TF-M Tests Improvements

Kevin Peng

July 2020

What Are the Improvements

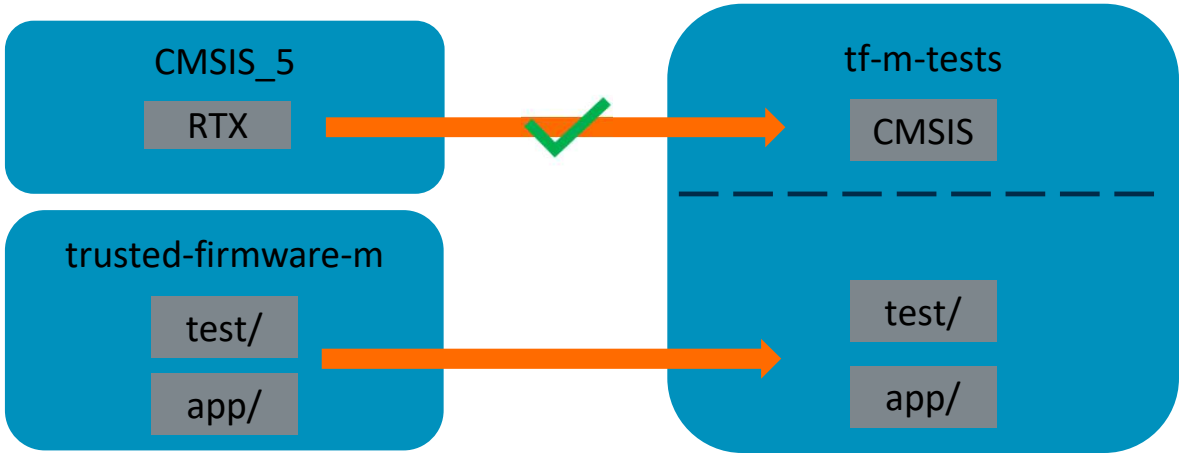
Write in your subtitle here

- The *tf-m-tests* repo
- Test Framework

The tf-m-tests repo

- A new *tf-m-tests* repo has been created under TF-M
- Intended to move in all the TF-M test codes

▼ *TF-M*
tf-m-tests.git
tf-m-tools.git
trusted-firmware-m.git



arm

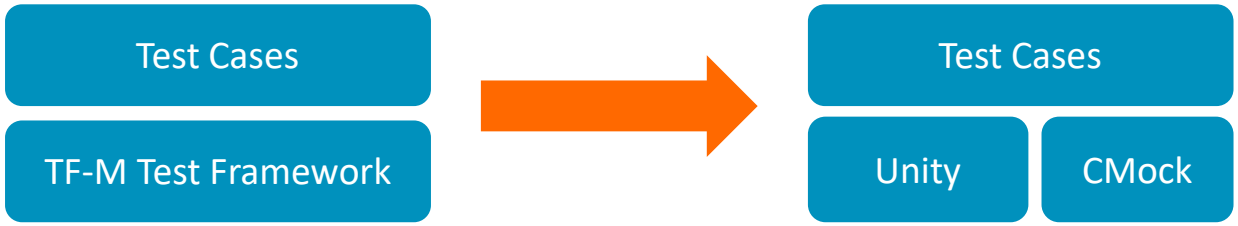
Migration of the test codes

- Phase 1
  - Move the codes as is
  - Only necessary changes to pass compilation
- Phase 2
  - Refine the build system
  - Refine the file structure

Test Framework

Introducing Unity + CMock

- Unity: Opensource test framework for C
- CMock: framework of automated mock and stub generation for C
- https://github.com/ThrowTheSwitch



Note: Mocks are optional for test cases

arm

Test Framework

Why do we need a new one

- The current test framework
  - Only has the limited automation for developing
  - Only a few test assertions
  - Does not support mock or stub
  - **Not so friendly for test developers**

Test Framework

Changes after using Unity

- Simpler code for test case and more readability

```c
static void tfm_attest_test_2001(struct test_result_t *ret)
{
    int32_t err;

    err = minimal_test();
    if (err != 0) {
        TEST_LOG("minimal_test() returned: %d\r\n", err);
        TEST_FAIL("Attest token minimal_test() has failed");
        return;
    }

    ret->val = TEST_PASSED;
}
```

```c
void test_tfm_attest_2001()
{
    TEST_ASSERT(minimal_test() == 0, "Attest token minimal_test() has failed");
    TEST_PASS();
}
```


arm

Test Framework

Changes after using Unity

- Easy development

Developer focus

```c
void test_tfm_attest_2001()
{
    TEST_ASSERT(minimal_test() == 0, "Attest token minimal_test() has failed");
    TEST_PASS();
}
```

