TRNG Firmware Interface[1] in TF-A

Jimmy Brisson

July 30, 2020

[1] PDF Spec Link

Outline

TRNG Firmware Interface ABI

Name Return	ID X0	X1 X1	X2 X2	X3 X3
Version	84000050	-	-	-
	Major Minor	-	-	-
Features	84000051	ID	-	-
	Success Flag	-	-	-
UUID	84000052	-	-	-
	uuid[31:0]	uuid[63:32]	uuid[95:64]	uuid[127:96]
RND 32	84000053	rnd size	-	-
	Success Flag	rnd[95:64]	rnd[63:32]	rnd[31:0]
RND 64	C4000053	rnd size	-	-
	Success Flag	rnd[191:128]	rnd[127:64]	rnd[63:0]

Porting

```
/* TRNG platform functions */
#if TRNG_SUPPORT
extern uuid_t plat_trng_uuid;
void plat_trng_init(void);
bool plat_get_entropy(uint64_t *out);
#endif
```

Implementation Notes

- Entropy ring buffer; Indexed in bits
- TF-A-Tests enforce spec conformance
- TF-A-Tests can't tests for randomness

Appendix: Dispatch

```
switch (smc_fid) {
case ARM_TRNG_VERSION:
        SMC_RET1(handle, MAKE_SMCCC_VERSION(TRNG_VERSION_MAJOR,
                        TRNG_VERSION_MINOR));
        break; /* unreachable */
case ARM_TRNG_FEATURES:
        if (is_trng_fid((uint32_t)x1)) {
          SMC_RET1(handle, TRNG_E_SUCCESS);
        } else {
          SMC_RET1(handle, TRNG_E_NOT_SUPPORTED);
        }
        break; /* unreachable */
case ARM_TRNG_GET_UUID:
        SMC_UUID_RET(handle, plat_trng_uuid);
        break; /* unreachable */
case ARM_TRNG_RND32:
        if (x1 == 0 || x1 > 96) {
          SMC_RET1(handle, TRNG_E_INVALID_PARAMS);
        }
        return trng_rnd32((uint32_t)x1, handle);
case ARM_TRNG_RND64:
        if (x1 == 0 || x1 > 192) {
          SMC_RET1(handle, TRNG_E_INVALID_PARAMS);
        }
        return trng_rnd64((uint32_t)x1, handle);
default:
        WARN("Unimplemented TRNG Service Call: 0x%x\n", smc_fid);
        SMC_RET1(handle, TRNG_E_NOT_IMPLEMENTED);
        break; /* unreachable */
}
```

Appendix: Entropy Buffer

```
/* Entropy pool */
/* For the proof below, note that the TRNG Firmware interface can request up to
 * 192 bits of entropy in a single call or 3, 64bit words per call. */
#define WORDS_IN_POOL (4)
uint64_t entropy[WORDS_IN_POOL];
uint32_t entropy_bit_index = 0;
uint32_t entropy_bit_size = 0;

#define BITS_PER_WORD (sizeof(entropy[0]) * 8)
#define BITS_IN_POOL (WORDS_IN_POOL * BITS_PER_WORD)


/* Note: This function assumes that the caller has taken the lock for the
 * entropy pool
 */
static bool trng_fill_entropy(uint32_t nbits)
{
        while (nbits > entropy_bit_size) {
                bool new_entropy_valid =
                        plat_get_entropy(&entropy[ENTROPY_FREE_INDEX]);
                if (new_entropy.valid) {
                        entropy_bit_size += BITS_PER_WORD;
                        assert(entropy_bit_size <= BITS_IN_POOL);
                } else {
                        return false;
                }
        }
        return true;
}
```