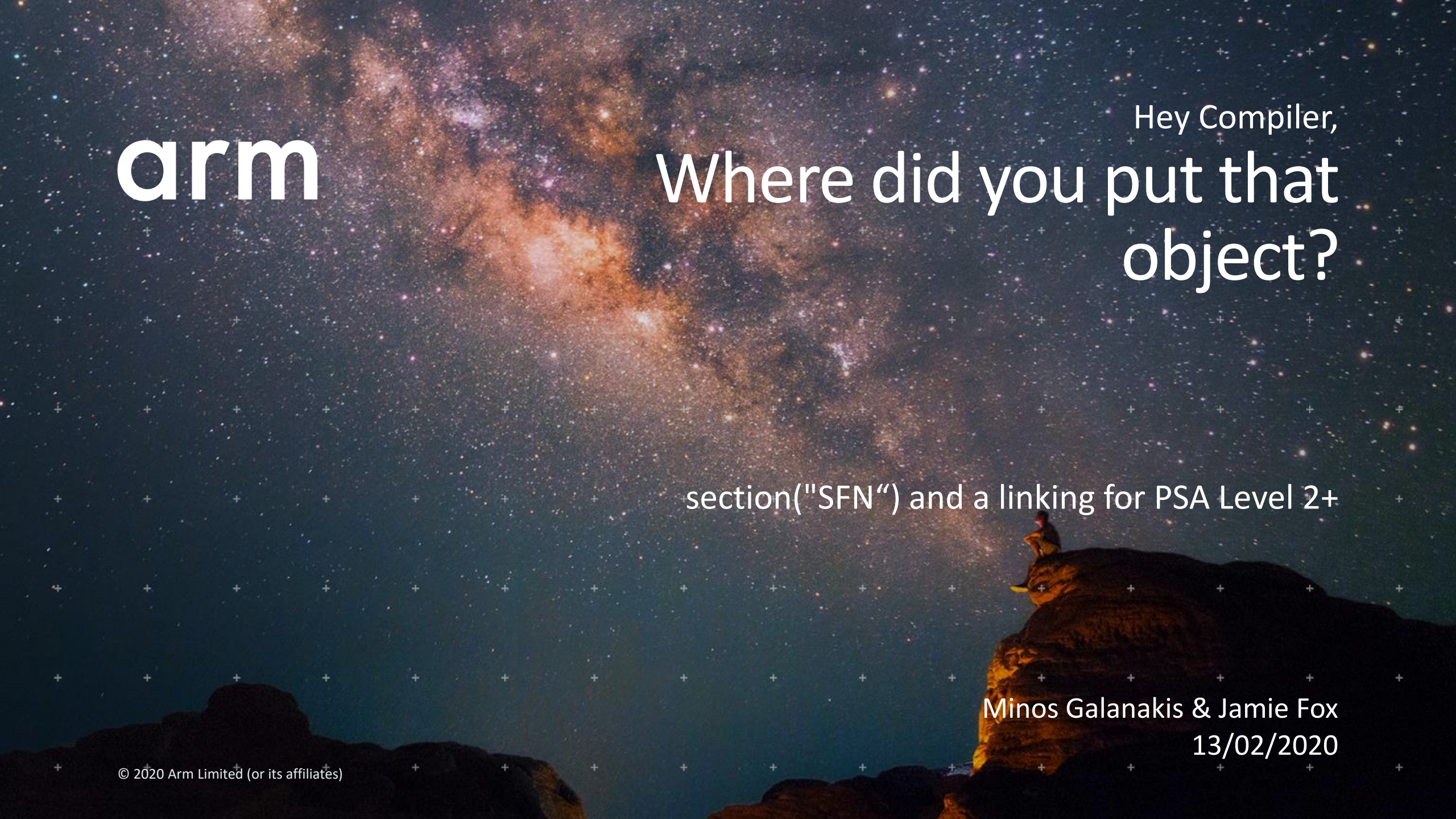


The “SFN” section

Tagging objects for Secure unpriviledged section

- Currently secure api’s are using the SFN attribute as a way of tagging methods, in order to be captured by the linker and placed into the TFM_UNPRIV_CODE section.
- The problem arises when using “const” method variables assuming that they will be placed in the .rodata section.
- The behavior of compilers is non-deterministic, even between different versions of the same toolchain, and can vary between different levels of optimization. The way a const is handled is a compile-time construct.

