webOS TV Secure Storage Requirement

Revision history:

Version	Date	Description
1.0	24 Jun 2022	Initial version
1.1	28 Jun 2022	Update key backup/restore section

Table of Contents

Documents:

- TEE Client API Specification v1.0
- TEE Internal Core API Specification v1.1.2
- TEE Internal API Specification v1.0
- TEE Protection Profile v1.2.1
- https://optee.readthedocs.io/en/latest/architecture/secure_storage.html

Secure storage

Secure Storage

Secure storage (a.k.a trusted storage) is used to securely store various kinds of data and keys in Trusted Application (TA) on TEE.

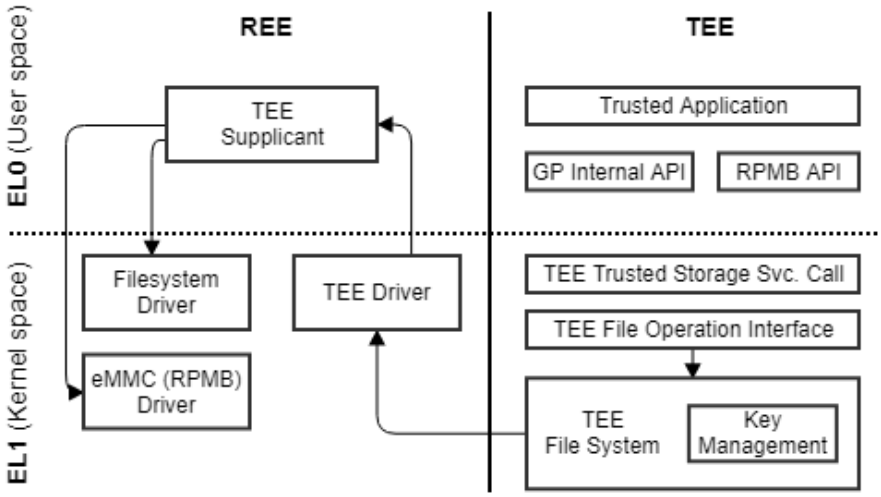
SoC vendor must provide secure storage functionality which complies (or equivalent to) GlobalPlatform TEE standard [**REQ-SS-001**].

> *GlobalPlatform TEE Protection Profile v1.2.1*
>
> *Trusted Storage*
> *Trusted storage (a.k.a secure storage, secure file system) indicates storage that is protected to at least the robustness level defined for OMTP Secure Storage (in section 5 of [OMTP-TR1]).*
> *It is protected either by the hardware of the TEE, or cryptographically by keys held in the TEE.*
> *If keys are used they are at least of the strength used to instantiate the TEE.*
> *A GlobalPlatform TEE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.*
>
> *Trusted Storage Security Objective*
> *The TEE shall provide Trusted Storage services for persistent TA general-purpose data and key material such that:*
>
> - *The confidentiality of the stored data and keys is enforced*
> - *The authenticity of the stored data and keys is enforced*

- *The consistency of each TA stored data and keys is enforced*
- *The atomicity of the operations that modify the storage is enforced.*

The TEE Trusted Storage shall be bound to the hosting device, which means that the storage space must be accessible or modifiable only by authorized TAs running on the same TEE and device as when the data was created.

Typically TEE OS doesn't have eMMC controller driver in TEE boundary, so SoC vendor must implement tee-supplicant module which runs in REE to access the file system and RPMB instead of TEE OS [**REQ-SS-001**].



Each TA's secure store must be cryptographically isolated from other TA's secure store [**REQ-SS-002**].

SoC vendor must provide secure storage isolation which complies (or equivalent to) OP-TEE implementation.

