PSA Storage

• Provides a key/value storage interface to access device-protected storage
• Simple access control: each partition can access only its own assets
• Easy-to-use APIs:  
  
  ```c
  psa_status_t psa_its_set(psa_storage_uid_t uid, 
                          size_t data_length, 
                          const void *p_data, 
                          psa_storage_create_flags_t create_flags)
  
  psa_status_t psa_its_get(psa_storage_uid_t uid, 
                          size_t data_offset, 
                          size_t data_size, 
                          void *p_data, 
                          size_t *p_data_length)
  
  psa_status_t psa_its_get_info(psa_storage_uid_t uid, 
                                struct psa_storage_info_t *p_info)
  
  psa_status_t psa_its_remove(psa_storage_uid_t uid)
  ```

• Two varieties:  
  • Internal storage provided by the PSA Root of Trust: PSA Internal Trusted Storage  
  • External storage protected by the Application Root of Trust: PSA Protected Storage
ITS vs PS

PSA Internal Trusted Storage (ITS)

- Internal storage only
- Storage is inherently trusted: no encryption, authentication or rollback protection required in service itself
- Small datasets (e.g. keys)
- PSA RoT Service
- Implemented by TF-M ITS service

PSA Protected Storage (PS)

- Can use external storage
- Storage may be accessible to attacker: option for encryption, authentication and rollback protection in service
- Large datasets
- Application RoT Service
- Implemented by TF-M Secure Storage (SST) service
TF-M Secure Storage

• SST and ITS services each provided by their own partition in TF-M
  • ITS is PSA RoT, SST is Application RoT
  • SST depends on Crypto, which depends on ITS

• Both services use same lightweight flash filesystem as backend
  • Non-hierarchical, integer file IDs, create/write/delete APIs
  • Reliability in case of power failure
    – Can use 2 or >=4 flash blocks
  • No fragmentation
  • Flash layer can use internal or external flash device

• ITS is smallest possible wrapper around FS
  • Main addition is access control based on client IDs

• SST also adds protection for data-at-rest
  • Encryption, authentication, rollback protection
    – Controlled by build flags, depending on required level of protection
  • Authentication & encryption: AEAD (AES-128-GCM) using HUK, via Crypto service
  • Rollback protection: collect MACs in table, keep version in NV counter
Upcoming features

• Sharing common filesystem code between ITS and PS
  • SST calls ITS APIs as its backend ‘filesystem’
  • SST partition essentially becomes an encryption, authentication and rollback protection layer on top of ITS
  • Shrinks the stack size of SST, at cost of concurrent requests to ITS/PS APIs requests having to wait

• Protected Storage 1.0

• New HUK management design, using Crypto service

• Smaller internal buffers
  • Support for different profiles

• Key diversification
  • One key per client, or per asset
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
tודה