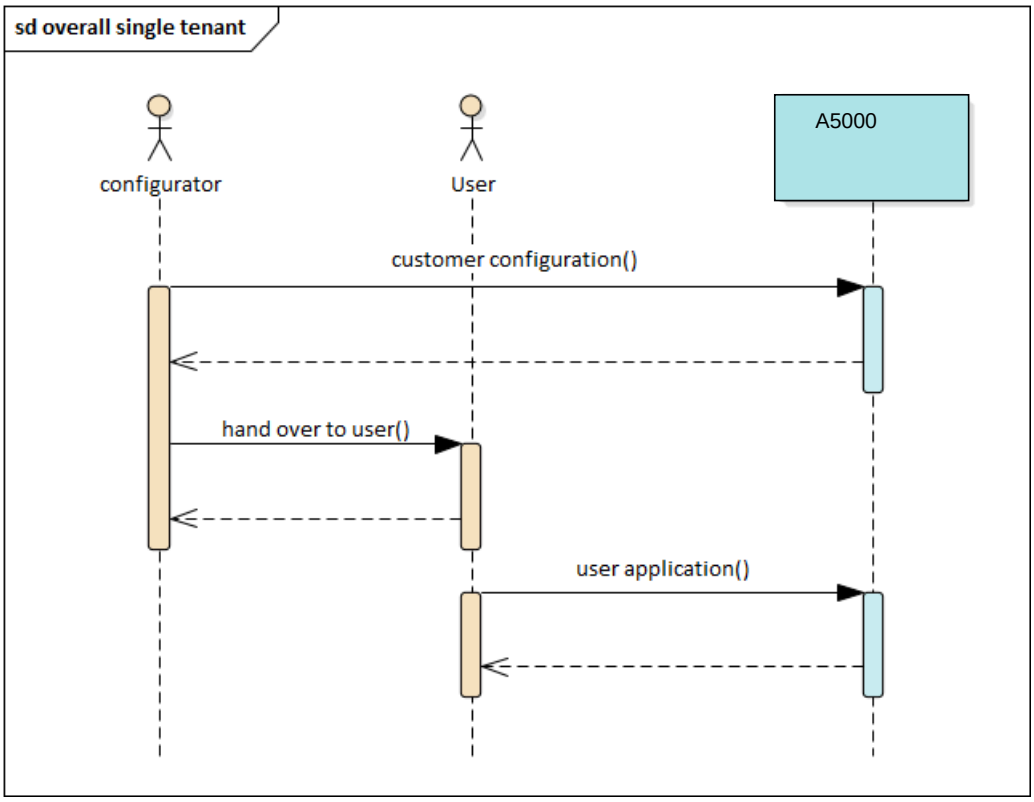

Figure 2. Overview single-tenant use

Note:

Configurator and User can be the same entity

3.3 How to update Platform SCP keys

The Platform default keys are available in [2].

SCP03 base keys can be updated as described in [9], using the existing DEK key as encryption key for the new keys to be set.

The Plug&Trust Middleware contains an example program to rotate (i.e. update) the Platform SCP03 keys.

The example program – and additional documentation - can be found in the demo folder.

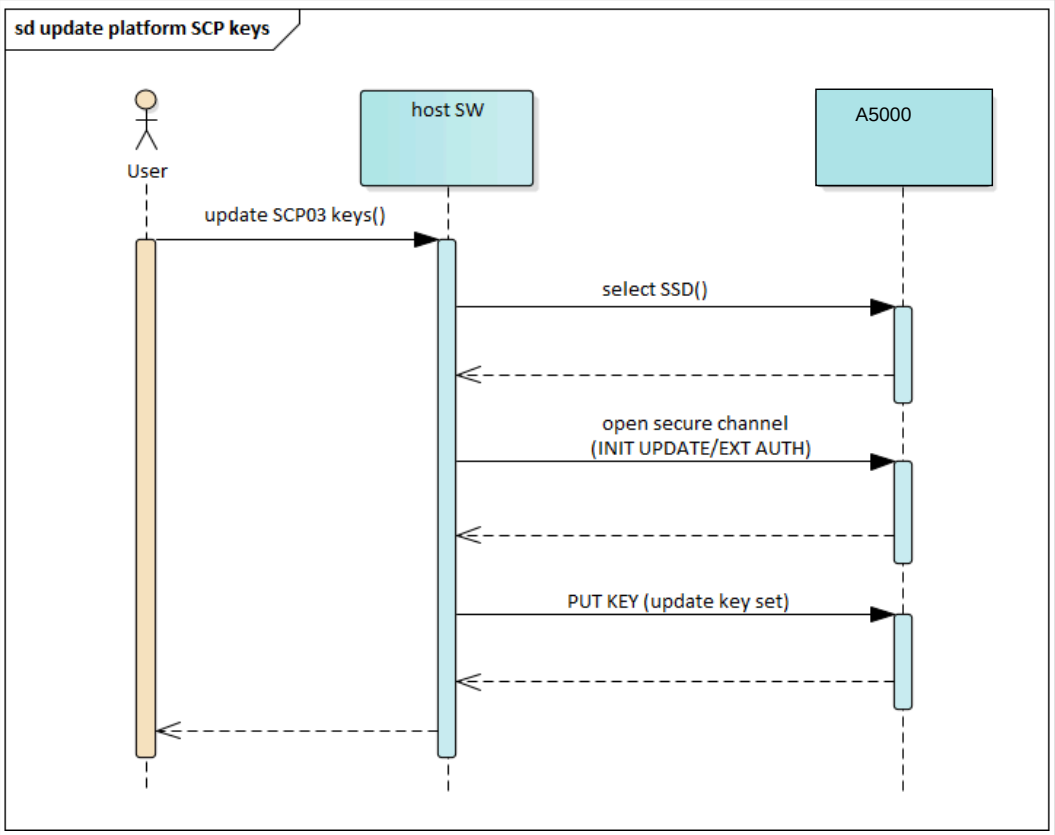

Figure 3. Enable Platform SCP

3.4 Attestation

Attestation is a means to prove that the data is within the secure authenticator and for the case of attesting an I2C Controller response, that the data is from the attached I2C bus. The secure authenticator “attests” the origin of the data by signing it with a trust provisioned attestation key by NXP. When key or file data are requested by the user, the user can request attestation for the returned data. Attestation is achieved by adding into the response of the requested data the chip unique identifier + freshness (i.e. a random value) + a timestamp (i.e. monotonic counter value) + a signature over the full payload (requested data + unique identifier + freshness + timestamp).

All the generic A5000 variants have an attestation key trust provisioned by NXP. For more details on the configuration profiles of the A5000 refer to [2].

The certificate is signed by NXP Root of Trust entity. Attestation requires trust which is ensured by the issued certificate. To verify the validity of the attestation, the signature on the attested object is checked against the attestation certificate.

Security recommendations on attestation are detailed in Attestation”.

Use cases

1. Generated Key attestation

A5000 can generate keys internally. The attestation mechanism is used to attest that the keys have been generated inside the A5000.

2. External Data attestation
   Customer might inject/provision data inside the A5000. The attestation mechanism can be used to prove that the data has been stored in the secure authenticator without modification.

For more information, see Attestation of provisioned objects.

4 A5000 configuration extendibility

This chapter provides information to those users and customers who desire to extend the A5000 product or a custom EdgeLock variant with customized keys and credentials besides the Ease Of Use configuration.

Security recommendations for extending A5000 Ease of Use configuration are provided in Extendibility and Multi-tenant.

4.1 Adding Secure Objects

Users can add Secure Objects to the Ease of Use configurations by creating new Secure Objects. During creation, the user needs to assign the object identifier (see Object identifier). For access control the user must set the policy according to the use case of their system (see section Secure Object policy). For any Secure Object the minimum access control policy for each user has to be set up.

Users have to choose between persistent and transient Secure Objects (in case both are supported; see [1]). Persistent Secure Objects value is always written to NVM while transient Secure Objects value is written into RAM.