> ***OP-TEE Secure storage***
>
> *Key Manager*
>
> *Key manager is a component in TEE file system, and is responsible for handling data encryption and decryption and also management of the sensitive key materials. There are three types of keys used by the key manager: the Secure Storage Key (SSK), the TA Storage Key (TSK) and the File Encryption Key (FEK).*
>
> *Secure Storage Key (SSK)*
>
> *SSK is a per-device key and is generated and stored in secure memory when OP-TEE is booting. SSK is used to derive the TA Storage Key (TSK).*
>
> *SSK is derived by*
>
> > $SSK = HMAC_{SHA256}(HUK, Chip\ ID \parallel \text{“static string”})$
>
> *The functions to get Hardware Unique Key (HUK) and chip ID depends on the platform implementation. Currently, in OP-TEE OS we only have a per-device key, SSK, which is used for secure storage subsystem, but, for the future we might need to create different per-device keys for different subsystems using the same algorithm as we generate the SSK; An easy way to generate different per-device keys for different subsystems is using different static strings to generate the keys.*
>
> *Trusted Application Storage Key (TSK)*
>
> *The TSK is a per-Trusted Application key, which is generated from the SSK and the TA's identifier (UUID). It is used to protect the FEK, in other words, to encrypt/decrypt the FEK.*
>
> *TSK is derived by:*
>
> > $TSK = HMAC_{SHA256}(SSK, TA\_UUID)$
>
> *File Encryption Key (FEK)*
>
> *When a new TEE file is created, key manager will generate a new FEK by PRNG (pesudo random number generator) for the TEE file and store the encrypted FEK in meta file. FEK is used for encrypting/decrypting the TEE file information stored in meta file or the data stored in block file.*

RPMB

RPMB (Replay Protected Memory Block) is a separate physical partition in the eMMC device designed for secure data storage.

Every access to RPMB is authenticated and it allows the host to store data in RPMB partition in authenticated and replay protected manner.

RPMB uses HMAC to authenticate request and response between host and eMMC driver stack.

RPMB key which is used to generate HMAC must be derived from HUK, and it must be programmed in eMMC securely and never be exposed to REE [**REQ-SS-003**].

RPMB packet must be generated inside TEE, that is, SoC vendor must not expose any interface to generate HMAC from REE [**REQ-SS-004**].

Because RPMB ensures only authenticity for a packet, SoC vendor must ensure confidentiality and integrity of data stored in RPMB which is equivalent to secure store key management [**REQ-SS-005**].

RPMB data which is stored by a TA must be cryptographically isolated from other TA's RPMB data [**REQ-SS-005**].

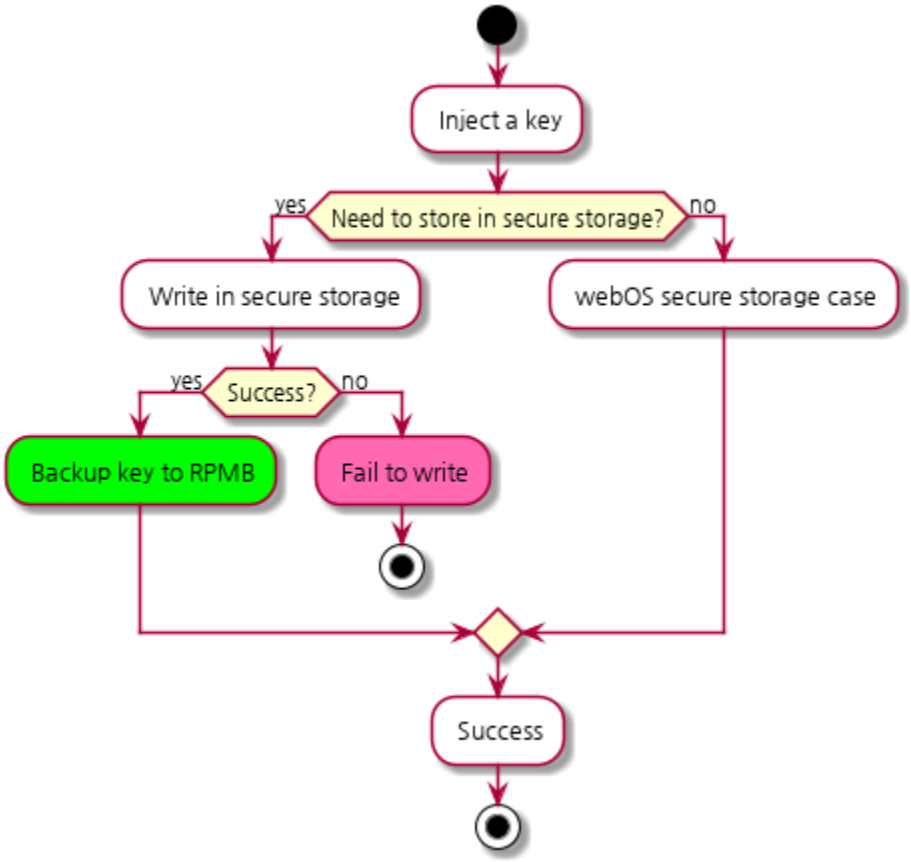
When store a key in secure storage, the same key must also be stored in RPBM to support backup and restore. Refer backup and restore section for details.

