Arm® True Random Number Generator Firmware Interface 1.0

Platform Design Document

Non-confidential

Document number: ARM DEN 0098

Contents

Release information

Date	Version	Changes
2022/Jan/14	1.0 REL0	• Add TRNG FW implementation suggestions. • Clarify high level system design.
2020/Sep/01	1.0 EAC0	• Initial release of the specification.

Arm Non-Confidential Document Licence (“Licence”)

This Licence is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“Licensee”) is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

(i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;

(ii) manufacture and have manufactured products which have been created under the licence granted in (i) above; and

(iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

LES-PRE-21585 version 4.0

About this document

Terms and abbreviations

Term	Meaning
Back end implementation	The entity generating entropy and providing the information exposed through the Front end implementation.
Client	The SW entity that requestes conditioned entropy via the Front end implementation.
Communication channel	The information path between the Back end and the Front end.
EL0	The lowest Exception level. The Exception level that is used to execute user applications in Non-secure state.
EL1	The privileged Exception level. The Exception level that is used to execute operating systems in Non-secure state.
EL2	The hypervisor Exception level. The Exception level that is used to execute hypervisor code in Non-secure state.
EL3	The Secure monitor Exception level. The Exception level that is used to execute Secure monitor code, which handles the transitions between Non-secure and Secure states. EL3 is always in Secure state.
Front end implementation	The TRNG FW interface ABI implementation (as defined in the section 2).
FW	Firmware
OS	Operating System
S-EL0	The Secure EL0 Exception level. The Exception level that is used to execute Trusted application code in Secure state.
S-EL1	The Secure EL1 Exception level. The Exception level that is used to execute Trusted OS code in Secure state.
S-EL2	The Secure EL2 Exception level. The Exception level that is used to execute hypervisor code in Secure state.
SoC	System on Chip
TRNG	True Random Number Generator - A device that produces a random stream of bits.
UUID	Unique Universal IDentifier

References

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Arm® TrustZone True Random Number Generator*. (ARM 100976_0000_00_en r0p0) Arm Ltd.

[2] *Arm® Architecture Reference Manual for Armv8-A architecture profile*. (ARM DDI 0487 E.a) Arm Ltd.

[3] *SMC CALLING CONVENTION System Software on Arm® Platforms*. (ARM DEN 0028 C) Arm Ltd.

[4] *Recommendation for the Entropy Sources Used for Random Bit Generation*. (10.6028/NIST.SP.800-90B B) NIST.

Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to errata@arm.com. Give:

- The title (True Random Number Generator Firmware Interface).
- The document ID and version (ARM DEN 0098 1.0).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Introduction

This document defines an interface between an Operating System (OS) and a TRNG FW implementation that provides a conditioned entropy source.

The conditioned entropy is commonly used to seed deterministic random number generators or to generate keys, among other use-cases.

This document considers that the TRNG FW implementation is composed of the following elements:

1. Back end implementation: the entity generating the conditioned entropy, with the following constituents:
   - the TRNG HW, like an Arm TrustZone TRNG [1], or some other raw noise source, and
   - the device driver which handles the TRNG HW, maintains entropy pools, and conditions the entropy if the TRNG HW only provides raw noise.
2. Communication channel: The IMPLEMENTATION DEFINED information exchange medium between the Back end and the Front end implementations.
3. Front end implementation: The implementation of the ABI defined in section 2.

A plausible system is shown in Figure 1. There, the Back end implementation is made of a TRNG HW and a driver implemented at EL3. The Front end implementation spans EL3 and EL2. The Back end and Front end implementations exchange information at EL3.

