TF-M External Trusted Secure Storage Proposal

Poppy Wu
Julien Su
Macronix International

Jun 23, 2022
Agenda

• PSA storage API spec review
• External trusted secure storage discussion
• Current external trusted secure storage implementation introduction
Agenda

• PSA storage API spec review
• External trusted secure storage discussion
• Current external trusted secure storage implementation introduction
PSA storage APIs spec review

Modern embedded platforms have multiple types of storage, each with different security properties.
- on-chip flash storage
- external storage that requires confidentiality, integrity, and replay protection from attackers with physical access to the device

The PSA Storage APIs provide key/value storage interfaces for use with device-protected storage.

- PSA Internal Trusted Storage API
  - It is intended to be used for assets that must be placed inside internal flash. Some examples of assets that require this are replay protection values for external storage and keys for use by components of the PSA Root of Trust.
  - PSA ITS APIs:
    - psa_its_set()
    - psa_its_get()
    - psa_its_get_info()
    - psa_its_remove()
The PSA Storage APIs provide key/value storage interfaces for use with device-protected storage.

- **PSA Protected Storage API**
  - It is intended to be used to protect storage media that are external to the MCU package, with a promise of data-at-rest protection, including device-bound encryption, integrity, and replay protection.
  - PSA PS APIs:
    - `psa_ps_set()`
    - `psa_ps_get()`
    - `psa_ps_get_info()`
    - `psa_ps_remove()`
    - `psa_ps_create()`
    - `psa_ps_set_extended()`
    - `psa_ps_get_support()`
Agenda

• PSA storage API spec review
• **External trusted secure storage discussion**
• Current external trusted secure storage implementation introduction
External trusted secure storage

- The third storage - External trusted secure storage
  - Unique identity
  - Mutual authenticated read/write between host and external trusted storage
  - Dynamic encrypted transaction on the interface bus against snooping
  - Hardware protection against tampering
  - Other security functions

- PSA Storage APIs spec hasn’t discussed this kind of storage yet.
# External trusted secure storage

## External trusted secure storage (ETSS) APIs

<table>
<thead>
<tr>
<th>PSA ITS APIs</th>
<th>PSA PS APIs</th>
<th>ETSS APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>psa_its_set()</td>
<td>psa_ps_set()</td>
<td>etss_set()</td>
</tr>
<tr>
<td>psa_its_get()</td>
<td>psa_ps_get()</td>
<td>etss_get()</td>
</tr>
<tr>
<td>psa_its_get_info()</td>
<td>psa_ps_get_info()</td>
<td>etss_get_info()</td>
</tr>
<tr>
<td>psa_its_remove()</td>
<td>psa_ps_remove()</td>
<td>etss_remove()</td>
</tr>
<tr>
<td></td>
<td>psa_ps_create()</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>psa_ps_set_extended()</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>psa_ps_get_support()</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TBD</td>
</tr>
</tbody>
</table>
External trusted secure storage (ETSS) requirements

1. The technology and techniques used by the ETSS service MUST allow for frequent writes and data updates.
2. The storage underlying the ETSS MUST support cryptographic functions to provide encryption, authentication, integrity or replay protection.
3. The storage underlying the ETSS MUST keep cryptographic keys secure.
4. The storage underlying the ETSS MAY support cryptographic keys secure update.
5. The storage underlying the ETSS MAY have several isolated regions.
6. The ETSS service MUST use the partition ID information associated with each request for its access control mechanism.
7. The ETSS service MUST provide protection from one PSA partition accessing the storage assets of a different partition.
8. The creation of a uid with value 0 (zero) must be treated as an error
9. TBD
Agenda

• PSA storage API spec review
• External trusted secure storage discussion
• Current external trusted secure storage implementation introduction
External trusted secure storage implementation

An example of external trusted secure storage medium

- Encryption/decryption/authentication/integrity
- High entropy random number
- Replay protection
- Secure storage area is partitioned into several isolated data zones configured with different access control, support multi-tenant isolation
- Security keys never leave
- Physical unique identity

Pre-provisioned secret keys
Bind individual secure zone with specific secret key
External trusted secure storage implementation

External Trusted Secure Storage (ETSS) partition
External trusted secure storage implementation

ETSS partition framework
External trusted secure storage implementation

**ETSS services**

- Factory provisioning
  - Security keys
  - Secure Flash configuration
  - Application-zone binding
  - ...

