

Public Use Document

Jentropy Engine

Version: 1.3.6

Document Version 1.3

June 2025

Prepared by:



<https://www.lightshipsec.com/>



Legion of the Bouncy Castle Inc.
(ABN 84 166 338 567)

<https://www.bouncycastle.org>

Guidance

1. Description.....	3
2. Security Boundary.....	3
3. Operational Environment	4
4. Configuration Parameters.....	4
5. Physical Security Mechanisms	5
6. Interfaces	5
7. Min-Entropy Rate.....	6
8. Health Tests	6
8.1 Error codes.....	6
9. Maintenance	6
10. User Verification	6
10.1 Collecting samples	7
10.2 Raw Data Analysis.....	7
11. Vendor Permissions and Relationship	7

1. Description

This is the SP 800-90B Non-Proprietary Public Use Document for the Jentropy Engine version 1.3.6 from Legion of the Bouncy Castle Inc. (hereafter referred to as "the Entropy Source").

The Entropy Source is based on the jitterentropy-library v3.6.1. 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)

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

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

The Entropy Source's security boundary is defined by the portion of the software package that was compiled from the Jitterentropy source code files.

The security boundary is shown in Figure below.

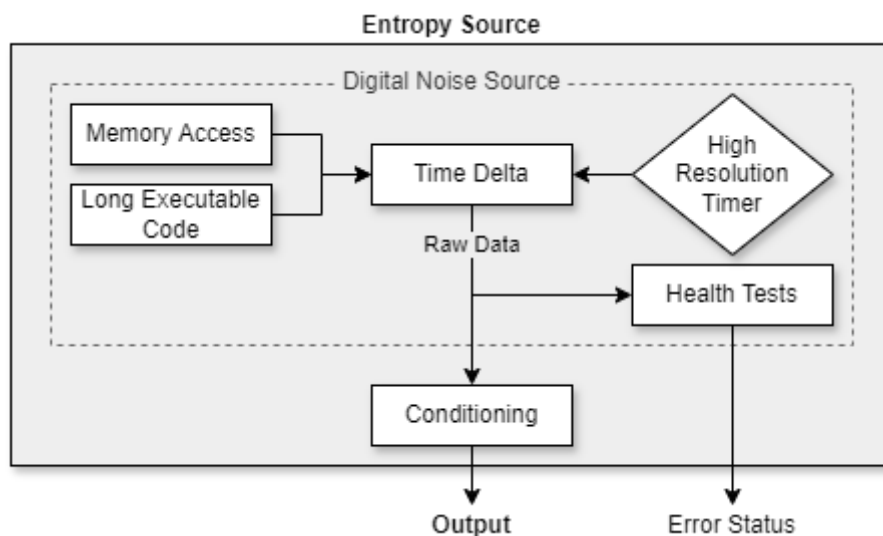


Figure 1. Entropy Source Diagram

The Entropy Source gets its entropy from the time deltas between repeated hash/memory collection operations.

The output is used to seed an SP 800-90A compliant DRBG that is outside of the boundary of the entropy source.

The security boundary of the Entropy Source contains the source code files provided as part of the software delivery. The source code contains API calls that are used by modules requesting entropy from the Entropy Source.

3. Operational Environment

The Entropy Source was tested in the following operational environments:

Table 1: Operational environments

CPU	OS
Intel® Core® i7-1165G7 (Tiger Lake) CPU Clock Speed: 2.80 GHz L1 Cache: 320KB L2 Cache: 5MB L3 Cache: 12MB	Ubuntu 22.04 LTS
ARM Neoverse V2 CPU Clock Speed: 2.80 GHz L1: 128KB L2: 4MB L3: 36MB	Ubuntu 22.04 LTS

4. Configuration Parameters

The table below list the Entropy Source's configuration parameters. These settings must be preserved to comply with the CPU Jitter ESV certificate.