Key backup and restore

Provisioned keys are critical asset of webOS TV, so SoC vendor must provide key backup and restoration feature [**REQ-SS-006**].

Backup

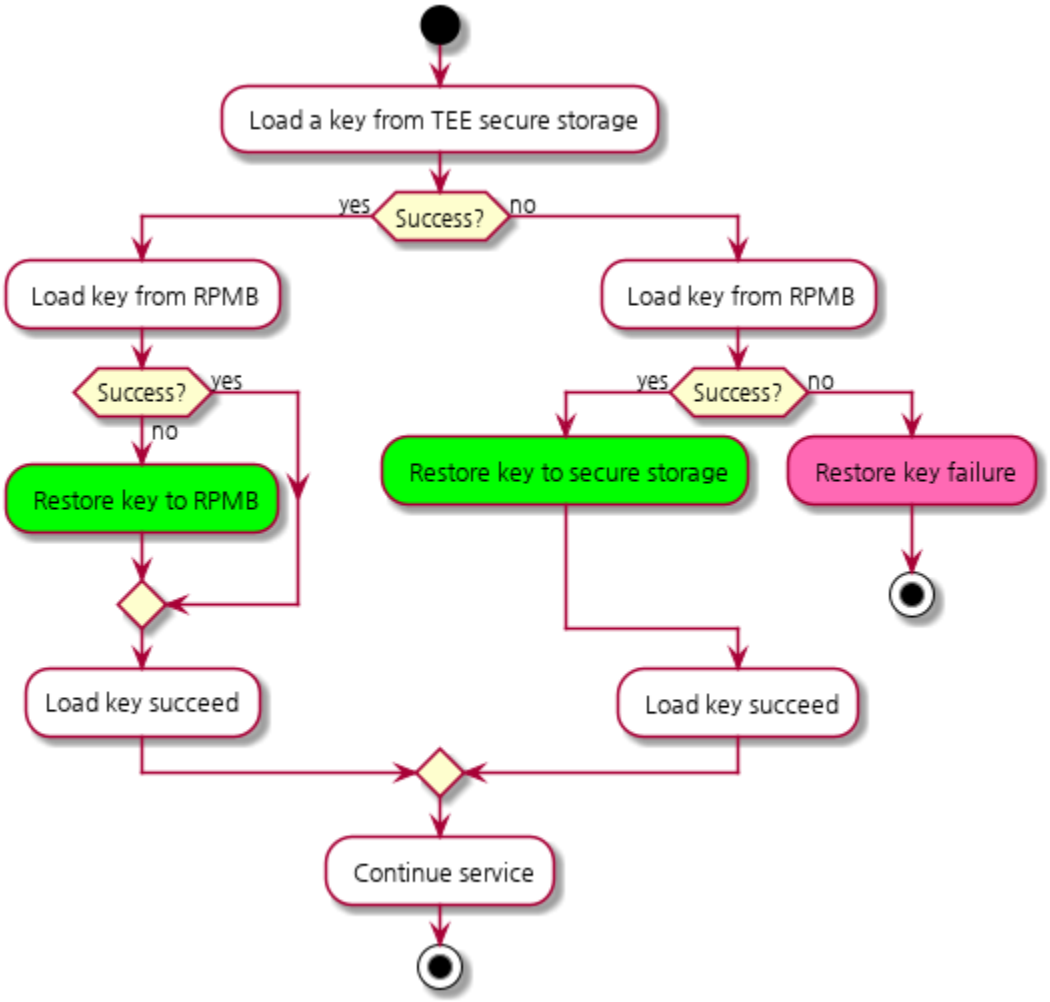
When storing a key in secure storage, the same key must also be stored in RPBM to support backup and restore as following diagram.



Restore

As following diagram, provisioned keys are stored in secure storage and RPMB respectively.

These provisioned keys will be used by specific webOS services and if provisioned key is unavailable from one area for any reason, key restoration must be conducted using the key stored in another area.



webOS TV keys

webOS TV platform has several types of keys and data as listed in below table and secure storage (except SEDATA and GENERAL) is used to protect provisioned data.