A quick example

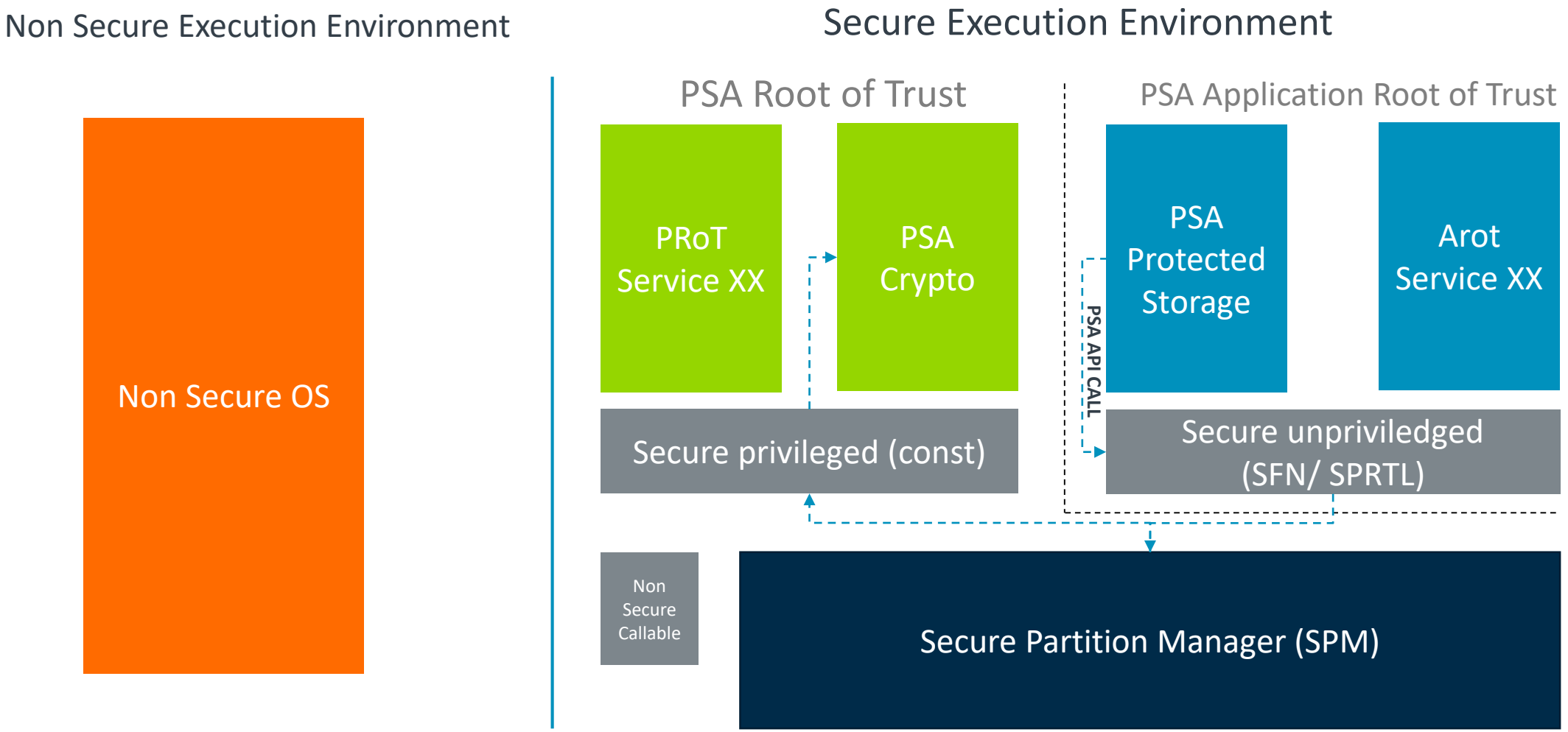
How could this affect an ARoT to PROT call

- Where will the “`tfm_crypto_pack_iovec iov`” be stored?
  1. ARMCLANG compiler 6.10 will place it in .rodata
  2. ARMCLANG compiler 6.12/6.13 will place it in stack
  3. GCC gcc version 7.2.1 will place it in stack
- There is a chance that method and the const data will not be placed not in the same region

