SP 800-90B Non-Proprietary Public Use Document

MPIC CPU Jitter LRNG Entropy Source v52



Document Version 1.1

April 2026

Document prepared by:



www.lightshipsec.com

Contents

1 Description 3
2 Security Boundary 3
3 Operating Conditions 4
4 Configuration Settings 4
5 Physical Security Mechanisms 4
6 Interfaces 4
7 Min-entropy rate 4
8 Health Tests 5
9 Maintenance 5
10 User Verification 5
10.1 Collecting samples 5
10.2 Raw Data Analysis 5
11 Vendor Permissions and Relationship 5

1 Description

1 This is the SP 800-90B non-proprietary Public Use Document for the MPIC CPU Jitter LRNG Entropy Source v52. The document describes version 3.4.0 of the Entropy Source.

2 The Entropy Source is based on the jitterentropy-library v3.4.0, which is a non-physical entropy source that is part of the kernel binary. The Entropy Source design includes elements that map to the conceptual components contained within an SP 800-90B entropy source:

- The primary noise source (memory and conditioning timing)
- Health tests
- A vetted conditioning algorithm (SHA3-256)

3 It makes no IID claim and thus meets all the requirements for non-IID compliance. The tested platforms are described in Table 1.

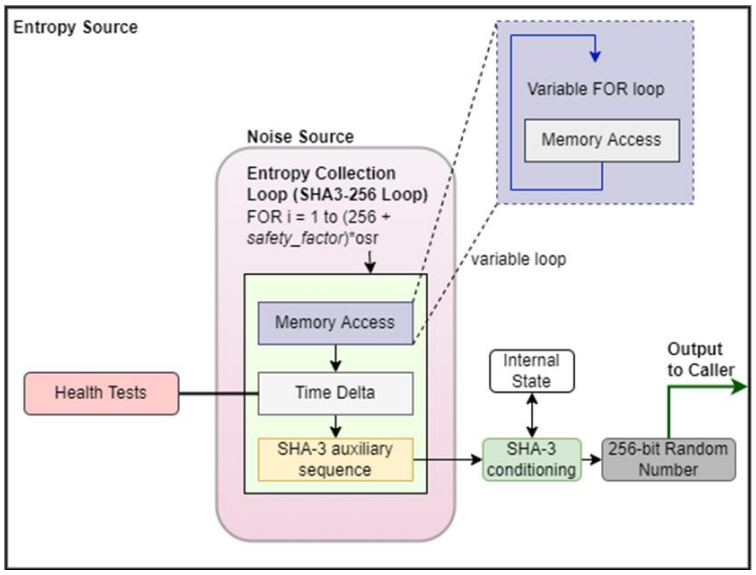
4 The Entropy Source is compliant to NIST Special Publication 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation (Last Modified Date: January 2018), IG D.J (Last Modified Date: November 5, 2021) and IG D.K (Last Modified Date: March 17, 2023).

2 Security Boundary

5 The entropy source, identified as the Jitter Entropy (JEnt) library version 3.4.0, operates within a defined security boundary. The executable binary file serves as the boundary for this non-physical, software-based entropy source. It is compiled directly from the C source code that implements it, which in this case is the Linux kernel source.

6 JEnt is based on the jitterentropy library. Its design maps to the conceptual components described in NIST SP 800-90B and includes:

- A primary noise source (based on memory access and conditioning timing),
- Health tests to ensure the proper functioning of the entropy source,
- A conditioning function implemented using SHA3-256.



3 Operating Conditions

7 The Entropy Source was tested in the following operational environments:

Table 1. Operational Environments

Label	CPU	Clock Speed	Cache	OS
OE1	NXP i.MX8MP (Arm Cortex-A53)	1.6 GHz	32KB	MPIC2LinuxDistribution XR 4.3

4 Configuration Settings

8 The only required configuration parameter for this kernel implementation is the kernel configuration system (Kconfig) parameter CONFIG_CRYPTO_JITTERENTROPY_OSR, which dictates the operational OSR at compile time.

Table 2. CPU Jitter Configuration Parameters

Parameter	Value	Type
CONFIG_CRYPTO_JITTERENTROPY_OSR	8	Kconfig

5 Physical Security Mechanisms

9 N/A – The Entropy Source is non-physical.

6 Interfaces

10 The Entropy Source provides the following interfaces to the calling application:

Table 3. Interfaces

Interface	API Command	Description
GetEntropy	jent_kcapi_random()	Obtains conditioned entropy for the caller. This is the main function of the entropy source, the one that shall be used to request entropy data. It returns 0 if the entropy was generated successfully.
GetNoise	jent_measure_jitter()	Obtains raw noise data for testing purposes. This function returns the result of the stuck test.
HealthTest	jent_kcapi_init()	Initializes the Entropy Source and performs statistical tests to verify that the underlying system offers the properties needed for measuring and collecting entropy.

11 The interfaces in the above table refer to those specified in section 2.3 of SP 800-90B.

7 Min-entropy rate

12 The Entropy Source generates outputs that are considered to have full entropy. A request for 256 bits of entropy results in 320 bits of entropy being added to the entropy pool. This ensures that outputs always possess full entropy.

8 Health Tests

13 The Entropy Source implements the following health tests:

- Stuck Test
- Repetition Count Test
- Adaptive Proportion Test
- Lag Predictor Test

14 The Entropy Source runs these tests when it starts and continuously while it is operating. All tests check for persistent failures based on their respective "cutoff" values, which represent expected error thresholds. All these tests are performed on raw data before any conditioning is done.

15 All health test failures are fatal. Once triggered, the Entropy Source will transition into an error state and will no longer provide entropy. To clear the error state, the calling application must unload the failed Entropy Source and instantiate a new instance.

9 Maintenance

16 N/A – The Entropy Source does not require maintenance.

10 User Verification

17 The Entropy Source is considered to provide 256 bits of entropy for each 256-bit of sample, or full entropy.

18 To confirm that the Entropy Source has been configured properly, the developer shall run the 90B Entropy Assessment tools on the noise source outputs in the same manner that was done during the Entropy Source Validation exercise. The steps involved are described below.

10.1 Collecting samples

19 The developer shall run the jitterentropy-hashtime tool included in the Entropy Source package. This script gathers 1,000,000 continuous samples of raw noise data. It also gathers 1,000,000 samples of restart data which comprised of 1000 files each containing the first 1000 samples generated after the Entropy Source has been restarted.

10.2 Raw Data Analysis

20 The developer shall run the noise source samples through NIST's SP 800-90B Entropy Assessment tools. The Entropy Source is configured properly if the assessment yields a min entropy rate of at least 1/OSR bits per sample.

11 Vendor Permissions and Relationship

21 The reuse of this ESV is not permitted.