Key category	Key type	Key Injection	Secure storage type	REE INPUT	REE OUTPUT	Note
Device unique key	**NONE**	Mass production line	Secure storage and RPMB	Solution specific form	None	During mass production, device unique keys are injected into specific TA (each device has different key). Provisioned keys MUST be accessed only from specific TA and never exposed to REE.
Common key	**SEDATA_TEE**	Device boot up	Secure storage and RPMB	Encrypted key	None	SEDATA_TEE key (PlayReady certificate and key) are injected into specific TA during device boot up. Provisioned keys MUST be accessed only from specific TA and never exposed to REE.
	KEY_ID	Device boot up	Secure storage and RPMB	Encrypted key	None	KEY_ID keys are injected into specific TAs during device boot up. Provisioned keys can be shared to other TA through Inter-TA interface (TEE_OpenTASession) Provisioned keys SHOULD NOT be exposed to REE.
	SEDATA	Device boot up	webOS secure storage	Encrypted key	Secure Data	SEDATA keys are injected into secure store TA during device boot up.
Runtime key	**GENERAL**	Runtime	webOS secure storage	Raw data	Secure Data	Data are injected into secure store TA during runtime.

Device unique key

Device unique key must be injected only into intended TA.

TA must store provisioned key in secure storage and RPMB [**REQ-SS-006**].

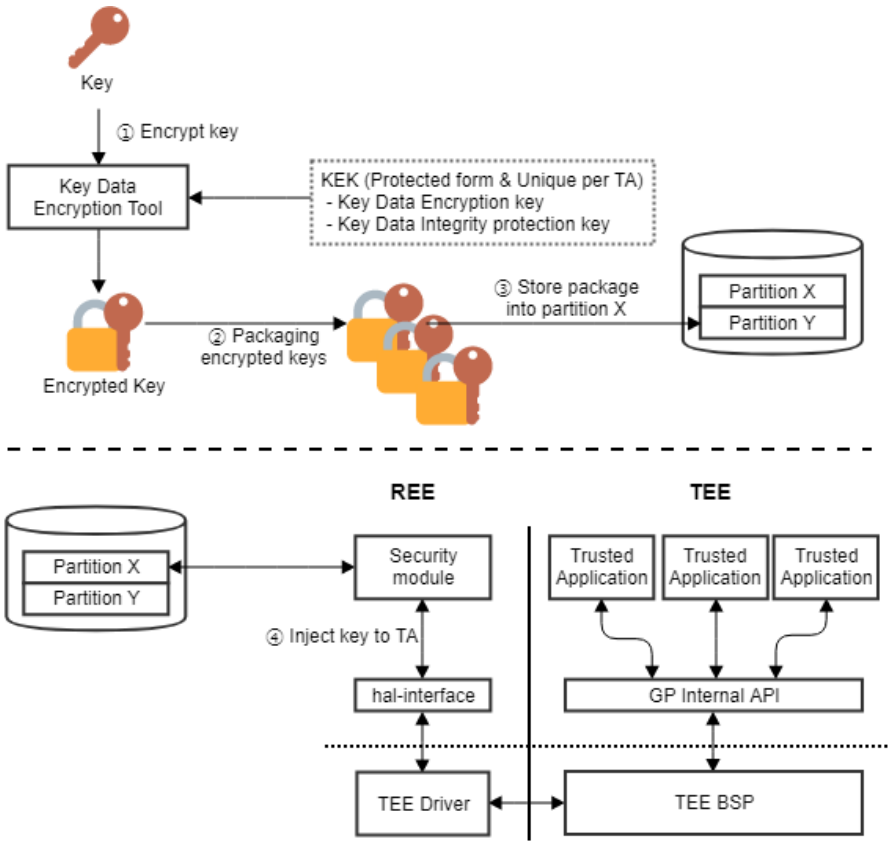
TA must not expose provisioned key to REE and must not share it with other TA through Inter-TA [**REQ-KP-001**].

Common key

Since the device unique key is unique for each device, it is injected in the mass production line, whereas the common key shares the same key for the same product, so the injection method is different from the device unique key.

Common keys are pre-encrypted using the key data encryption tool and pre-encrypted keys are stored in device's specific partition.

During device's boot up, security module loads pre-encrypted keys from the partition and inject those into Trusted Application on TEE.



Common keys are pre-encrypted by the key data encryption tool and decrypted in specific TA after injection.