tfm_crypto_secure_api.c

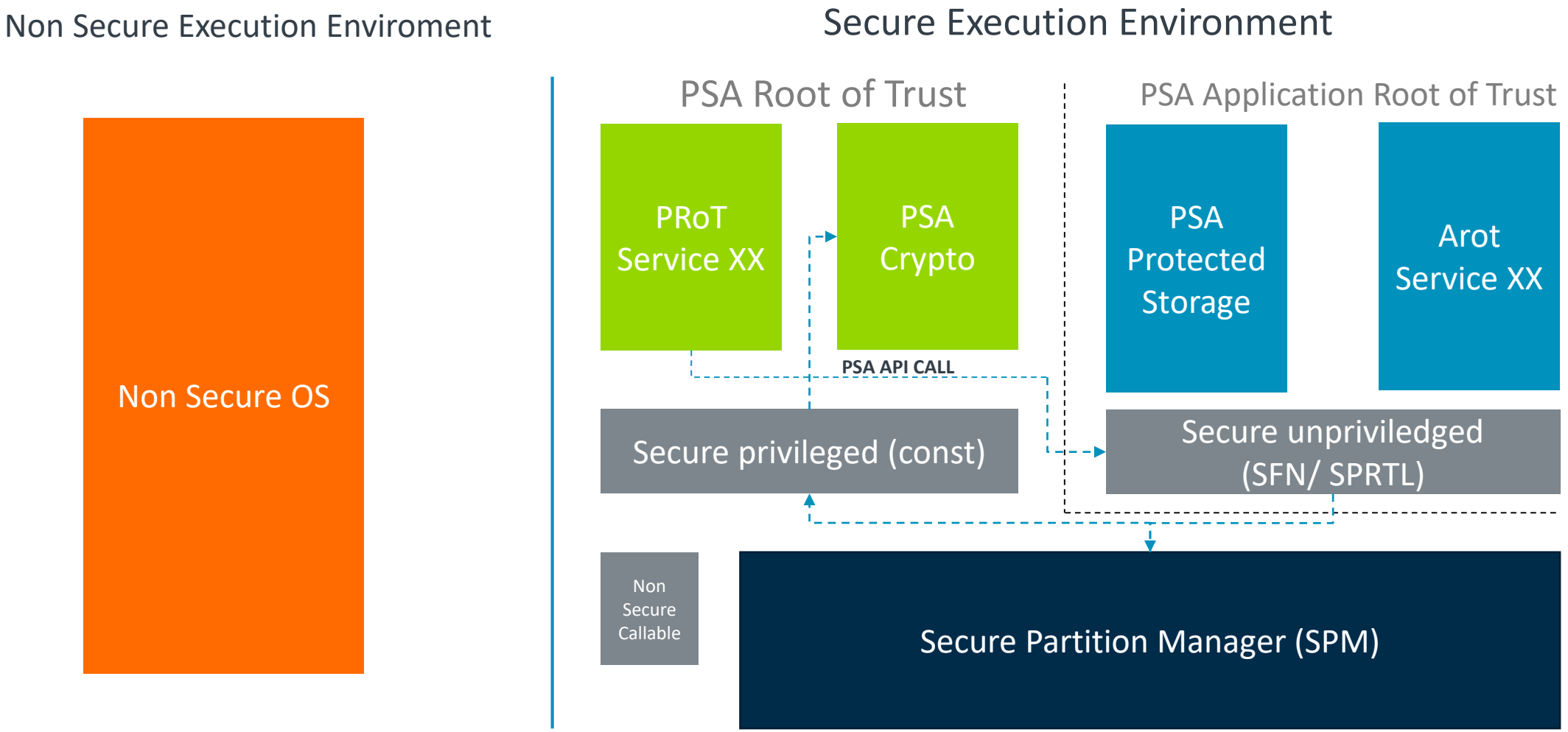
```
__attribute__((section("SFN")))
psa_status_t psa_allocate_key(psa_key_handle_t *handle)
{
#ifdef TFM_CRYPTO_KEY_MODULE_DISABLED
    return PSA_ERROR_NOT_SUPPORTED;
#else
    psa_status_t status;
    const struct tfm_crypto_pack_iovec iov = {
        .sfn_id = TFM_CRYPTO_ALLOCATE_KEY_SID,
    };
    psa_invec in_vec[] = {
        {.base = &iov, .len = sizeof(struct tfm_crypto_pack_iovec)},
    };
    psa_outvec out_vec[] = {
        {.base = handle, .len = sizeof(psa_key_handle_t)},
    };
```