For transient Secure Objects, it is possible to export the value (in encrypted form) of the Secure Object to the host controller and later on import the value again, as long as the Secure Object has not been deleted from the A5000. Persistent Secure Objects cannot be exported or imported. Import and Export of a Secure Object requires the policy POLICY_OBJ_ALLOW_IMPORT_EXPORT to be set. To prevent re-import by unauthorized parties, POLICY_OBJ_ALLOW_IMPORT_EXPORT MUST be restricted to an authentication credential. To prevent modification of the Secure Objects attributes its deletion MUST be restricted by binding policy POLICY_OBJ_ALLOW_DELETE to an authentication credential.

4.2 Reserved identifiers

To prevent denial of service attacks reserved identifiers for secure objects, which are not pre-provisioned by NXP, MUST be created, preferrably to device individual values.

This is especially important for the following identifiers:

- RESERVED_ID_TRANSPORT
- RESERVED_ID_FACTORY_RESET
- RESERVED_ID_I2CM_ACCESS
- RESERVED_ID_RESTRICT

INTERNAL NOTE: artf1266018

4.3 Creating Crypto Objects

By default, users have the possibility to do either cipher, signature or digest operations in one shot, meaning the input data are passed to the A5000 and the output data are returned directly. The A5000 does not keep any state in that case.

However, when users have the need to pass several blocks consecutively to the A5000 for one of those cipher, signature or digest operations, a Crypto Object can be allocated.

A Crypto Object will keep the state of a crypto operation and allow typically to do init/(n times) update/final operations.

4.4 Adding an attestation key

Customers can provide their own attestation key(s) (and related certificates) to perform the attestation as explained in Attestation. These keys must be injected or generated either in an secure environment or via a secure channel.

4.5 Adding Cloud Connection keys

A5000 Ease of Use configuration can be extended with credentials to onboard and connect securely to various clouds. Customers might decide to use their own PKI and CA.

More details on this use case can be found in [3] and [4].

4.6 Apply transport lock

Transport lock use provides security recommendations to securely use the transport lock.

4.6.1 Simple Use Case

The transport lock is a secure object which can be used to protect the modules on the logistic chain.

The transport lock MAY be used as tamper seal between entity A and entity B. Entity A applies a lock and share the key with entity B to give authorized access only to entity B. In this case entity B MAY be the final receiver of the devices.

4.6.2 Updatable Transport Lock

There might be more than two entities in the logistic chain. In this scenario each entity MAY be both a customer and a configurator.

In the case of a cascade logistic chain, entity A MAY make the Transport Lock updatable.

Receiving entity B can remove the lock of A and update the lock for further entities into the logistic chain.

Entity B MAY apply a specific lock for each entity which will receive the product.

4.7 Factory reset

Factory key reset allows to delete all objects except for those where the origin attribute is configured to "provisioned". This is the case for all keys belonging to the NXP Ease of Use configuration as well as mandatory secure objects such as UUID.

To prevent the unintended deletion of keys the secure object with ID RESERVED_ID_FACTORY_RESET MUST be set.

Note: *Certificates trust provisioned by NXP will be deleted after factory reset.*

Note: *Platform SCP keys are unaffected by the factory reset procedure.*

4.8 Object deletion

As described above, some of the credentials injected into the Ease of Use configuration, such as the Cloud Onboarding certificates, can be deleted by the customer if desired.

To delete them, the credentials must be overwritten first. This will change the origin from “provisioned” to either “internal” or “external” (depending on the write method) and will allow deletion either via individual deletion or via the factory reset.

Note: *To ensure the correct execution of the deleteAll command, a ReadIDList command or individual CheckObjectExists commands must be performed under the protection of a secure channel (with R-MAC) to ensure the proper deletion of the previously existing object(s). The response must indicate that the object has been correctly deleted.*

4.9 Importing external objects

Note: *The APDU “ImportExternalObject” must not be used without first contacting NXP to avoid potential problems. If you have used or plan to use the APDU “ImportExternalObject,” please make sure you contact your NXP representative.*

Users might import credentials using importExternalObject or writing to a new object using either application-SCP03 or ECKey session.

ImportExternalObject, same as WriteSecureObject, can be performed by any party as long as not disabled by DisableSecureObjectCreation or TransportLock. The A5000 keypair (ID 0x7FFF0202, RESERVED_ID_EXTERNAL_IMPORT) is provisioned already by NXP as specified in [2]. The public key of the external party needs to be inserted as authentication object into any other chosen secure object ID, to use this feature.

Any command to write or update a credential can be sent. In addition to an unprotected command, ImportExternalObject commands are signed, encrypted and implicitly authenticated.

The APDUs for ImportExternalObject can be replayed. Access control via policies and/or versioning of the SecureObject can be used to restrict this.

INTERNAL NOTE: This is due to partially overwriting RSA keys, which can be exploited to leak keys. https://collabnet.level1.ddm.nxp.com/svn/iot_rs_repo/SE051/Security/ExportImportWrite_Attack/SE05x attacks mitigation v2.xlsx

Note that APDU "WriteECKey" does not check consistency of the components being written. Also, the APDU does not check P1_KeyType when executed on an existing object.

INTERNAL NOTE: artf1277049

INTERNAL NOTE: artf1279513

APDU "WriteSecureObject", when executed on an existing object, does not serve with attribute checks for "Authentication indicator", "Minimum tag length for AEAD operations",

"Minimum output length" or "Maximum authentication attempts". The policy check of APDU "WriteECKey" does not cover verification of the existing ECKey object's curve.

INTERNAL NOTE: artf1279790

4.10 Single-tenant use cases

4.10.1 Cloud Connection

The Ease of Use configuration can be used to onboard and connect securely to various clouds as explained in the Cloud onboarding AN of A5000. More details on connections to specific clouds can be found on the related Application Notes of SE05x:

- SE05x for secure connection to Azure IoT Hub
- SE05x for secure connection to AWS IoT Core
- SE05x for secure connection to GCP
- SE05x for secure connection to IBM Watson IoT

4.10.2 Device to Device Authentication

Every A5000 product variant comes pre-provisioned with credentials which can be used for Device to Device authentication.

More details on the use case can be found in [7].

4.10.3 Attestation of provisioned objects

Note that for attestation, the key pair that performs the attestation (i.e. signing) needs to be part of a trusted certificate chain.

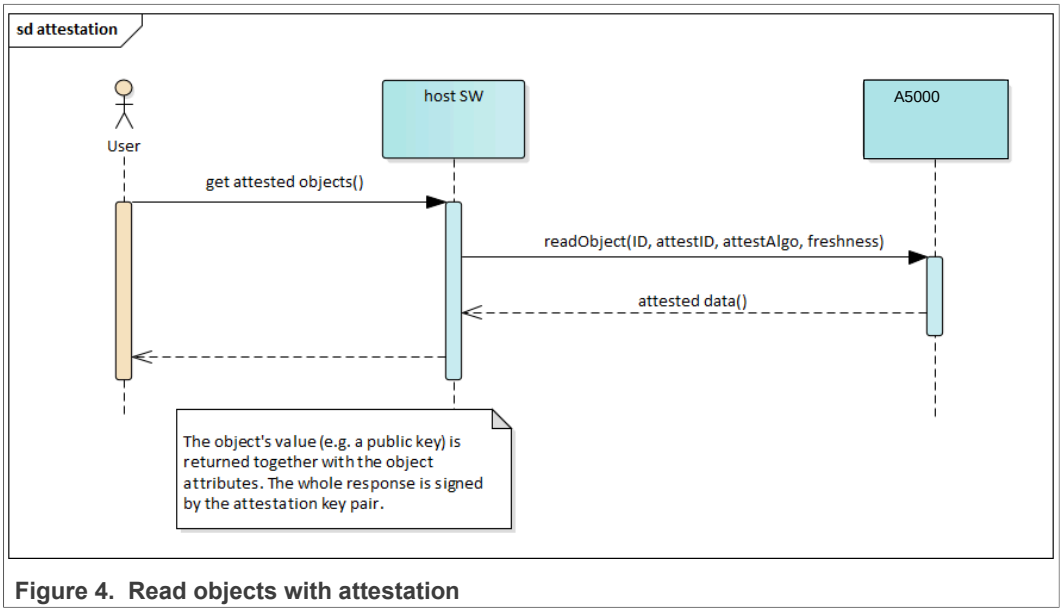

Figure 4. Read objects with attestation

4.10.4 User application

Any command can be sent in the default session (no authentication to the application nor command wrapping are needed).