So SoC vendor must provide a way to share the same KEK (key encryption key, KEK consist of encryption key and integrity protection key) between key data encryption tool and a TA.

SoC vendor must use different key for encryption and integrity protection [**REQ-KP-002**].

Even though key data encryption tool is running on Linux host machine (ex, x86) with limited access, internal developers can easily access encryption tool.

So SoC vendor must protect KEK which is used by encryption tool, that is, KEK is never exposed in plain text form (ex, tool's source code or separate binary file without protection) [**REQ-KP-003**].

Based on common key's purpose, common keys can be stored in different TAs (refer TA design option for details).

So KEK must be unique per TA, that is, each TA must have unique KEK [**REQ-KP-002**].

Common key decryption

Security module loads pre-encrypted keys from the partition and inject those into TA.

TA must check integrity of pre-encrypted key using integrity protection key first and then decrypt it using encryption key.

When storing device common key, SoC vendor must comply requirements noted in TA design option.

Runtime key

Runtime keys are generated by several REE components and delivered to a TA for data protection as plain text form.

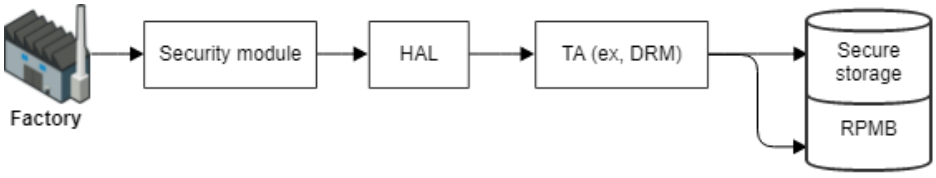
SoC vendor must comply requirements noted in TA design option.

TA design option

SoC vendor must comply TA design option [**REQ-KP-004**].

Device unique key

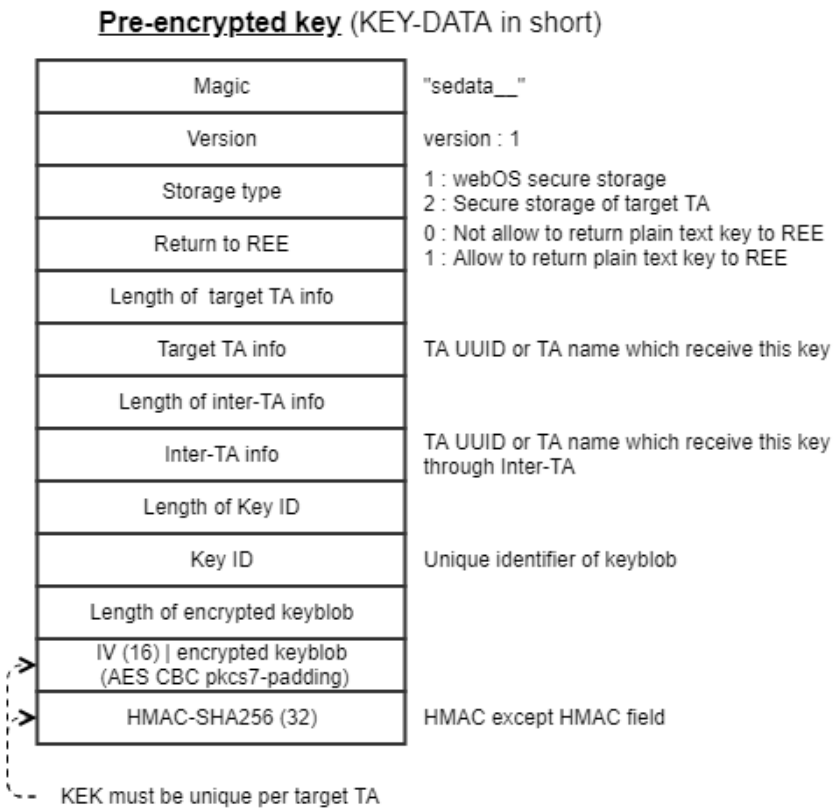
Device unique key must be injected only into intended TA and there is no design option [**REQ-KP-005**].



Common key

In order to sustain security properties of the key from key data generation to key data storage, pre-encrypted key has various metadata.

Target TA info is mandatory field and only TA which is matched with target TA info can access pre-encrypted key.