Table 2. CPU Jitter Configuration Parameters

Parameter	Default Value	Location
JENT_CONF_DISABLE_LOOP_SHUFFLE	Defined	jitterentropy.h
JENT_HEALTH_LAG_PREDICTOR	Defined	jitterentropy.h
JENT_RANDOM_MEMACCESS	Defined	jitterentropy.h
ENTROPY_SAFETY_FACTOR	64	jitterentropy.h
JENT_MIN_OSR	3	jitterentropy.h
JENT_MEMORY_SIZE	2 ¹⁷	jitterentropy.h
JENT_MEMORY_BITS	17	jitterentropy.h
JENT_MEMORY_BLOCKS	512	jitterentropy.h
JENT_MEMORY_BLOCKSIZE	128	jitterentropy.h
JENT_MEMORY_ACCESSLOOPS	128	jitterentropy.h
JENT_CONF_ENABLE_INTERNAL_TIMER	Not Defined	Makefile
JENT_APT_WINDOW_SIZE	512	jitterentropy.h
JENT_LAG_WINDOW_SIZE	2 ¹⁷	jitterentropy.h

Parameter	Default Value	Location
JENT_LAG_HISTORY_SIZE	8	jitterentropy.h
JENT_POWERUP_TESTLOOPCOUNT	1024	jitterentropy-base.c
CLEARCACHE	100	jitterentropy-base.c
MIN_HASH_LOOP	0	jitterentropy-noise.c
MIN_ACC_LOOP_BIT	0	jitterentropy-noise.c

Table 3. CPU Jitter Configuration Flags (for jent_entropy_init_ex and jent_entropy_collector_alloc)

Parameter	Argument Name	Meaning
JENT_DISABLE_MEMORY_ACCESS	flags	Disable the memory access timing.
JENT_FORCE_INTERNAL_TIMER	flags	Force use of pthreads-based internal timer, rather than the underlying hardware timer.
JENT_DISABLE_INTERNAL_TIMER	flags	Prevent the use of the pthreads-based internal timer.
JENT_FORCE_FIPS	flags	Force fips_enabled to be set (enabling health test reporting and use of the safety factor).
JENT_MAX_MEMSIZE_*	flags	Allows for run-time selection of the memory region size.
osr	osr	Sets the internal osr rate to max(osr, JENT_MIN_OSR).

5. Physical Security Mechanisms

N/A – The Entropy Source is non-physical

6. Interfaces

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

Interface	API Command	Description
GetEntropy	generateSeed()	Generates a random bit stream and returns it to the caller. This is an indirect call to the Entropy Source via Java SecureRandom API.
GetNoise	–	Not supported
HealthTest	generateSeed()	Initializes the Entropy Source and performs statistical tests to verify that the underlying system offers the properties needed for measuring and collecting entropy. In addition, calls to generateSeed() will also perform the necessary continuous tests as given in NIST SP 800-90A.

Table 4. Interfaces

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

7. Min-Entropy Rate

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 ensure that outputs always possess full entropy.

8. Health Tests

The Entropy Source implements the following health tests:

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

These tests are run when the Entropy Source 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.

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.

8.1 Error codes

The table below lists the Entropy Sources error codes:

Code	Cause
-1	Entropy_collector is NULL
-2	RCT has failed
-3	APT has failed
-4	The timer cannot be initialized
-5	Lag predictor test has failed

Table 5. Error codes

9. Maintenance

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

10. User Verification

The Entropy Source is considered to provide 256 bits of entropy for each 256-bit of sample – or full entropy. To achieve this, the Entropy Source implements an oversampling rate (OSR) of 3, and a safety factor of 64. For each request for 256-bits of entropy, the Entropy Source performs $(256 + 64) \times 3 = 960$ operations and calculates a time delta for each.

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 describe below.

10.1 Collecting samples

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 is comprised of 1000 files each containing the first 1000 samples generated after the Entropy Source has been restarted.

10.2 Raw Data Analysis

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

Use of this Entropy Source is restricted to Legion of the Bouncy Castle Inc per the *reuse status* field on the ESV certificate. Use of this certificate requires written and signed permission from the Legion of the Bouncy Castle Inc point of contact (as indicated on the certificate).

This Entropy Source may only be ported to an operational environment listed in of this document.