Data are encrypted and protected by Platform SCP.

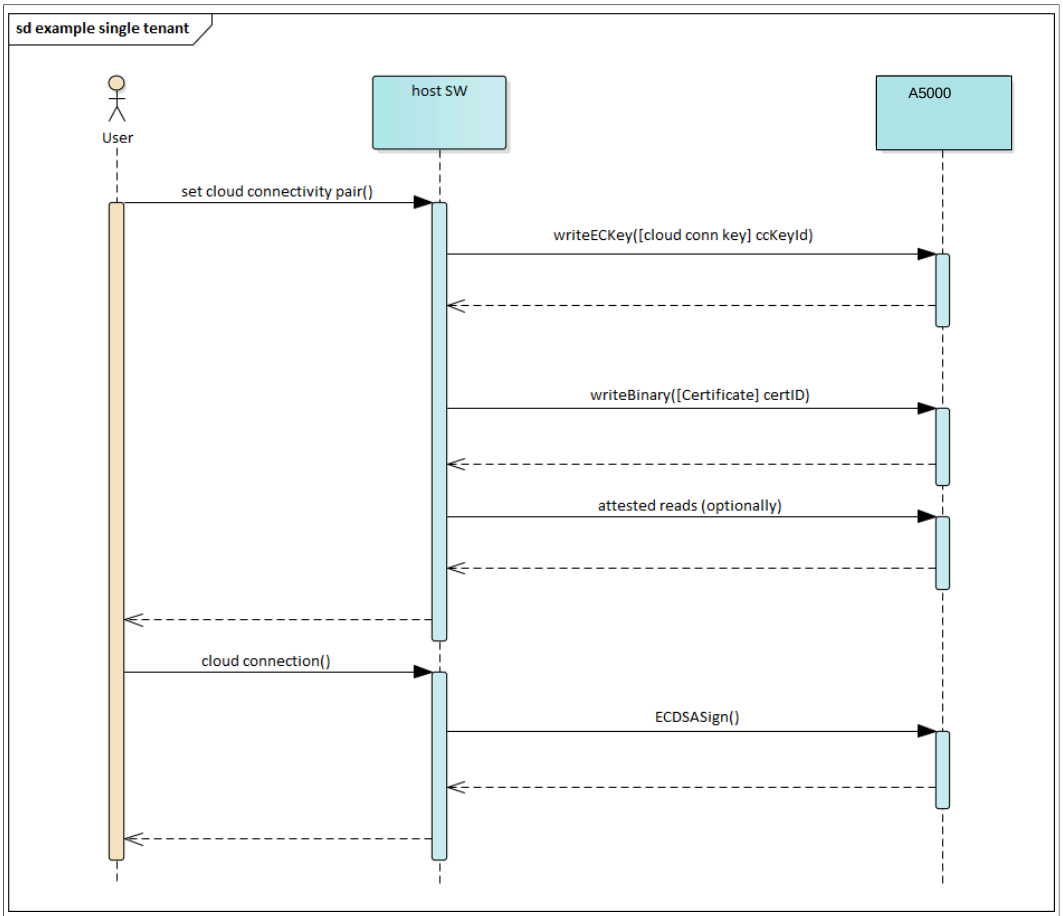

Figure 5. Single-tenant user application example

5 Multi-tenant use of A5000

5.1 A5000 features for multi-tenant use

5.1.1 Authentication Objects

An Authentication Object is a specific Secure Object that allows users to mutually authenticate against the A5000 application. In that sense the value of the object is protecting access to the A5000.

Users who use an Authentication Object to authenticate against the A5000 application are referred to as ***bound*** users (as opposed to the *unbound* user).

5.1.1.1 Authentication Object Creation

The entity that creates the Authentication Object is referred to as the ***authentication object owner***. The owner can be a single entity or multiple entities (whoever knows the value of the Authentication Object's value).

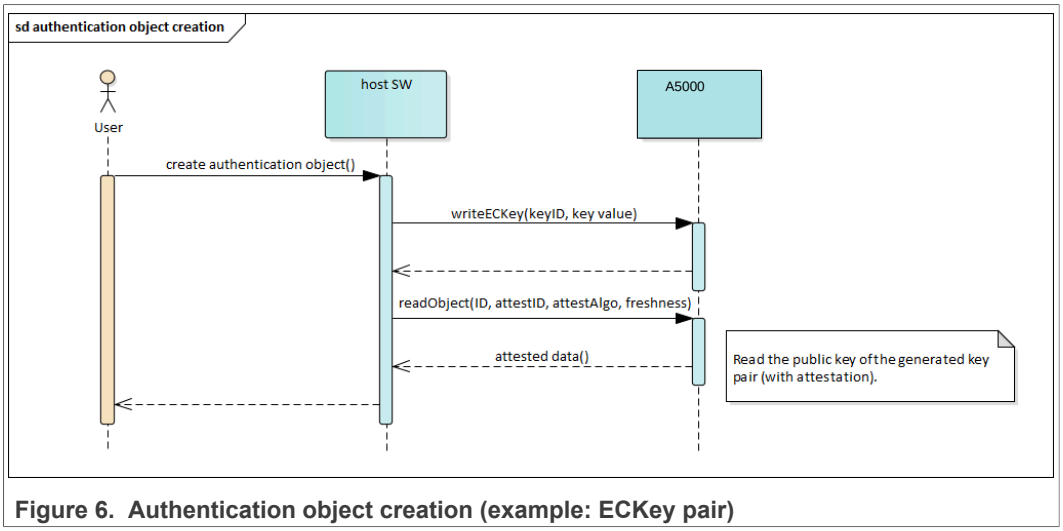

Figure 6. Authentication object creation (example: ECKey pair)

5.1.2 Sessions

The A5000 allows to open a **session** with any of the following Secure Objects:

- userID (to open a UserID session)
- AES128 key (to open a SCP03 session)
- ECKey pair or EC public key (to open a ECKey session)

UserID sessions are using communication in plain, whilst SCP03 or ECKey sessions rely upon SCP03 secure messenging and therefore provide end-to-end protection (see: security recommendation.

If a user opens a session:

- the access rights for the unbound user do not longer apply
- the user will become known by the Authentication Object ID that was used to open the session and becomes a ***bound*** user.

By default, if a SCP03 or ECKey session is established, **application level SCP** will apply end-to-end for this particular session. On application level SCP is using SCP03 secure messaging; the difference between an SCP03 and a ECKey session is the authentication method, where SCP03 is based on symmetric crypto and ECKey is using asymmetric crypto for the authentication.

Authenticated User Session provides recommendations for a secure use of sessions.

Note that in cases where Platform SCP is used, the application level SCP is wrapped inside of the Platform SCP channel, see: [1].