SEDATA_TEE

There exist only one SEDATA_TEE type key (PlayReady certificate and private key) and SEDATA_TEE key must be stored in PlayReady TA [**REQ-KP-005**].

SoC vendor must follow the requirement of device unique key.

Metadata

SoC vendor must use following metadata for SEDATA_TEE key [**REQ-KP-007**].

Field	Value
Storage type	2 - Must be stored in TA secure storage and RPMB
Return to REE	0 - Not allow to return plain text key to REE
Target TA info	TA Name (ex. PlayReady)
Inter-TA info	Empty - Not allow to share the key through Inter-TA

KEY_ID

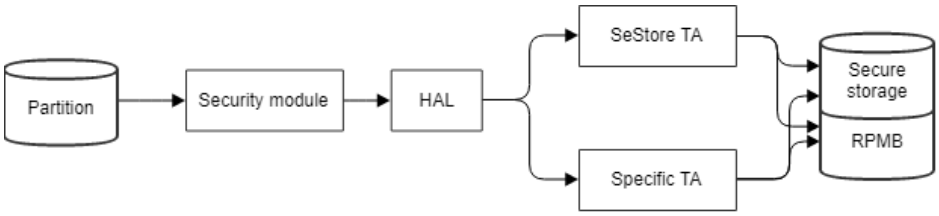
If a KEY_ID type key is intended for only specific TA (used by only a TA), SoC vendor must store the key in that TA [**REQ-KP-001**][**REQ-KP-006**].

(ex, If exist HDCP TA, then HDCP key must be injected only to HDCP TA.)

If there doesn't exit specific TA which handles a KEY_ID type key, it can be stored in SeStore TA (Common TA).

(ex, a Key needs to be stored securely but doesn't have huge requirement which require to develop separate TA)

Regardless of which TA stores the key, KEY_ID type key must be stored in secure storage and RPMB [**REQ-SS-006**].



Metadata for specific TA

If KEY_ID key is stored in specific TA, SoC vendor must use following metadata [**REQ-KP-007**].

Field	Value
Storage type	2 - Must be stored in TA secure storage and RPMB
Return to REE	0 - Not allow to return plain text key to REE
Target TA info	Specific TA Name (ex. hdcp)
Inter-TA info	Empty - Not allow to share the key through Inter-TA

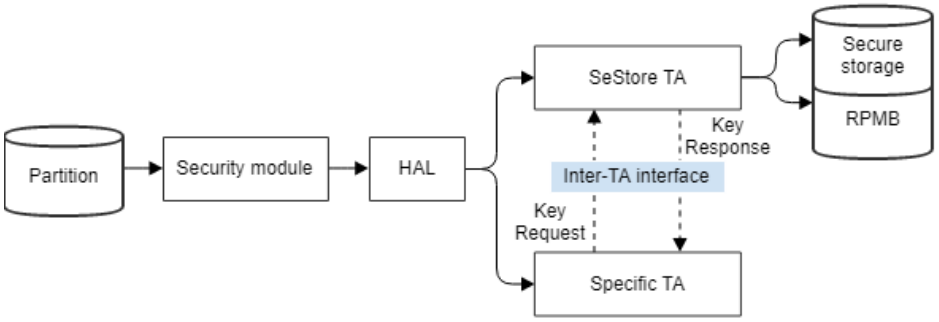
Metadata for SeStore TA

If KEY_ID key is stored in SeStore TA, SoC vendor must use following metadata [**REQ-KP-007**].

Field	Value
Storage type	2 - Must be stored in TA secure storage and RPMB
Return to REE	[Conditional] 0 - Strongly recommend not to allow returning plain text key to REE 1 - Need to provide dedicated REE (HAL) interface per plain text key Need to apply access control
Target TA info	SeStore TA Name
Inter-TA info	[Conditional] Empty if not use Inter-TA key distribution

Runtime Key distribution through Inter-TA

For somewhat reason, SoC vendor may decide to store some of keys in SeStore (common) TA and distribute those keys through Inter-TA in runtime.



This design option is not recommend but if select, SoC vendor must describe specific TA's information in metadata's Inter-TA info field [**REQ-KP-007**].

When SeStore TA receive key distribution request from other TA, SeStore TA must check whether the caller TA's identity is same with Inter-TA info in metadata [**REQ-KP-008**].

SoC vendor must provide a way to check caller TA's identity securely [**REQ-KP-008**].