PSA LV2 ARoT to PRoT call example



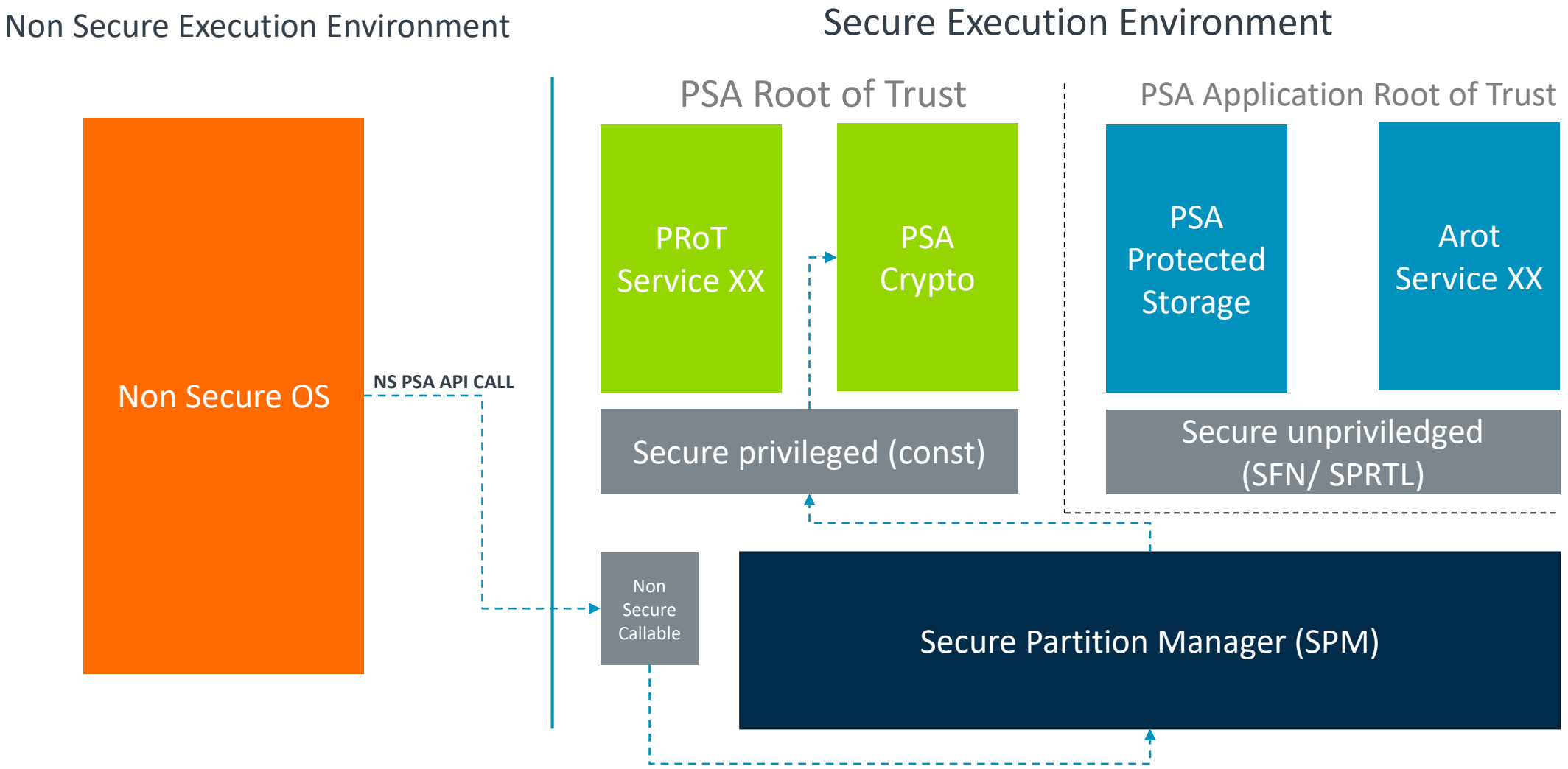
arm

PSA LV2 PRoT to PRoT call example



arm

PSA LV2 Non Secure to Secure PRoT call example



arm

Why would that be a problem?

- With PSA Level 1, all partitions share access privileges for all the segments in the secure side.
- With PSA Level 2 and 3, access to the Secure Privileged .rodata is not guaranteed. In the previous example, when Protected Storage calls PSA Crypto, the api call `psa_allocate_key` will reside “SNF” section marked as TFM_UNPRIV_CODE while the const iov parameter’s location is determined by the toolchain. It can be stored in the stack, which will work, or in the .rodata of the PSA CRYPTO partition marked as Privileged code, which will cause it to fail.
- The processor is executing secure non-priviledged code without dropping its privileged state. It is replying on the MPU for guarantying the +ro state of the code.

What should we be aiming for?

- With an open source project aimed at supporting multiple tool-chains, the objects should be placed in a deterministic way, so partitions and memory protection can be planned and enforced correctly.
- It may be wise to group the common purpose methods into an object/library and configure its properties and section during linking.
- If that approach is taken, what guarantees are there the API's or the compiler will not generate intermediate static objects, compromising security?
- Taking into consideration platform variations, there can be more than one configurable ways of addressing the problem.

Minimizing impact

- This is a low risk item and does not currently compromise security.
- The “SFN” section will be removed, and the API calls will be bundled onto the Secure Partition Runtime Library (design review at patch #3457)
- The design requirements for STRTL should extend to any code shared between secure privileged and unprivileged(+ro, thread safe, added overhead for validating isolation)
- Const and static variables should be banned on all API calls.
- Contributions to API calls should be thoroughly inspected for gadgets.
- Further Suggestions?

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה