TF-M Library model vs. TF-M SFN model
Based on a quantitative approach
Purpose

- Compare TF-M Library model and SFN model implementation *based on some quantitative data*
  - Memory footprint
  - Performance
  - Development/management effort

- Implementation details/functionalities are not the focus
  - Refer to [FF-M 1.1 extension](#) Appendix C for detailed analysis on these existing frameworks
TF-M Library model

A TF-M specific lightweight framework implementation

- Secure Processing Environment (SPE) as a secure library
  - Based around a set of secure service functions
  - Those functions run as callbacks from Secure Partition Manager (SPM)
  - Each secure service function exports its corresponding veneer function

- Use cases
  - Isolation level 1
  - Highly resource-constrained devices
  - Single Armv8-M TrustZone scenario
Secure Function (SFN) Model

- A new framework defined in FF-M 1.1 extensions
  - A simpler programming model compared to IPC model
    - Reuse RoT service secure function call concept from TF-M Library model and integrate this into FF-M
      - RoT services are implemented as Secure Functions (SFN) that are called by the framework when the client makes a request to the service
    - Identical PSA Client APIs with IPC model
    - Reduce framework overhead for systems that do not require high levels of isolation
  
- Use cases
  - Isolation level 1
  - Highly resource-constrained devices

![Diagram of Secure Function (SFN) Model]
Library model vs. SFN model

Memory footprint

- Similar flash consumption
  - Code + RO data + RW data
- SFN model consumes less RAM than Library model does
  - RW data + ZI data

<table>
<thead>
<tr>
<th></th>
<th>Flash consumption (bytes)</th>
<th>RAM consumption (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library model</td>
<td>49456</td>
<td>20016</td>
</tr>
<tr>
<td>SFN model</td>
<td>50052 1.2%▲</td>
<td>15644 -21.8%▼</td>
</tr>
</tbody>
</table>

* Test config: Profile Small, Armclang 6.18, AN521, MinSizeRel build type
Library model vs. SFN model

- SFN model costs longer in client calls than Library model does
  - Additional operations required by FF-M
    - Client permission verification
    - RoT service version validation
    - Input parameter overlapping checks to avoid double-fetch inconsistency
    - Message construction and parse
    - RoT service invokes `psa_read()` to read input parameters

![NS client request approx. execution time (CPU cycles)](chart1)

- Secure client request approx. execution time (CPU cycles)

![Secure client request approx. execution time (CPU cycles)](chart2)

*TF-M Profiler tool
*Test config: Profile Small, GNU Arm, Musca-S1, Debug build type
Library model vs. SFN model

- SFN model costs longer in client calls than Library model does (cont’d)
  - TF-M specific implementation of SFN model
    - SFN models shares some common routines with IPC model to simplify implementation/maintenance
      - Such as dynamic handle/message instance allocation
      - Can be optimized further if required
    - Entry functions for TF-M stateless RoT services
      - TF-M Secure Partition implement an entry function to dispatch stateless RoT service callbacks
      - Reduce consumption of stateless handles to reserve indexes for 3rd-party RoT service usage
      - 3rd-party RoT services can export RoT service callbacks directly without an entry function

```c
psa_status_t its_service_sfn(const psa_msg_t *msg)
{
    switch (msg->type) {
        case TFM_ITS_SET:
            return tfm_its_set_req(msg);
        case TFM_ITS_GET:
            return tfm_its_get_req(msg);
        ...
    }
}
```
Library model vs. SFN model

Development/Maintenance effort

- How many conditional checks/branches are maintained for Library mode/SFN model?
  - Each one wraps Library/SFN model specific implementation in shared routines with IPC model
    - Changes of Library/SFN may impact IPC model, and vice versa
    - “Bidirectional” development/maintenance effort with IPC model
  - Library model: TFM_PSA_API/TFM_LIB_MODEL
    - Dedicated standalone SPM/HAL implementation
  - SFN model: CONFIG_TFM_PSA_API_SFN_CALL/CONFIG_TFM_SPM_BACKEND_SFN
    - Share common routines/implementation with IPC model

<table>
<thead>
<tr>
<th></th>
<th>Library model</th>
<th>SFN model</th>
</tr>
</thead>
<tbody>
<tr>
<td>C code</td>
<td>114</td>
<td>9</td>
</tr>
<tr>
<td>Linker scripts</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Build system</td>
<td>74</td>
<td>11</td>
</tr>
<tr>
<td>(including manifest tool)</td>
<td>215</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>215</td>
<td>20</td>
</tr>
</tbody>
</table>

```c
#ifndef TFM_PSA_API
$<$<BOOL:${TFM_PSA_API}>>:...>
#endif

#elif CONFIG_TFM_PSA_API_SFN_CALL == 1
$<$<BOOL:${CONFIG_TFM_SPM_BACKEND_SFN}>>:...>
#endif
```
Library model vs. SFN model

Observations

- Similar memory footprint
- SFN model is “slower” due to more execution steps compliant with FF-M
- Less development/maintenance effort for SFN model, with essential IPC model
Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
 شكراً
धन्यवाद
हादेस