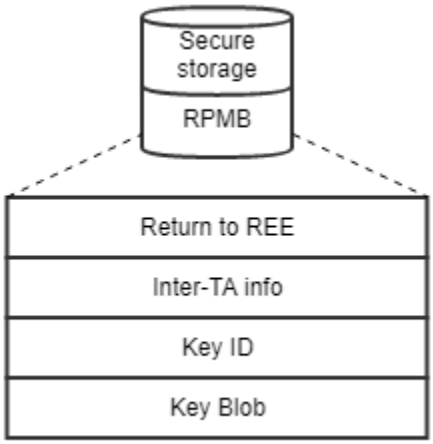
SoC vendor can use GlobalPlatform TEE API or any other equivalent access control mechanism.

```
TEE_GetPropertyAsIdentity(TEE_PROPSET_CURRENT_CLIENT, "gpd.client.identity", &identity);
```

Secure storage data format of SeStore TA

Even though SeStore TA uses secure storage and RPMB to store injected keys, there exists several design options that can lead security risk.

To minimize the risk, SoC vendor must store additional metadata with the key in secure storage [**REQ-KP-007**].



SEDATA

SEDATA type key is a key which requires data at rest protection, but not require data in use protection.

That is, SEDATA key is cryptographically protected by SeStore TA but raw key is shared to REE during runtime.

SEDATA is pre-encrypted keys and injected into SeStore TA.

After SeStore TA validates and decrypts pre-encrypted key using KEK, SeStore TA must encrypt raw key and metadata using new encryption key.

Newly encrypted key (a.k.a Secure Data) is returned to REE and stored in dedicated partition by security module.

Metadata

SoC vendor must use following metadata for SEDATA key [**REQ-KP-007**].

Field	Value
Storage type	1 - Will be stored in webOS secure storage
Return to REE	1 - Allow to return plain text key to REE
Target TA info	SeStore TA Name
Inter-TA info	Empty - Not allow to share the key through Inter-TA

To protect Secure Data, SoC vendor must comply following key hierarchy [**REQ-KP-008**].

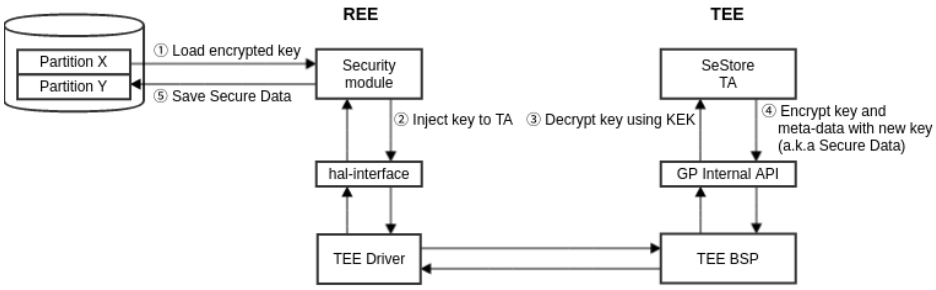
DUK is device unique key and used to protected SDEK and SDHK.

SoC vendor must ensure that DUK is unique per device and SoC vendor can derive DUK from HUK.

SDEK is secure data encryption key and randomly generated per a SEDATA.

SDHK is secure data HMAC key and randomly generated per a SEDATA.