- Secure storage services
  - Reuse ITS filesystem based on secure Flash operations

- Secure services
  - Based on secure Flash security features
  - Monotonic counter increase/get
  - Generate random number
  - Derive unique value from PUF

---

**Factory provisioning**

**Field deployment**
External trusted secure storage implementation

Factory provisioning

Provisioning blob example
• Keys derive info
  
  ```
  [ 
    { key_id, 
      key_info, 
      key_salt },
    ...
  ]
  
  • Secure Flash
    configure info
  • Application-zone
    binding info
  • ...
  ```

PSA import key
store provision data

Device

Crypto Partition

MCU

Secure Flash

Factory tool

Provisioning blob
External trusted secure storage implementation

Factory provisioning

- Keys derive info
- Secure Flash configure info
- Application-DataZone binding info [ { app_id, key_id, zone_id, counter_id}, ...
  ...

Provisioning blob example

Application-1

Data Zone-1

Application-2

Data Zone-2

Application-3

Data Zone-3

Applications

Secure Flash

Application-DataZone binding example
External trusted secure storage implementation

Field deployment

Application-1

client_id

etss_set

filesystem context

secure Flash framework

key_id/zone_id/counter_id

Data Zone-1

Reuse TF-M ITS filesystem

- Application-DataZone binding info [ app_id(client_id), key_id, zone_id, counter_id], ...

Provisioning info

As TF-M ITS filesystem is a concise filesystem, the size of its metadata region is small, it’s acceptable to create a filesystem context upon each data zone.
**External trusted secure storage implementation**

**ETSS partition APIs**

```c
etss_err_t tfm_etss_secure_flash_provisioning(size_t data_length, const void *p_data);
etss_err_t tfm_etss_set(psa_storage_uid_t uid, size_t data_length, const void *p_data,
                         psa_storage_create_flags_t create_flags*);
etss_err_t tfm_etss_get(psa_storage_uid_t uid, size_t data_offset, size_t data_size,
                         void *p_data, size_t *p_data_length);
etss_err_t tfm_etss_get_info(psa_storage_uid_t uid, struct psa_storage_info_t *p_info);
etss_err_t tfm_etss_remove(psa_storage_uid_t uid);
etss_err_t tfm_etss_mc_increment(uint8_t mc_id);
etss_err_t tfm_etss_mc_get(uint8_t mc_id, uint8_t *buf, uint32_t buf_size);
etss_err_t tfm_etss_generate_random_number(uint8_t *buf, uint32_t buf_size);
etss_err_t tfm_etss_get_puf(uint8_t *buf, uint32_t buf_size, uint32_t *puf_len);
```

*: Currently etss_set() service always provides confidentiality and replay protection regardless of PSA_STORAGE_FLAG_CONFIDENTIALITY and PSA_STORAGE_FLAG_REPLAY_PROTECTION, so only PSA_STORAGE_FLAG_WRITE_ONCE has practical effect.
Thank you for your Attention
Copyright© Macronix International Co., Ltd. 2022. All rights reserved, including the trademarks and tradename thereof, such as Macronix, MXIC, MXIC Logo, MX Logo, Integrated Solutions Provider, Nbit, Macronix NBit, HybridNVM, HybridFlash, HybridXFlash, XtraROM, KH Logo, BE-SONOS, KSMC, Kingtech, MXSMIO, RichBook, OctaBus, ArmorFlash, LybraFlash. The names and brands of third party referred thereto (if any) are for identification purposes only.