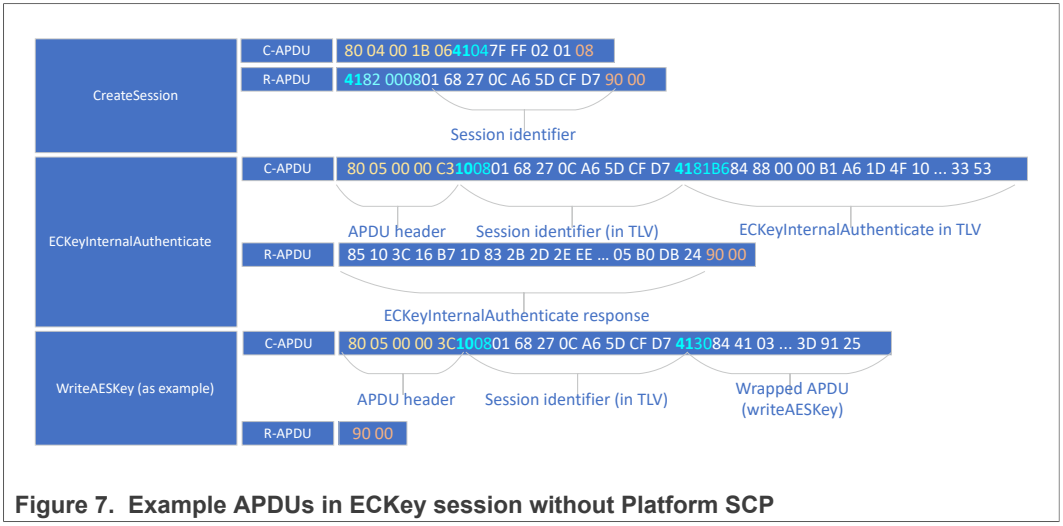

Figure 7. Example APDUs in ECKey session without Platform SCP

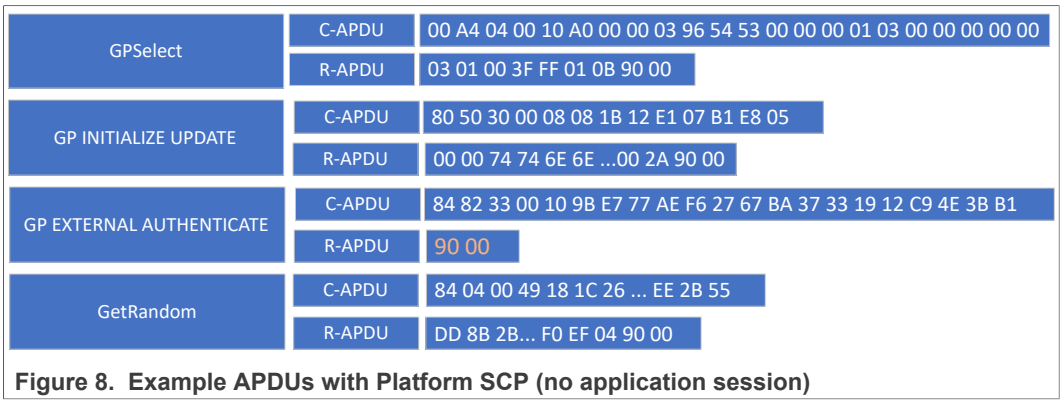

Figure 8. Example APDUs with Platform SCP (no application session)

5.1.2.1 Session policies

Besides Secure Object policies, the user can also assign a policy to a session. If a policy is passed as argument during session creation, the session will be limited in its lifetime by the number of APDUs that are sent within the session. So the user passes the maximum number of APDUs and when this maximum is reached, the session is broken down and the user is no longer authenticated.

Note that the user can call SessionRefresh to extend the lifetime of a session.

5.1.2.2 Example UserID session

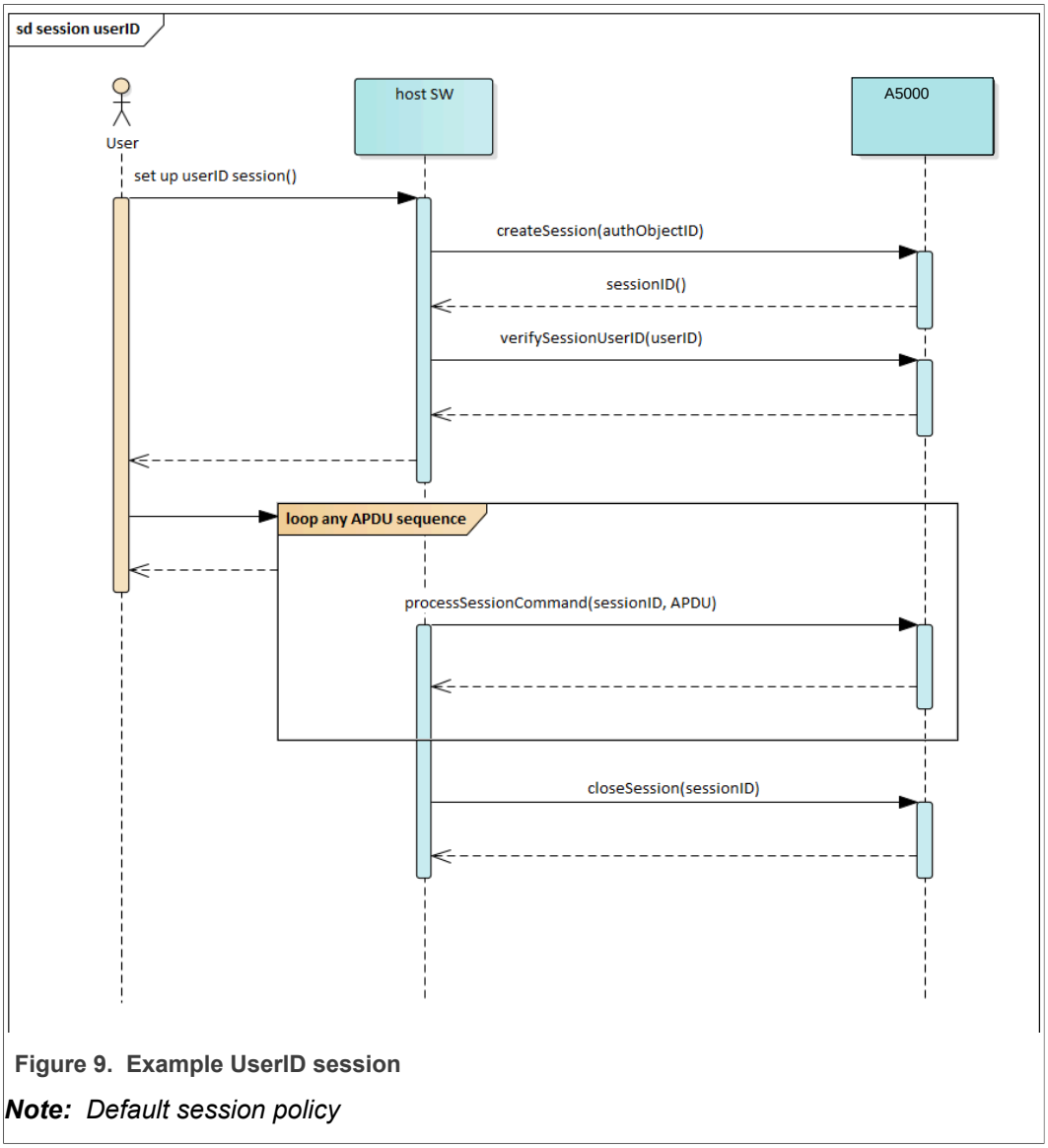

Figure 9. Example UserID session

Note: *Default session policy*

5.1.2.3 Example SCP03 session

Here the SCP03 protocol as defined in Global Platform (Card Specification v 2.2 – Amendment D) is used. The Global Platform SCP03 session is encapsulated within an application session as shown in the picture below and is then called “application SCP03 session”.