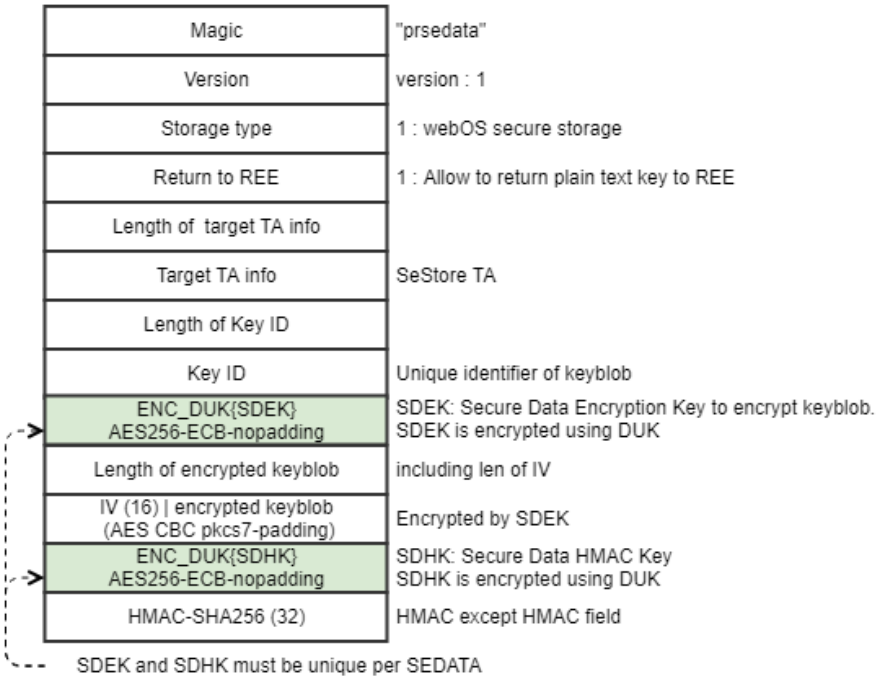
SoC vendor must ensure confidentiality using SDEK and integrity using SDHK of Secure Data.



Secure Data format

Because Secure Data is encrypted and accessed only from SeStore TA, target TA info in metadata must be SeStore TA [**REQ-KP-007**].

SoC vendor must generate Secure Data only when metadata of KEY-DATA is Storage type:1, Return to REE:1 and Target TA info:SeStore TA [**REQ-KP-009**].



Runtime key

Follow SEDATA design option except input data to SeStore TA is plain text and Key ID is empty.

Secure storage Security Requirement

Secure storage

REQ-ID	Requirement
REQ-SS-001	MUST support secure storage functionality which complies (or equivalent to) GlobalPlatform TEE standard. MUST support tee-supplicant module which runs in REE to access the file system and RPMB instead of TEE OS.

REQ-SS-002	MUST support cryptographic isolation for secure storage of each TA and it MUST complies (or equivalent to) OP-TEE implementation. MUST provide a document which describes details (key length, algorithm and key hierarchy) of cryptographic keys. MUST use AES256 CBC/CTR with random IV or AES256 GCM (confidentiality), and HMAC-SHA256 (integrity) at least.
REQ-SS-003	MUST derive RPMB key from HUK, program it in eMMC securely and never expose it to REE.
REQ-SS-004	MUST not expose any REE interface which allow to generate HMAC for RPMB packet.
REQ-SS-005	MUST support confidentiality and integrity of data stored in RPMB. MUST use AES256 CBC/CTR with random IV or AES256 GCM (confidentiality), and HMAC-SHA256 (integrity) at least. MUST support cryptographic isolation for RPMB data of each TA. MUST provide a document which describes details (key length, algorithm and key hierarchy) of cryptographic keys.
REQ-SS-006	MUST support key backup and restore feature.
REQ-SS-007	MUST not erase provisioned keys during factory data reset.

Key Provisioning

REQ-ID	Requirement
REQ-KP-001	Keys stored in specific TA except SeStore TA MUST not be shared (exposed) to REE and MUST not support runtime key distribution through Inter-TA.
REQ-KP-002	MUST use unique KEK (encryption key and integrity protection key) per TA. MUST use different key for encryption and integrity protection. MUST use AES256 CBC/CTR with random IV or AES256 GCM (confidentiality), and HMAC-SHA256 (integrity) at least. MUST provide a document which describes details (key length, algorithm and key hierarchy) of cryptographic keys.
REQ-KP-003	MUST not expose KEK used by key data encryption tool in plain text form. MUST provide a document how the KEK is protected in key data encryption tool.
REQ-KP-004	MUST comply TA design option
REQ-KP-005	MUST provision device unique key and SEDATA_TEE key to intended TA (not SeStore TA).
REQ-KP-006	MUST provision a key to specific TA if exist dedicated TA for the key.
REQ-KP-007	MUST comply with key metadata requirement.
REQ-KP-008	SHOULD not support runtime key distribution through Inter-TA. MUST support a way to check caller's identity and do access control based on metadata if support runtime key distribution.
REQ-KP-008	MUST comply with SEDATA key requirement. MUST use AES256 CBC/CTR with random IV or AES256 GCM (confidentiality), and HMAC-SHA256 (integrity) at least.
REQ-KP-009	MUST generate Secure Data only when metadata requirement it met.