Linaro connect
San Francisco 2017

Secure storage in OP-TEE

Jens Wiklander

Linaro
connect
San Francisco 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- What is secure storage?
- Timeline - secure storage improvements
- Encryption keys
- Secure Object
  - Hash tree
  - Hash tree header
  - Data block encryption
  - Atomic updates
- Object list
- Anti-rollback with RPMB
- What’s next?

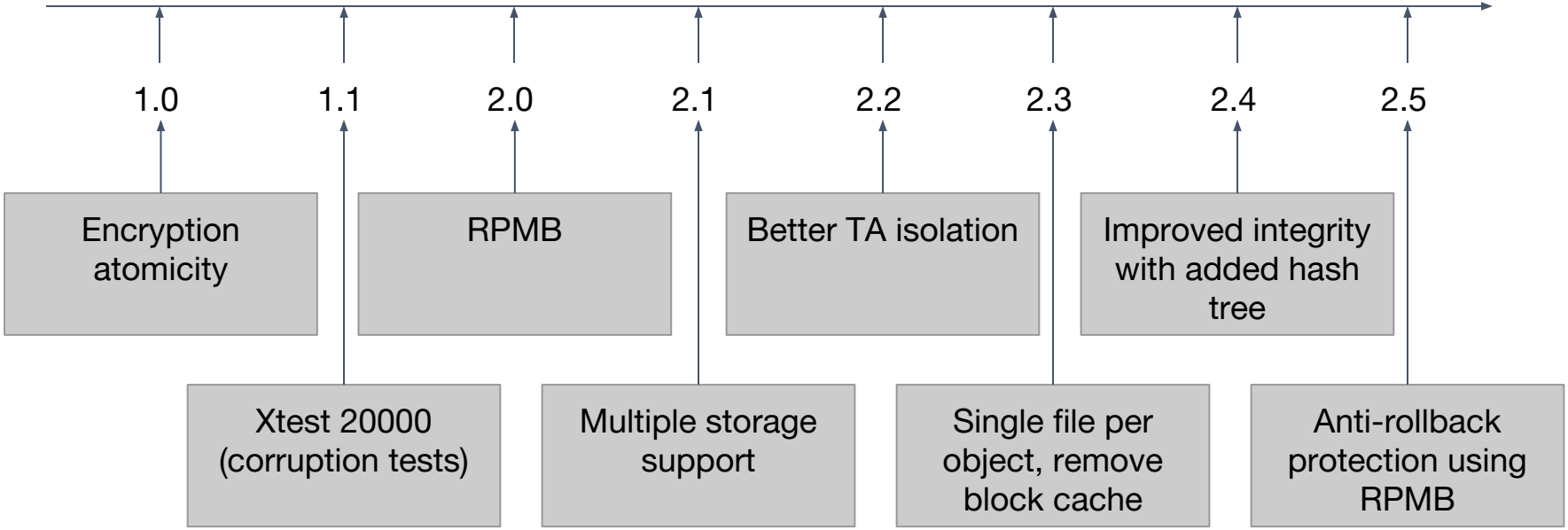
What is Secure Storage?

- Persistent data store for crypto keys or other application-specific data
- Accessible to Trusted Applications only
  - Each TA has its own storage (TA isolation)
- Isolated from the non-secure world
  - Secure Storage data can’t be read, modified or deleted by user applications or the OS kernel
- OP-TEE implements the GlobalPlatform™ TEE Internal Core API v1.1
  - Chapter 5: Trusted Storage API for Data and Keys ; Persistent Object [Enumeration] Functions and Data Stream Access Functions

TEE_OpenPersistentObject()
TEE_CreatePersistentObject()
TEE_CloseAndDeletePersistentObject1()
TEE_RenamePersistentObject()
TEE_ReadObjectData()
TEE_WriteObjectData()
TEE_TruncateObjectData()
TEE_SeekObjectData()

TEE_AllocatePersistentObjectEnumerator()
TEE_FreePersistentObjectEnumerator()
TEE_ResetPersistentObjectEnumerator()
TEE_StartPersistentObjectEnumerator()
TEE_GetNextPersistentObject()