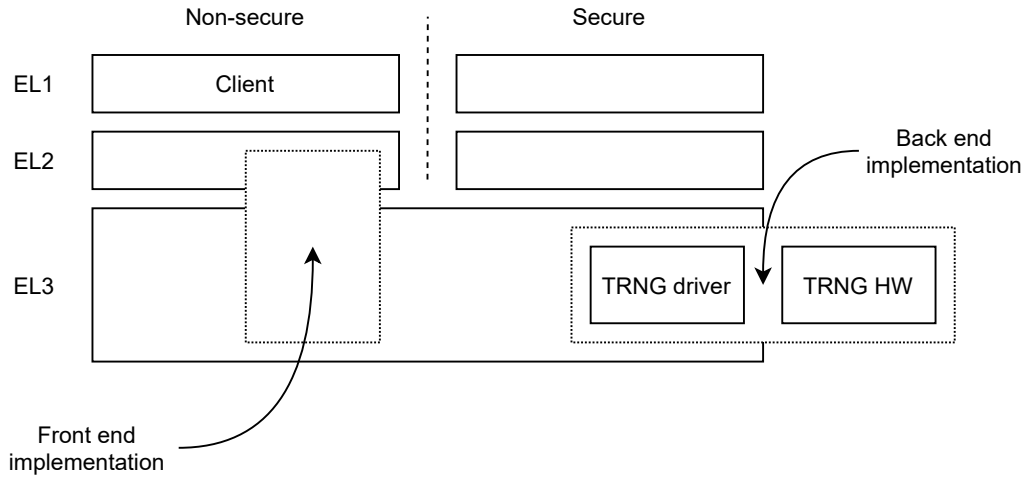

Figure 1: Example system diagram

The main intent of this document is to define the properties of the Front end implementation, see sections 1.2.1 and 2. The properties of the Back end implementation and Communication channel are stated in section 1.1.2.

The ABI defined in this document requires a FW implementation compliant with the SMCCC version 1.1 or later.

1.1 Back end implementation considerations

Different platforms can opt to implement a subset of the exception levels defined in the Armv8-A architecture [2]. The Exception level and Security state where the TRNG Back end can be implemented depend on the available Exception levels as described in Table 3.

Table 3: Back end implementation possible Exception levels

EL0	EL1	EL2	EL3	Availability
Y	Y	N	N	Not implemented
Y	Y	Y	N	Implemented in Non-secure EL2
Y	Y	Y/N	Y	Implemented in Secure EL0/EL1/EL2

This document describes a TRNG interface for invoking TRNG services from Normal world. This does not preclude invocations from Secure world. They are beyond the scope of this specification.

A Back end implementation must register with the Front end implementation (EL3/EL2 FW) through an IMPLEMENTATION DEFINED mechanism. The communication between the Front end and the Back end implementation is IMPLEMENTATION DEFINED. There exists at most one Back end implementation registered with the Front end implementation.

1.1.1 Back end UUID

The TRNG Back end has an associated UUID. This UUID is allowed to vary between different TRNG Back end FW versions **OR** different TRNG HW versions. Any specific (Back end FW version, TRNG HW version) pair must have the same UUID.

The TRNG Back end implementer must ensure the following:

- The UUID is defined for a (TRNG Back end FW version, TRNG HW version) pair.
- The UUID must be such that UUID[31:0] $\neq$ 0xFFFF_FFFF in accordance with the SMCCC defined UUID format [3].
- The (TRNG Back end FW version, TRNG HW version) pair UUID is clearly defined in the TRNG Back end implementer product documentation.

1.1.2 Back end implementation requirements

The conditioned entropy that is provided by a TRNG Back end can be used in security sensitive use-cases. The security properties of these use-cases root themselves on the guarantees provided by the TRNG Back end.

A Back end implementation must be analyzed. The outcome of this analysis should guarantee that:

- Each conditioned entropy bit is discarded after being provided to a client.
- If multiple entropy sources exist in the system, they must be uncorrelated.
- The entropy source complies with the guidelines that are detailed in [4] (NIST 800-90B).
  - The Back end implementation must implement a health test on the raw noise.
  - The Back end implementation must implement a vetted conditioning component.
- The Communication channel between the Front end and the Back end implementations must not be tampered with:
  - Any software component that is not trusted by any caller must be unable to eavesdrop or alter the information exchanged in the Communication channel.
  - Any state associated to the entropy held by the Back end must not be physically accessible or subject to being probed from outside the SoC. Exemption: a Back end implemented at EL2 may have limited ability to shield entropy associated state from external probing. A TRNG Back end implemented at EL2 is deemed compliant even if this property is not met.

1.2 TRNG FW implementation considerations

The TRNG FW implementation may span several Exception Levels. The entropy is propagated, across the different Exception Levels in the entropy delivery path, between the Back end implementation and the entropy Client. As a result of this propagation, it is possible that the entropy bits are stored in various memory locations controlled by the multiple Exception Levels during hand-offs, or due to context switches. If this entropy is left in memory, after delivery to the Client or to the next EL in the entropy delivery path, there is an increased risk of the entropy leaking to entities other than the Client, via side-channels.

1.2.1 Implementation guidance

The TRNG FW implementation should:

- Discard all entropy bits from all memory locations once the entropy has been delivered to the requester or to the next EL in the entropy delivery path. The entropy bits can be discarded by, for example, overwriting them with zero.