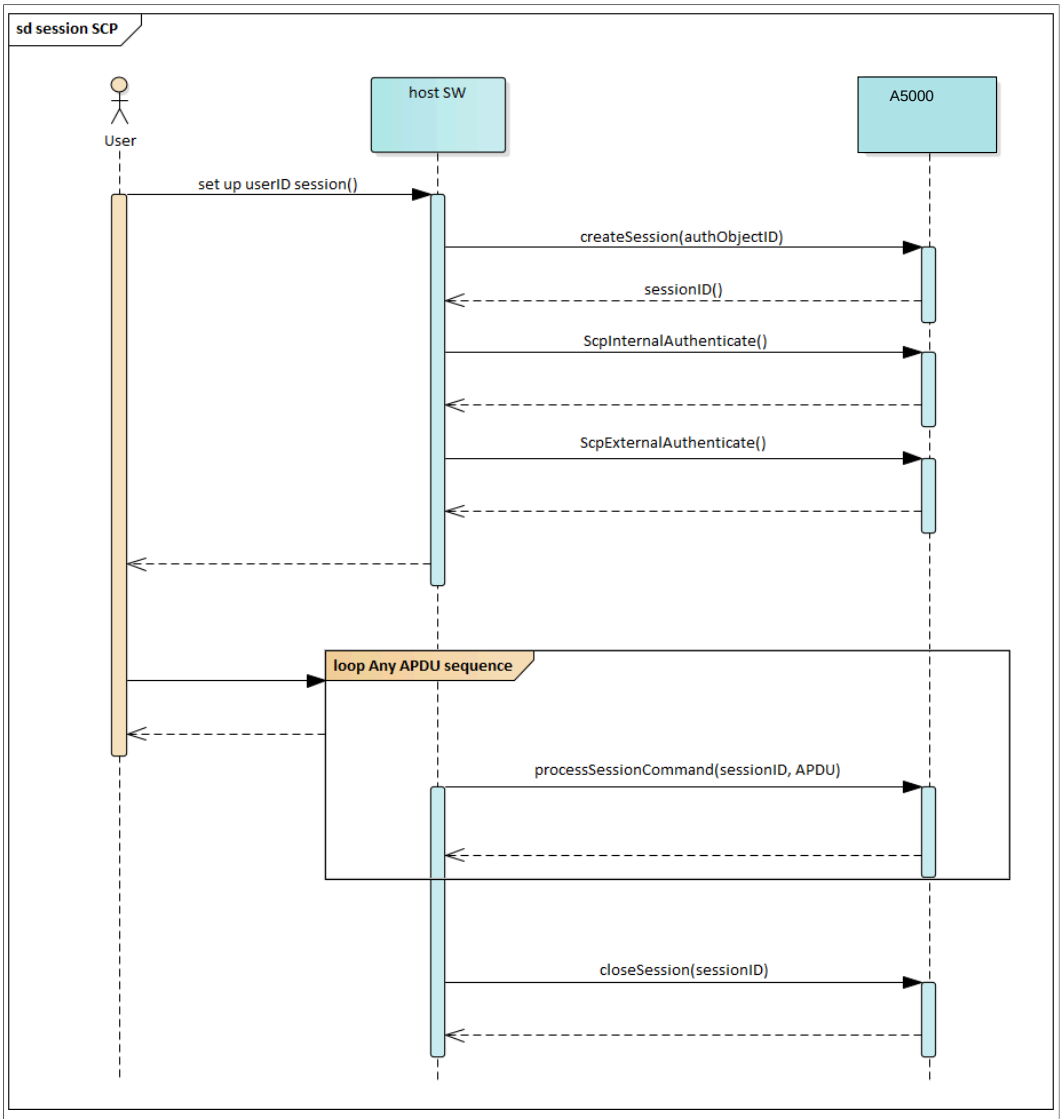

Figure 10. Example SCP session

Note: *Default session policy*

5.1.2.4 Example ECKey session

ECKey sessions uses ECKeys stored in the A5000 to open an bound user session. The authentication is done using the asymmetric ECKey and the session is encrypted using the SCP03 protocol. The ECKey authentication is defined in [1].

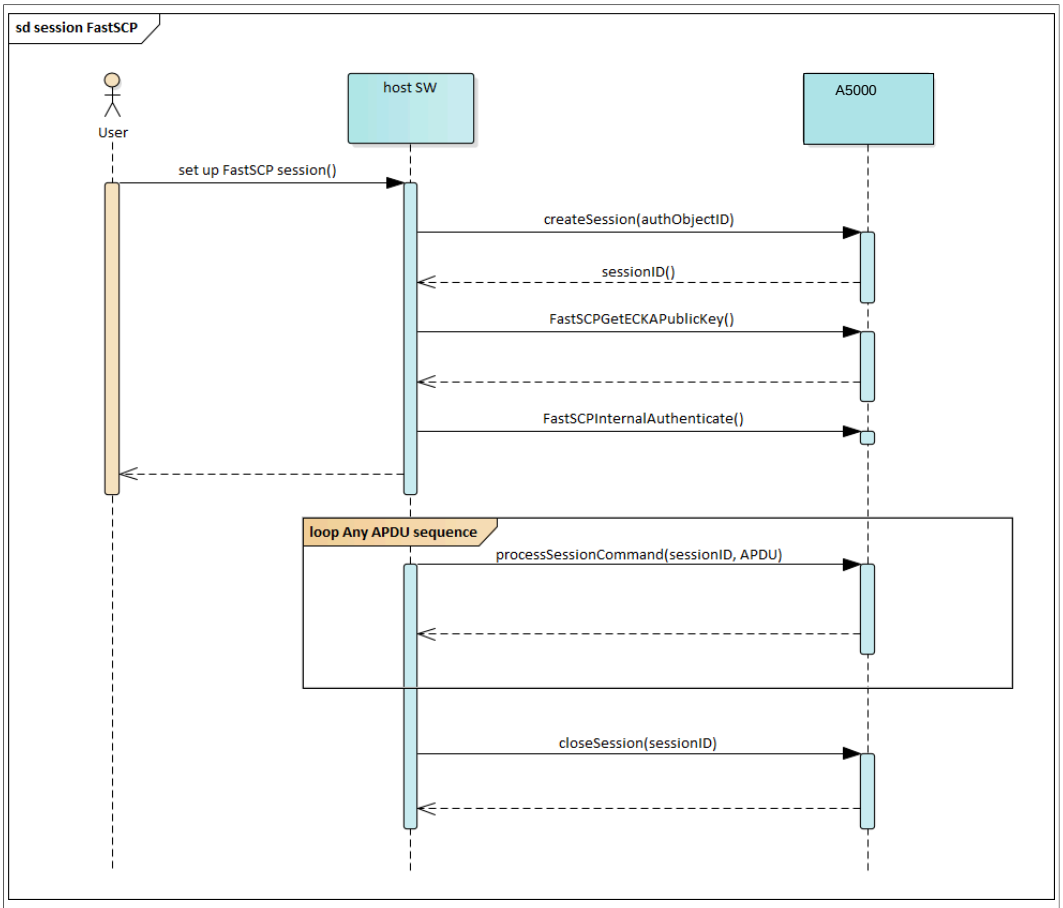

Figure 11. Example ECKey session

Note: *Default session policy*

5.1.3 Secure Object policies for multi-tenant use

When multiple users are configured or expected, the Secure Object policies need to be set up to allow the right users to perform the right operations (and forbid to the wrong user).

As a (simple) example, there could be a mapping of users to object policies as follows:

Table 2. Secure Object policies for multi-tenant use

	User A	User B
POLICY_OBJ_ALLOW_DELETE	Granted	Denied
POLICY_OBJ_ALLOW_READ	Granted	Granted
POLICY_OBJ_ALLOW_WRITE	Granted	Denied
POLICY_OBJ_ALLOW_SIGN	Denied	Granted

6 Trust Provisioning

This chapter focuses on the credentials provisioning.

Credentials provisioning provides recommendations on how to provision credentials in A5000 in a secure manner.

6.1 Trusted or untrusted environments

The provisioning phase of the device is critical to the security of the product, since sensitive data and key material are being generated on or injected into the A5000.

A device can operate in different types of environments:

- **Trusted environment** – a trusted environment is a secured environment under control of a trusted party. The level of security depends on the environment and should be set appropriate to the threats and security objectives relevant to the device and its assets
- **Non-trusted environment** – a non-trusted environment is an unsecured environment and no control can be applied to it.

6.2 A5000 Trust Provisioning

The IoT device identity should be unique, verifiable and trustworthy so that device registration attempts and any data uploaded to the OEMs servers can be trusted. The A5000 is designed to provide a tamper-resistant platform to safely store keys and credentials needed for device authentication and registration to OEMs cloud service. Leveraging the A5000 security IC, OEMs can safely authenticate their devices without writing security code or exposing credentials or keys.

The following options are available for provisioning the EdgeLock A5000 security IC:

- EdgeLock 2GO Ready: Every EdgeLock A5000 product variant comes pre-provisioned with keys which can be used for all major use cases, including device-to-device authentication.
- EdgeLock 2GO Custom: NXP offers a customization service for injecting the credentials that you need during the A5000 IC manufacturing. Please contact NXP for more information on this service.
- EdgeLock 2GO Managed: NXP offers a cloud service for remotely configurating your A5000. EdgeLock 2GO Managed is a secure and flexible way for provisioning the keys and certificates required on your devices and to manage the lifecycle of your device credentials. You can find more information and request an evaluation account at www.nxp.com/EdgeLock2GO .
- EdgeLock A5000 provisioning by OEMs, distributors or third-party partners: OEMs can provision EdgeLock A5000 on their own or select a distributor or third-party partner for provisioning the A5000.