Timeline - secure storage improvements

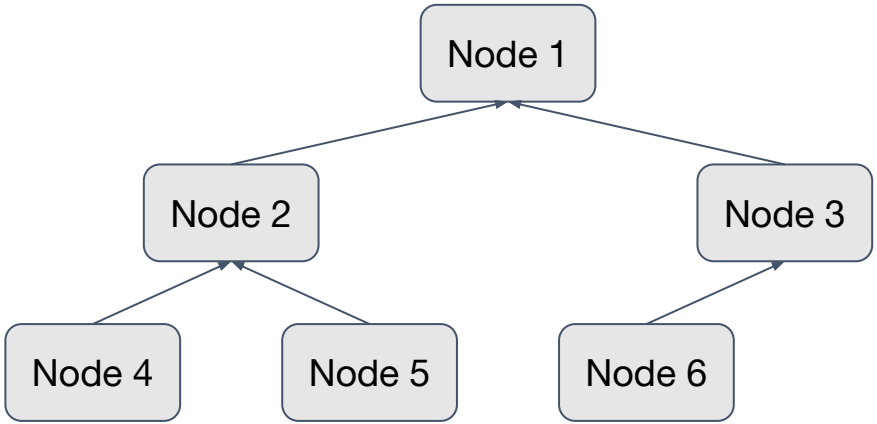


Encryption keys

- Authenticated block encryption (AES-GCM), one File Encryption Key (FEK) per file
- FEK is AES-encrypted using a 256-bit Trusted application storage Key (TSK) then stored in the metadata of the file
- TSK is derived from Secure Storage Key (SSK) and the Trusted Application UUID using HMAC-SHA256
- SSK is derived from a Hardware Unique Key (HUK) and a constant string using HMAC-SHA256

Hash tree

- A **complete** binary tree
- Each node protects one data block (tag and IV from AES-GCM operation)
- Hash calculated as: SHA-256(tag || IV || flags || $hash_{child0}$ || $hash_{child1}$)
- Nodes start counting from 1 and data blocks from 0
  - this means that node 1 holds tag and IV for data block 0



Hash tree header

- File encryption key, FEK_C
- Counter
  - In case of no rollback protection, this is used to select the latest version
- Metadata
  - Number of nodes
  - Payload length

Hash tree header continued

The hash tree header is constructed with the following steps

1. A new Initialization Vector, IV, is initialized from RNG
2. If FEK_C is not initialized yet, FEK_P is initialized from RNG and Encrypted FEK, $FEK_C = AES\text{-}ECB_{ENC}(SSK, FEK_P)$
3. Counter is increased by one or if not yet initialized set to 1
4. AAD = Node1.hash || Counter || FEK_C || IV
5. $(Tag, Metadata_C) = AES\text{-}GCM_{ENC}(FEK_P, IV, Metadata, AAD)$
6. The header is finally assembled as:
   IV || Tag || FEK_C || $Metadata_C$ || Counter

Data block encryption

An encrypted data block, C, is constructed with the following steps

1. A new Initialization Vector, IV, is initialized from RNG
2. P, is the unencrypted block of data
3. FEK = AES-ECB$_{DEC}$(SSK, FEK$_C$)
4. Additional Authenticated Data, AAD = FEK$_C$ || IV
5. Tag and C is produced with: (Tag, C) = AES-GCM(FEK$_P$, IV, P, AAD)

Tag and IV are saved in the node protecting the encrypted data block.

Atomic updates

- All parts of a secure storage object exists in two backup versions, 0 and 1
- The backup version in use is called the active version and the other version the inactive version
- The different parts are
  - Hash tree header
  - Hash tree node
  - Encrypted Data block
- All updates are done in the inactive versions until finally the hash of the inactive node1 has been written into the object list database

Object list database

Secure objects created by Trusted Applications, TAs, are indexed in a special secure object

UUID of TA	Uniquely identifies a secure object
Object Identifier	
Hash	Hash of node1 in secure object
file_number	Global unique file number

Object list database continued

- The object list database is stored under the name “dirf.db” in normal world
- If RPMB is available the hash of node1 is stored in RPMB and has full anti-rollback protection
- If RPMB is unavailable only a consistent state of all objects can be provided, that is, rollback can’t be applied on a single object

ENGINEERS
AND DEVICES
WORKING
TOGETHER

What's next?

- RPMB: don't program key unless some debug/testing CFG_ is set
- Improve derivation of SSK from HUK
  - Should be done by the hardware crypto module. HUK should never be read by software.
  - Unfortunately, we have no such driver upstream :(
- Reduce heap usage
  - Large objects uses much memory for nodes
- Storing Trusted Applications in secure storage

Linaro connect
San Francisco 2017

Thank You

#SFO17

SFO17 keynotes and videos on: connect.linaro.org

For further information: www.linaro.org