1.3 Calls defined per ABI version

Table 4 relates the ABI version to the defined calls and their requirement status.

Table 4: ABI required functions per version

Call name	Mandatory from	Optional from
TRNG_VERSION	v1.0	–
TRNG_FEATURES	v1.0	–
TRNG_GET_UUID	v1.0	–
TRNG_RND	v1.0	–

1.4 Discovery

1.4.1 ABI discovery

The SMCCC mandates the SMCCC implementation to return NOT_SUPPORTED if the called function is not implemented [3].

The presence of the TRNG ABI must be discovered by calling TRNG_VERSION. A TRNG ABI implementation is present if and only if a call to TRNG_VERSION returns a non-negative value in W0.

The TRNG_FEATURES function must be present in any TRNG ABI implementation. The TRNG_FEATURES function is implemented if a call to TRNG_VERSION returns a non-negative value in W0.

The presence of the remaining functions in the TRNG ABI is determined through calls to TRNG_FEATURES passing the FID of the call as the argument in W1 (trng_func_id). See section 2.2 for information on TRNG_FEATURES.

Mandatory functions are guaranteed to be present if the TRNG ABI version is greater or equal than the version of the ABI the particular function was mandated on. See section 1.3 for information on ABI versions and mandatory functions.

1.4.2 TRNG Back end discovery

The presence of the TRNG ABI implies the presence of the TRNG Back end.

A TRNG ABI is only reported to be present after the TRNG Back end ABI has registered with the Front end implementation.

The TRNG Back end must ensure it has registered with the Front end implementation prior to any potential TRNG ABI Client (at EL1 or EL2) issuing a single instruction in any PE in the system.

1.5 Invocation considerations

The interface, described in section 2 of this document, complies with the SMCCCv1.1 calling convention [3].

Table 5 lists the recommend conduits as a function of the implemented exception levels:

Table 5: Recommended conduit

EL3	EL2	Conduit
Y	Y	SMC
Y	N	SMC
N	Y	HVC
N	N	–

2 Interface

2.1 TRNG_VERSION

The function returns the implemented TRNG ABI version. The version is composed of two fields termed major and minor revision of 15 and 16 bis respectively.

2.1.1 Function definition

Function ID (W0)	0x8400_0050		
Returns			
	Success (W0 $\geqslant$ 0)	W0[31]	MBZ
		W0[30:16]	Major revision
		W0[15:0]	Minor revision
		W1 – W3	Reserved (MBZ)
	Error (W0 $<$ 0)		
		NOT_SUPPORTED	Function not implemented

Table 6: TRNG_VERSION function definition

2.1.2 Usage

The function returns the major and minor revision as an aggregate 31-bit integer in R0/W0. The W0[30:16] bits contain the major revision, the least significant bits (W0[15:0]) contain the minor revision. A minor revision increment cannot break backwards compatibility with older ABI versions. A major revision can introduce changes which break compatibility with past ABI versions. The return value can be used by the caller as a discovery mechanism for the ABI functions that are listed as mandatory in Table 4.

2.1.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that SMCCC_VERSION reports a SMCCC version greater or equal than 1.1 [3] before calling TRNG_VERSION.

2.1.4 Implementation responsibilities

The Implementation has the following responsibilities:

- The implementation must return NOT_SUPPORTED if a Back end is not present.
- The Implementation must return the greatest version value that respects the constraints listed in Table 4.

2.2 TRNG_FEATURES

2.2.1 Function definition

Function ID (W0)	0x8400_0051		
Parameters			
	W1 trng_func_id		
Returns			
	Success (W0 $\geqslant$ 0)		
		SUCCESS	Function is implemented.
		> 0	Function is implemented and has specific capabilities, see function definition.
	Error (W0 $<$ 0)		
		NOT_SUPPORTED	Function with FID=trng_func_id is not implemented

Table 7: TRNG_FEATURES function definition

2.2.2 Usage

The caller can determine if functions defined in the TRNG ABI are present in the ABI implementation. The caller can determine function specific features (signaled by a positive return status in W0). The function specific features must be described in the function definition.

2.2.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure the TRNG ABI is present before calling this function.

2.2.4 Implementations responsibilities

The function implementation has the following responsibilities:

- The implementation must return NOT_SUPPORTED if trng_func_id is a value not defined in the TRNG ABI.

2.3 TRNG_GET_UUID

The function returns the UUID of the TRNG Back end. See Section 1.1.1 for more information on the Back end UUID.