7 Security Recommendations

This chapter describes requirements and recommendations which have to be followed to use the production in a secure manner. Not complying bears a risk of security gaps.

“Must”, “should” and “may” are used in compliance with RFC2119 (see [8].)

- **Must** indicates an absolute security requirement

- **Should** indicates a security recommendation, meaning there may exist valid reasons in particular circumstances to ignore a particular item
- **May** mean that an item is optional

7.1 Generic recommendations (all use cases)

7.1.1 Platform and application level SCP

Either Platform SCP or application level SCP MUST be used to protect locally against eavesdropping and malicious command injection.

In operation stage the secure authenticator MUST be configured to mandate platform SCP or mandate application level SCP by secure object policy. In order to guarantee confidentiality and integrity on the APDU interface the full security level, which means command and response MAC and encryption (see APDU specification) MUST be enabled. The NXP PlugAndTrust MW uses the full security level by default.

Confidentiality, integrity and authenticity of the host SCP key set MUST be enforced as required during provisioning of the keys, outside of secure authenticator, in the host and its memories.

- Access control on host keys MUST be enforced when available, or any mechanism that protects the keys from being disclosed
- Further security properties MUST be preserved by environmental measures.

Default SCP keys MUST be updated at first use of the product. It is recommended to use die individual keys to allow a mutual binding between the host and the secure authenticator.

Confidentiality, integrity and authenticity of the SCP key set MUST be enforced as required during provisioning of the keys, outside of A5000, in the host and its memories.

- SCP keys MUST be stored securely in the host. With PlatformSCP it is recommended to only have the encryption (ENC) and macing (MAC) keys stored on the host while having the data encryption (DEK) key, which is used to update the keyset within the A5000, not stored on the host. For more information on the intended usage and details of the cryptographic keys used with Platform SCP please refer to [9].
  INTERNAL NOTE: artf1265793
- Access control on host keys MUST be enforced if available, or any mechanism that protects the keys from being disclosed.
- Further security properties MUST be preserved by environmental measures.

7.1.2 Initial State

Upon reception of A5000, the authenticity of the provisioned Secure Objects MAY be checked by reading the Trust Provisioned flag of the Secure Object.

- The presence of the Trust Provisioned flag ensures the secure object has been securely provisioned by NXP.

During A5000 lifecycle, the authenticity of the provisioned secure objects MAY be checked by reading the Secure Object content with attestation mode.

7.1.3 Attestation

7.1.3.1 Attestation keys

Secure Objects used for attestation MUST have the POLICY_OBJ_ALLOW_ATTESTATION explicitly set and MUST NOT have the Policy rule POLICY_OBJ_ALLOW_SIGN or POLICY_OBJ_ALLOW_DECRYPT set.

INTERNAL COMMENT: Rationale: Otherwise a sign operation on ECDSA key resp. a sign or decrypt operation with RSA allows to fake an attestation.

When an attestation key is used the attestation key MUST be trusted (e.g. by validating the certificate chain associated with the public key)

It is recommended to use a pre-provisioned secure object for attestation and check the integrity of the key with an existing certificate. This is the case for credentials trust provisioned by NXP, indicated by the ORIGIN_PROVISIONED and related certificates.

Note: The product variant "C" comes with a certificate for the attestation key pre-provisioned.

7.1.3.2 Read with attestation

When reading with attestation, all returned fields MUST be checked for each attestation to prevent reuse of attestation.

- The attestation signature MUST be verified with a trusted attestation key.
- The timestamp field for consecutive attestations MUST be checked to be consecutive. Note: the timestamp must never have a value lower than the one from the previous attested read.
- The freshness (user input) in the C-APDU MUST be unique/random.
- The object id, type and object attributes MUST be compared against expectation.

The attested read to an ECKey object does not give evidence on correctness of the private key. The attested read to an ECKey object does not give evidence on correctness of the keys curve.

7.1.3.3 Distinguishing between Attestation Responses

To distinguish an attestation response of an application version equal or higher than 7.0 from a previous version the TAG_TS (0x4F) of the timestamp shall be used. If TAG_TS (0x4F) is present in a validly signed attestation response it indicates an attestation response of application version equal to or higher than 7.0.

Note that TAG_ATT_SIG (0x52) is not part of the signed content and MUST NOT be used to descriminate versions.

7.1.4 Secure Object policy

If access control to a Secure Object is required the user MUST set the policy of the object.

- NXP defined the default policy without considering any end use case application. It is solely responsibility of the developer to assess whether such policy can be used in their application. NXP recommend to set the policy according to needs and security of the end application.

- It is not recommended to use the default policy as NXP intended it for prototyping and it will therefore allow more access rights to credentials than necessary for the particular use case.
- For access control the user must set the policy according to the use case of their system (see section Secure Object policy). For any Secure Object the minimum access control policy for each user has to be set up.

7.1.5 Key usage

It is recommended to use a key only for one dedicated purpose (e.g. signature creation, encryption or decryption) and to not mix use cases such as:

- Signature of data
- Certificate signing
- key encipherment
- data encipherment
- key agreement

Example: A key used for message decryption shall not be used for signature generation. This is independent of the low level crypto operations allowed for the key, which can be the same operation.

Example: An AES key used for data encipherment shall not be used for signing data using CMAC.

INTERNAL NOTE: the RSA signature generation requires the Policy "DECRYPT" to be enabled for signing a hash created by the MW.

7.1.6 Key Derivation Functions

When a function allows optionally output to be stored into a target object instead of returning via the R-APDU, the POLICY_OBJ_FORBID_DERIVED_OUTPUT can be applied on the source object to prevent output being returned to an external party and as such mandate the use of a target object on the A5000. For more details see [1]

When a Key Agreement and a Key Derivation Function are combined in succession, while the intermediate result remains on the A5000, (e.g. ECDHGenerateSharedSecret followed by an HKDFExtractAndExpand) to avoid manipulation as well as execution of different intermediate commands during the entire processing, it MUST be executed within a session.

A target object used in TLSCalculatePreMasterSecret with key exchange algorithm ECDHE_PSK cannot be protected against manipulations by setting ALLOW_DERIVED_INPUT, which is verified against only one of the two source objects. Protection of such target object against manipulations could instead be achieved with POLICY_OBJ_REQUIRE_SM, with restricting ALLOW_DERIVED_INPUT to an authentication object or by mandatory Platform SCP.

In case function HKDF shall return a derived secret into a target object, related source object can be assigned with a minimum output length in its attributes to preserve confidentiality of this derived secret.

An ECKeyPair used as source object in TLSCalculatePreMasterSecret for key exchange algorithm ECDH or ECDHE_PSK can be assigned with POLICY_OBJ_FORBID_DERIVED_OUTPUT to prevent its misuse in

ECDHGenerateSharedSecret for retrieval of related shared secret in its response. (As it shares ALLOW_KA with APDU ECDHGenerateSharedSecret.)

7.1.7 Crypto Operation Cross-Usage

7.1.7.1 Crypto Operation Policies

All modes of the AES crypto operations use the same cryptographic primitive for encryption and for modes CTR, CCM and GCM//GMAC encryption and decryption are mathematically equal. Therefore it is not possible to prevent a decryption operation in mode CTR, CCM or GCM/GMAC by omitting the policy POLICY_OBJ_ALLOW_DEC while POLICY_OBJ_ALLOW_ENC is present.

FORBID_EXTERNAL_IV can be used in combination with POLICY_OBJ_ALLOW_ENC to generate the IV on the A5000 and return it, thereby making a decryption in AES CTR, CCM or GCM/GMAC mode not practical as the IV cannot be chosen.

INTERNAL NOTE: artf1263019

There is no mode out of ECB, CBC, CTR, CCM and GCM/GMAC for which encryption operations could be prevented by omitting the policy POLICY_OBJ_ALLOW_ENC while POLICY_OBJ_ALLOW_DEC is set.

