

SP 800-90B Non-Proprietary Public Use Document

Ciena Kernel CPU Jitter RNG

Version 3.4.1

Ciena Corporation
7035 Ridge Road
Hanover, MD 21076
USA

Document Version 1.1
August 8, 2024

Revision History

Version	Date	Change
V1.0	8/1/2024	Initial draft
V1.1	8/8/2024	Minor edits

Table of Contents

1. Description 4
2. Security Boundary 4
3. Operating Conditions 5
4. Configuration Settings 5
5. Physical Security Mechanisms 5
6. Conceptual Interfaces 6
7. Min-Entropy Rate 6
8. Health Tests 6
9. Maintenance 6
10. Required Testing 7

1. Description

Ciena developed their CPU Jitter kernel loadable module as a replacement for their prior hardware-based noise source. The module incorporates version 3.4.1 of CPU Jitter RNG to provide high-quality random output through a /dev/hwrng character device available to user space applications. This allows the module to seamlessly replace prior hardware-based RNG sources, while eliminating any need for changes to user-space application behavior.

The Ciena modules implements Kernel CPU Jitter RNG version 3.4.1 without any modifications. Kernel CPU Jitter RNG is a non-physical true random number generator publicly available as an open-source implementation. Kernel CPU Jitter RNG makes no IID claim and thus meets all the requirements for a non-IID entropy source. Table 1 summarizes the Ciena platforms and their respective processors for which testing was performed.

2. Security Boundary

The security boundary of the CPU Jitter implementation contains the source code files provided as part of the software delivery. The source code contains API calls which are used by modules requesting entropy from CPU Jitter.

The noise source is implemented by collecting and accumulating time jitter variances caused by memory accesses and the execution time of a defined set of instructions. The accumulation of jitter variances in the form of concatenated time deltas is fed through the SHA3-256 vetted conditioning function of the entropy source. Successive outputs of SHA3- 256 are accumulated until the requested number of bits has been reached.

The boundary of the CPU Jitter implementation is show in Figure 1.

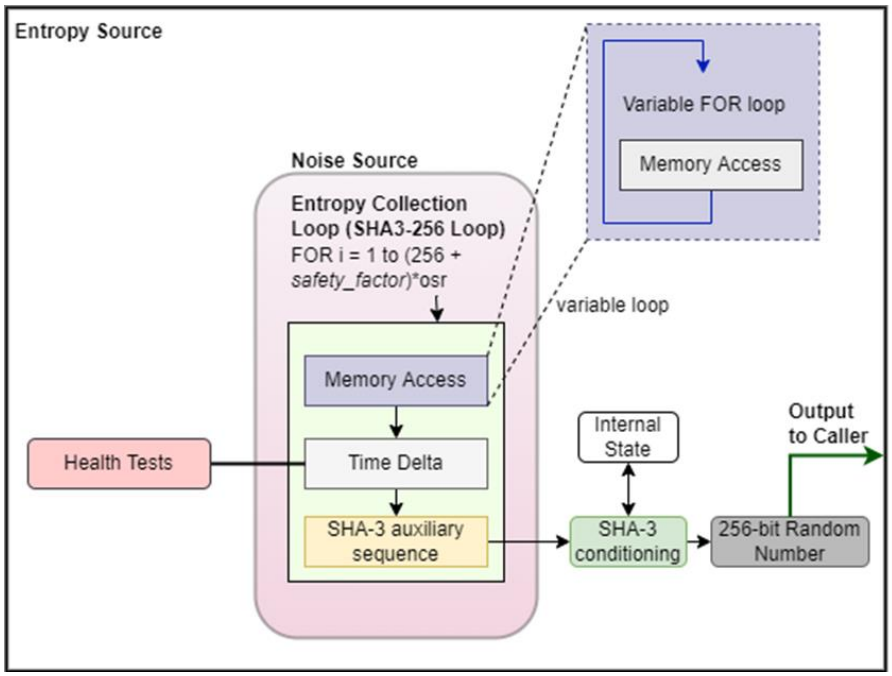


Figure 1 - Entropy Source Diagram

3. Operating Conditions

The noise source is non-physical, and thus the operating conditions are inherited from the operational environment in which the entropy source is installed, as shown in Table 1 below

Platform	Processor	Clock Speed	Cache	Platform Temperature	Platform Voltage (AC)
Waveserver Ai	AMD Zynq Ultrascale+	1.5 GHz	L1 Cache: 32K L2 Cache: 1M L3 Cache: N/A	0°C - 100°C	0.825 V – 0.876 V
Waveserver 5	ARM Marvell CN913x	2.2 GHz	L1 Cache: 32K L2 Cache: 512K L3 Cache: N/A	0°C - 70°C	1.16 V – 1.24 V

Table 1 – Operating Environment and Conditions

4. Configuration Settings

Table 2 summarizes the configuration settings used by the Ciena Kernel CPU Jitter. These settings must be preserved to comply with the CPU Jitter ESV certificate.