2.3.1 Function definition

Function ID (W0)	0x8400_0052		
Returns			
	Success (W0 $\neq$ -1)		
		W0	UUID[31:0]
		W1	UUID[63:32]
		W2	UUID[95:64]
		W3	UUID[127:96]
	Error (W0 = -1)		
		W0	NOT_SUPPORTED

Table 8: TRNG_GET_UUID function definition

2.3.2 Usage

The caller obtains the UUID that is reported by the Back end. The returned UUID can, for instance, be used by a caller to identify a Back end that has vulnerabilities. The caller must not rely on the returned UUID as a trustworthy TRNG Back end identity.

2.3.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure this function is implemented before calling it. This function is discoverable by calling TRNG_FEATURES with trng_func_id set to 0x8400_0052.
- The caller must not rely on the returned UUID as a trustworthy TRNG Back end identity.

2.3.4 Implementation responsibilities

The function implementation has the following responsibilities:

- The implementation must ensure that a Back end UUID is such that UUID[31:0] $\neq$ 0xFFFF_FFFF. Otherwise the function return would be indistinguishable from NOT_SUPPORTED.

2.4 TRNG_RND

The function returns N bits of conditioned entropy if successful, otherwise returns immediately with an error code. The call is non-blocking.

2.4.1 Function definition

Function ID (W0)	0x8400_0053		
	0xC400_0053		
SMC32 Parameters			
	W1 N bits of entropy ($1 \leqslant N \leqslant 96$)		
SMC64 Parameters			
	X1 N bits of entropy ($1 \leqslant N \leqslant 192$)		
SMC32 Returns			
	Success (W0 = 0):		
		W0	MBZ
		W1	Entropy[95:64]
		W2	Entropy[63:32]
		W3	Entropy[31:0]
	Error (W0 < 0)		
		W0	NOT_SUPPORTED
			NO_ENTROPY
			INVALID_PARAMETERS
		W1 – W3	Reserved (MBZ)
SMC64 Returns			
	Success (X0 = 0):		
		X0	MBZ
		X1	Entropy[191:128]
		X2	Entropy[127:64]
		X3	Entropy[63:0]
	Error (X0 < 0)		
		X0	NOT_SUPPORTED
			NO_ENTROPY
			INVALID_PARAMETERS
		X1 – X3	Reserved (MBZ)

Table 9: TRNG_RND function definition

2.4.2 Usage

The caller requests N bits of conditioned entropy, where $1 \leqslant N \leqslant$ MAX_BITS with MAX_BITS = 96 for SMC32 calls and MAX_BITS = 192 for SMC64 calls.

A MAX_BITS-bits wide value (Entropy) is returned across the W1/X1 to W3/X3 registers. The requested conditioned entropy is returned in Entropy[N-1:0]. The bits in Entropy[MAX_BITS-1:N] are 0. If N=MAX_BITS, then Entropy[MAX_BITS-1:N] is a 0-bit length field.

The call is non-blocking, when the available entropy is less than N, the call must returns NO_ENTROPY.

There are system designs where FW may limit the rate at which entropy is requested by a caller. To enable this entropy request rate limitation, the call is allowed to return NO_ENTROPY even if sufficient conditioned entropy exists to service the request.

2.4.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that the function is implemented before calling it. This function is discoverable by calling TRNG_FEATURES, with trng_func_id set to 0x8400_0053 for the SMC32 call and trng_func_id set to 0xC400_0053 for the SMC64 call.
- The caller cannot request more than MAX_BITS bits of conditioned entropy per call.
- The caller must ensure that only the value in Entropy[N-1:0] is consumed and that the remaining bits in Entropy[MAX_BITS-1:N] are ignored.

2.4.4 Implementation responsibilities

The function implementation has the following responsibilities:

- The implementation must ensure that any bit of conditioned entropy given to a client is immediately discarded.
- The implementation must return immediately, with status NO_ENTROPY, if the available conditioned entropy is determined to be less than N.
- The implementation must return INVALID_PARAMETERS if $N >$ MAX_BITS.
- The implementation can return NO_ENTROPY even if sufficient conditioned entropy exists to service the request.

2.5 Return codes

The status return codes that are defined for TRNG ABI calls are listed in Table 10.

Table 10: Return status codes

Name	Value
SUCCESS	0
NOT_SUPPORTED	-1
INVALID_PARAMETERS	-2
NO_ENTROPY	-3