7.1.7.2 Crypto operations in AEAD modes

The internal authentication keys in Authenticated Encryption with Associated Data modes AES CCM and AES GCM/GMAC can be constructed for their return by using the same cipher key in modes:

- AES CTR in ENC/DEC
- AES ECB in ENC
- AES CBC in ENC

Therefore to protect confidentiality of the internal authentication keys in AES CCM and AES GCM/GMAC when POLICY_OBJ_ALLOW_ENC is set, it SHALL be combined with FORBID_EXTERNAL_IV. When the policy POLICY_OBJ_ALLOW_DEC is set, or POLICY_OBJ_ALLOW_ENC without FORBID_EXTERNAL_IV, access to the cipher key MUST be restricted by setting at least one of the following:

- setting POLICY_OBJ_REQUIRE_SM
- configuring an Authentication Object for the operation
- mandating Platform SCP

INTERNAL NOTE: artf1299929

7.1.8 Transport Layer Security

A target object used in TLSCalculatePreMasterSecret with key exchange algorithm ECDHE_PSK cannot be fully protected against manipulations by setting ALLOW_DERIVED_INPUT, which is verified against only the key pair but not the HMACKey source object. Protection of such target object against manipulations could instead be achieved with POLICY_OBJ_REQUIRE_SM, with binding ALLOW_DERIVED_INPUT to an authentication object or by mandatory Platform SCP. For more details see [1]

INTERNAL NOTE: artf1262827

7.2 Extendibility and Multi-tenant

7.2.1 Transport lock use

7.2.1.1 Transport lock

An entity A can apply a lock on A5000 and share the key with entity B to give authorized access only to entity B. In this case entity B MAY be the final receiver of the devices

If a transport lock is expected in A5000 configuration, customer MUST verify that the lock is still applied upon receipt of A5000.

- In locked state, only GetVersion, GetUniqueID, GetRandom and CreateSession commands are allowed. Command ReadIDList should fail with response SW_COMMAND_NOT_ALLOWED.
- If an unexpected behavior is noticed, this MUST be reported to the entity which applied the transport lock.

If a transport lock is applied, A5000 MAY be unlocked. Unlocking response MUST be SW_NO_ERROR.

- Unlocking is achieved by authenticating to the reserved authentication object with identifier 0x7FFF0200. If this would fail or unexpected behavior is noticed, this MUST be reported to NXP.

7.2.1.2 Transport Lock Provisioning

These recommendations apply to customers or 3rd Party programming facilities who need transport lock to perform device provisioning.

The transport lock MAY be used as a tamper seal to distribute devices to other parties in a cascade logistic chain.

- In this scenario, the transport lock can be considered as seal, which hampers manipulations to the product during transport.

If more than one customer is intended to perform provisioning in the supply chain, each customer MUST update the transport lock.

- In this case the transport lock MUST be configured with write access policy.

7.2.2 UserID sessions

UserID sessions are not providing authentication functionality but are there to logically group Secure Objects.

7.2.2.1 UserID Secure Object

For secure use of UserID Secure Objects, the maximum authentication attempts TAG_MAX_ATTEMPTS MUST be set to a value different from zero.

- A UserID Secure Object TAG_MAX_ATTEMPTS with a value of zero means infinite authentication attempts. The UserID can be compromised by brute-force attack.

- Note that a TAG_MAX_ATTEMPTS with a value different from zero will cause a flash write at each UserID verification due to counter pre-decrement.

7.2.2.2 Security claims on user sessions

UserID sessions MUST NOT be used alone if confidentiality or integrity of communications are required.

- UserID sessions are not secure intrinsically as the UserID can be eavesdropped, and following communications are not encrypted.
- To ensure confidentiality and integrity of communications SCP03 or ECKey sessions MUST be used.

7.2.3 Secure messaging

If confidentiality is required on a secure object, the Secure Object policy MUST either have

- the rule POLICY_OBJ_REQUIRE_SM set or
- the Authentication Object ID referring to an existing key authentication object (AES128 key for an SCP03 session or ECCKey for a ECKey session) or
- Platform SCP has to be configured mandatorily.
- The highest security level MUST be set to enable C-ENC, C-MAC, R-ENC and R-MAC.

INTERNAL NOTE: artf1265971 - The minimum security level is C-MAC which only protects the command APDU against manipulation (integrity). Confidentiality of the command (C-ENC) and response (R-ENC) as well as response integrity (R-MAC) add the required maximim security.

The secure channel concept and the used security levels of the secure channel MUST be tailored to the use case, ideally selecting always the highest possible security level.

INTERNAL NOTE: artf1265113.

7.2.4 ECKey sessions

The ephemeral key to establish a secure session with ECKey MUST be secured at a similar level as the static authentication private key of the host. Therefore, it MUST be destroyed when the secure channel is no longer needed. Furthermore, a newly created ephemeral key MUST be used in the following cases:

- for every new established secure channel with ECKeySessionInternal Authenticate
- for every new import of a secure object using ImportExternalObject

To ensure fresh authentication of the A5000, one must enable R-MAC for the secure channel and validate the received MACs.

It is recommended to use attested read to retrieve the public key for the authentication of the SE or validate the public key with a certificate.

INTERNAL NOTE: artf1265589

7.2.5 Credentials provisioning

These recommendations apply to customers or 3rd Party programming facilities who perform provisioning

Note: Provisioning can be protected/restricted by establishing authenticated session with restricted access rights.

7.2.5.1 Remote provisioning

For transferring secret key during remote provisioning, application level SCP or Secure Object Import MUST be used.

- Use of application level SCP (SCP03 or ECKey sessions) or Secure Object Import in addition to Platform SCP is mandatory to ensure end-to-end confidentiality and integrity of secrets during remote provisioning.

7.2.5.2 Asymmetric keys and key pairs

Confidentiality, integrity and authenticity of key pairs that are provisioned into A5000 MUST be enforced as required for their use during provisioning and outside of A5000.

- Key pairs MAY be generated on-chip so that private keys can stay in A5000.
- Further security properties MUST be preserved by environmental measures.

On-chip key genererated EC keys MUST always be generated as key pairs, by setting P1KeyType to P1_KEY_PAIR.

INTERNAL NOTE: artf1330338

If only integrity and authenticity of key pairs that are provisioned into A5000 is required, attestation of the Secure Object with a provisioned attestation key MAY be used.

Key pairs provisioned into A5000 MUST be die-individual.

- Device individual key pairs hamper the exploitation of successful attacks on other devices.

INTERNAL NOTE: artf1268977

7.2.5.3 Symmetric keys

Confidentiality, integrity and authenticity of symmetric secrets that are provisioned into A5000 MUST be enforced as required for their use also during provisioning and outside A5000.

- A5000 supports Symmetric Keys with write of the wrapped key value according to RFC3394. Related unwrapping key must have policy POLICY_OBJ_ALLOW_RFC3394_UNWRAP set and MUST NOT have POLICY_OBJ_ALLOW_DEC set.
  Internal note: artf1272901
- Further security properties MUST be preserved by environmental measures.

If only integrity of symmetric keys that are provisioned into A5000 is required the attestation of the Secure Object with a provisioned attestation key MAY be used.

- The timestamp and freshness fields of the attestation must be checked according to Attestation.

When the use case allows it, symmetric secrets provisioned into A5000 SHOULD be die-individual.

- Symmetric Keys hamper exploitation of successful attacks on other devices.

7.2.6 General purpose storage

Integrity and authenticity of GP data that are provisioned into A5000 MUST be enforced as required for their use during provisioning and outside of A5000.

- Integrity of GP data that are provisioned into A5000 is supported by attestation of the GP data with a provisioned attestation key.
- Further security properties MUST be preserved by environmental measures.

8 Functional Recommendations

8.1 Wear-out prevention

NVM writes have the risk to wear out the flash and thus permanently make the device unusable).

The default configuration of the secure authenticator avoids as much as possible NVM writes: only when storing keys or files permanently into the device, flash write operations are done.

Creation and deletion of any Secure Object or Crypto Object is causing flash write operations. For transient Secure Objects and for Crypto Objects, any update of the value of the Secure Object is not causing additional flash write operations. For persistent Secure Objects, any update of the value of the Secure Object causes flash write operations.

Additional flash writes are done when users opt for putting a maximum number of authentication attempts on Authentication Objects. In that case, any authentication attempt is logged and causing additional flash write operations.

8.2 Power modes of A5000

A5000 supports the following kinds of power saving operations:

- “Off”: For this scenario, Vin is not supplied anymore. As a consequence, the IC looses all its internal states that are not yet persisted in NV memory. A full startup sequence needs to be executed.
- “Deep Power Down” via ENA pin: This mode is entered if the ENA pin is set to low potential. The behavior from IC point of view is identical to the power down mode described above. All transient states are lost.
- “Power Down”: When using I^2C interface, a command can be sent on the I^2C link to bring the device into a sleep mode. In Power Down mode, all transient states are kept and the communication can continue with the next APDU. When using the GlobalPlatform APDU Transport over I2C protocol refer to the S(RELEASE request) block in [10] as well as the Power Saving Timeout value (PST) as part of the CIP. When the NXP SE05x T=1 Over I2C protocol is used refer to the End of APDU session request S-Block in [11]
- Active mode: This mode is automatically entered when waiting for next command apdu.

Depending on the startup performance and power saving requirements one of the modes above should be chosen when the A5000 is not actively used.

8.2.1 Application Circuit Basic I²C usage

Configuration used:

- I²C from host to A5000
- A5000 turned off via V_{cc}
- Contactless and I²C controller not part of application schematic below
- Alternatively the V_{cc}-pin can be connected to the V_{in}-pin instead of the V_{out}-pin. The V_{out}-pin is not connected in this case.

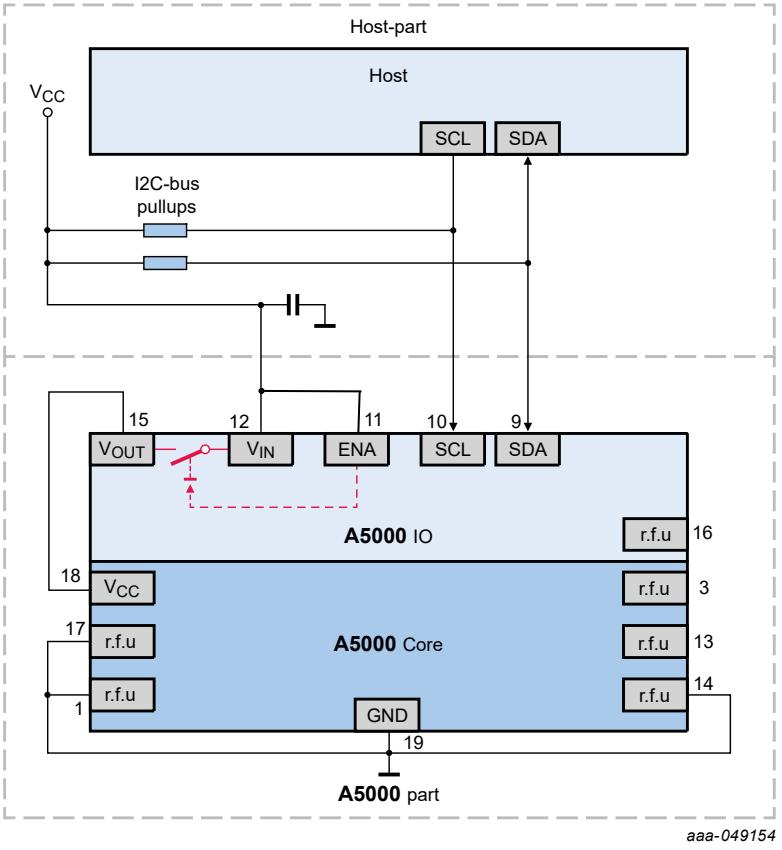

Figure 12. Application Circuit I²C

8.2.2 Application Circuit I²C with Deep Power Down

Configuration used:

- I²C from host to A5000
- A5000 turned off using Deep Power Down mode (ENA pin low)
- Contactless and I²C controller not part of application schematic below

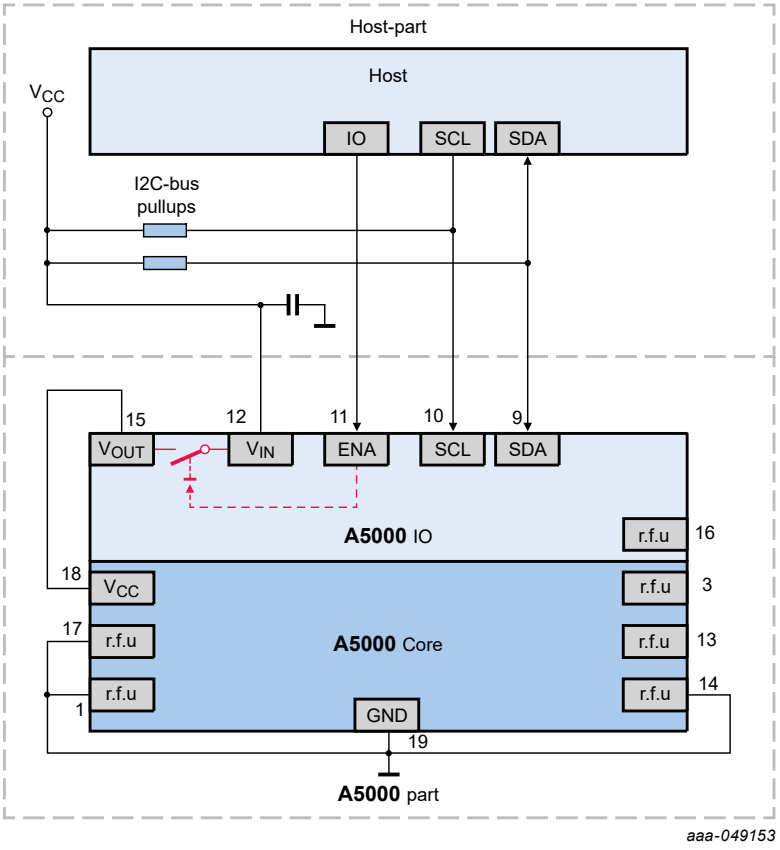

Figure 13. Application circuit I2C with Deep Power Down

No extra capacitor to be placed on the connection between V_{out} and V_{cc}.

9 References

[1] Application Note: A5000 Authentication Application APDU Specification, AN13187. Available under: https://www.nxp.com/docs/en/application-note/AN13187.pdf

[2] Datasheet: A5000, EdgeLock® Secure Authenticator, 6676xx. Available under: https://www.nxp.com/docs/en/data-sheet/A5000-DATASHEET.pdf

[3] Application Note: SE05x for secure connection to Azure IoT Hub, AN12402. Available under: https://www.nxp.com/docs/en/application-note/AN12402.pdf

[4] Application Note: SE05x for secure connection to AWS IoT Core, AN12404. Available under https://www.nxp.com/docs/en/application-note/AN12404.pdf

[5] Application Note: SE05x for secure connection to GCP, AN12401. Available under: https://www.nxp.com/docs/en/application-note/AN12401.pdf

[6] Application Note: SE05x for secure connection to IBM Watson IoT, AN12403. Available under: https://www.nxp.com/docs/en/application-note/AN12403.pdf

[7] Application Note: SE05x for device-to-device authentication, AN12399. Available under: https://www.nxp.com/docs/en/application-note/AN12399.pdf

[8] RFC2119. Available under: https://tools.ietf.org/html/rfc2119

[9] GlobalPlatform Card Specification Version 2.3.1, GPC_SPE_034. Available under: https://globalplatform.org/specs-library/card-specification-v2-3-1

[10] GlobalPlatform APDU Transport over SPI/I2C, Version 1.0, GPC_SPE_172. Available under: https://globalplatform.org/specs-library/apdu-transport-over-spi-i2c-v1-0-gpc_spe_172

[11] NXP SE05x T=1 Over SPI/I2C Specification, UM11225. Available under: https://www.nxp.com/webapp/Download?colCode=UM11225

[12] Application Note: EdgeLock™ SE05x Quick start guide with Visual Studio project examples, AN12398. Available under: https://www.nxp.com/docs/en/application-note/AN12398.pdf

10 Legal information

10.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

10.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com

Date of release: 18 November 2022
Document identifier: AN13266