Tools generated codes

```c
/*=======MAIN=====*/
//int tfm_attest_test_main(void);
int tfm_attest_test_main(void)
{
  UnityBegin("attestation_ns_interface_testsuite.c");
  run_test(test_tfm_attest_2001, "test_tfm_attest_2001", 28);
  run_test(test_tfm_attest_2002, "test_tfm_attest_2002", 46);
  run_test(test_tfm_attest_2003, "test_tfm_attest_2003", 67);
  run_test(test_tfm_attest_2004, "test_tfm_attest_2004", 91);
  run_test(test_tfm_attest_2005, "test_tfm_attest_2005", 113);
  return UnityEnd();
}
```

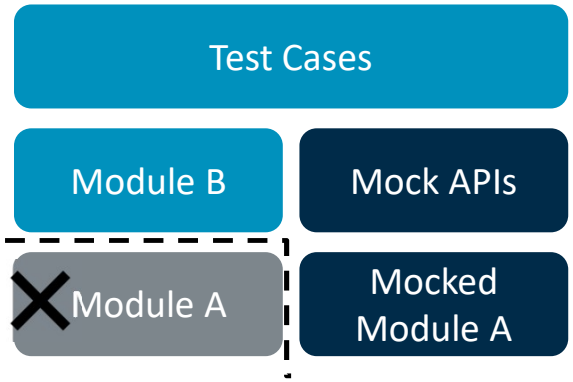
Don’t have to write the following codes

```c
static struct test_t attestation_interface_tests[] = {
#ifdef INCLUDE_TEST_CODE /* Remove them from release build */
    {&tfm_attest_test_2001, "TFM_ATTEST_TEST_2001",
     "Minimal token test of attest token", {TEST_PASSED} },
    {&tfm_attest_test_2002, "TFM_ATTEST_TEST_2002",
     "Minimal token size test of attest token", {TEST_PASSED} },
    {&tfm_attest_test_2003, "TFM_ATTEST_TEST_2003",
     "Short circuit signature test of attest token", {TEST_PASSED} },
#endif
    {&tfm_attest_test_2004, "TFM_ATTEST_TEST_2004",
     "ECDSA signature test of attest token", {TEST_PASSED} },
    {&tfm_attest_test_2005, "TFM_ATTEST_TEST_2005",
     "Negative test cases for initial attestation service", {TEST_PASSED} },
};

void
register_testsuite_ns_attestation_interface(struct test_suite_t *p_test_suite)
{
    uint32_t list_size;

    list_size = (sizeof(attestation_interface_tests) /
                 sizeof(attestation_interface_tests[0]));

    set_testsuite("Initial Attestation Service non-secure interface tests"
                  "(TFM_ATTEST_TEST_2XXX)",
                  attestation_interface_tests, list_size, p_test_suite);
}
```

Test Framework

The CMock framework

- Mock framework lets you control the behaviors of the modules that your main test object interacts with



Module A interface:
int moduleAFunc(int a, int b)

Mock APIs:

```
void moduleAFunc_ExpectAndReturn(int a, int b, int toReturn);
void moduleAFunc_ExpectAndThrow(int a, int b, EXCEPTION_T error);
void moduleAFunc_IgnoreAndReturn(int toReturn);
```

```
void test_case_1(void)
{
   int a =1, b = 2, c;
   moduleAFunc_ExpectAndReturn(1, 2, 3);
   c = moduleAFunc(a, b); // c is 3
   TEST_ASSERT(c = 3);
}
```


arm

Test Framework

Why Unity

Pros

- It’s pure C
- Automation scripts
- Mock feature
- Easy integration – only 3 source files for each(Unity & CMock)

Cons

- Extra build env ruby – the automation tools are written in ruby

Comparison to other (a few popular ones) frameworks

- Google Test – Aims for C++
- CppUTest – Written in C++, and test cases in C++
- Check - only supports a handful of assertions
- Cmocka - no scripts and requires the standard C library

Unity users:

- a:Fr
- mbed-OS

Unity + CMock

How are they managed

- MIT License – permissive license
- Import the source code as local copy
  - Less than 10 files, include source codes and scripts
  - Easy for customization
  - Won’t upgrade frequently
  - Won’t upstream
- Security – no considerations as test purpose only

arm

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