Parameter	Default Value	Description
JENT_CONF_DISABLE_LOOP_SHUFFLE	Defined	Disables pseudo-random looping e.g., lower boundary entropy equals upper boundary entropy.
JENT_MEMORY_ACCESSLOOPS	128	Memory access loops
JENT_RANDOM_MEMACCESS	Enabled	Enables random memory access using bit versus blocks
ENTROPY_SAFETY_FACTOR	64	Entropy safety factor
JENT_MIN_OSR	3	Set in code by JENT_CONF_DISABLE_LOOP_SHUFFLE (Enabled: 3 (default), Disabled: 1)
JENT_MEMORY_BITS	17	Number of bits in a memory block (i.e., the memory block size in bits is 2^JENT_MEMORY_BITS)
JENT_MEMORY_SIZE		Memory block size
JENT_FORCE_FIPS	Enabled	Fips mode flag is set

Table 2 - Configuration Settings

5. Physical Security Mechanisms

The Ciena Kernel CPU Jitter RNG is enclosed entirely within the module's cryptographic boundary which is assessed for FIPS 140-3 level 2 Physical Security compliance.

6. Conceptual Interfaces

The entropy source provides the following interfaces:

- jent_read_entropy(): Obtains conditioned entropy for the caller. This is the main function from the entropy source, the one that shall be used to request entropy data. The entropy gathering logic creates 256 bits per invocation. This interface corresponds to the GetEntropy() conceptual interface from SP800-90B.
- jent_hash_time(): Obtains raw noise data for testing purposes. This interface corresponds to the GetNoise() conceptual interface from SP800-90B.

7. Min-Entropy Rate

The Ciena Kernel CPU Jitter RNG modules generate an output that is considered to have full entropy per SP800-90B section 3.1.5. The entropy source uses SHA3-256 which is a vetted conditioning component providing 256-bits of entropy per request of 256-bits, or full entropy.

8. Health Tests

The Ciena Kernel CPU Jitter RNG implements the following health tests:

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

Per the SP 800-90B requirements, health tests are run when CPU Jitter starts, and are then run continuously while it is operating. All tests check for persistent failures based on their respective "cutoff" values, which represent expected error thresholds.

On-demand health tests are performed by deallocating the current Jitter RNG instance and reallocating a new instance. The startup health tests are then re-executed and the samples used for those tests are discarded.

The following error codes can occur:

-1 Entropy_collector is NULL – the entropy source will need to be reinstantiated.
-2 RCT test fails – the entropy source will need to be reinstantiated.
-3 APT test fails – the entropy source will need to be reinstantiated.
-4 The timer cannot be initialized – the entropy source will need to be reinstantiated.
-5 Lag predictor test fails – the entropy source will need to be reinstantiated.

9. Maintenance

There are no maintenance requirements for the Ciena Kernel CPU Jitter RNG modules.

10. Required Testing

Raw data was collected by running the test harness provided within the CPU Jitterentropy v3.4.1 package: invoke_testing.sh. This gathers 1,000,000 8-bit raw and sequential samples through the jent_measure_jitter() function.

Similarly, the Jitterentropy v3.4.1 test harness gathers 1000 samples of 8-bit raw and sequential samples 1000 times killing the Jitterentropy collector each time, thus gathering 1,000,000 samples of raw data for the restart tests.

Raw noise data:
Consists of at least 1,000,000 64-bit samples truncated to 8 bits and must be collected from the operational environment at its normal operating conditions and processed by the SP 800-90B entropy tool that is provided by NIST. The Kernel CPU Jitter RNG states that the source can deliver full entropy if and only if the min-entropy is at least 1/osr (osr=3 is default) bits of entropy per time stamp.

Restart data:
Must be collected at normal operating conditions through the jent_measure_jitter() interface following the restart procedure specified in SP 800-90B (i.e., 1,000 samples from 1,000 restarts each) and processed by the NIST SP 800-90B entropy tool. The minimum of the row-wise and column-wise entropy rate must be more than half that of the raw noise entropy rate.