

Red Hat Enterprise Linux Kernel CPU Time Jitter RNG Entropy Source version 3.4.0

SP 800-90B Non-Proprietary Public Use Document

Document Version: 1.0

Document Date: 2026-09-15

Red Hat, LLC

100 E. Davie Street

Raleigh, NC 27601

www.redhat.com

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

© 2026 Red Hat, LLC / atsec information security corporation

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table of Contents

1 Description 3
2 Security Boundary 4
3 Operating Conditions 4
4 Configuration Settings 5
5 Physical Security Mechanisms 5
6 Conceptual Interfaces 5
7 Min-Entropy Rate 6
8 Health Tests 6
  8.1 Start-up Health Tests 6
  8.2 Continuous Health Tests 6
    8.2.1 Repetition Count Test 6
    8.2.2 Adaption Proportion Test 6
    8.2.3 Stuck Test 7
  8.3 On-Demand Health Tests 7
  8.4 Error Codes 7
9 Maintenance 7
10 User Verification 7
11 Vendor Permissions and Relationship 8

© 2026 Red Hat, LLC / atsec information security corporation

This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 Description

The Red Hat Enterprise Linux Kernel CPU Time Jitter RNG Entropy Source version 3.4.0 is a non-physical entropy source, part of the kernel binary, that feeds the kernel DRBGs. The noise generation of this entropy source is based on the tiny variations in the execution time of the same piece of code. The execution time of this piece of code is made unpredictable by the complexity of the different hardware components that comprise modern CPUs and the different internal states that the operating system can have at a certain point in time.

This entropy source is compliant with SP800-90B (January 2018) and FIPS 140-3 Implementation Guidance (Last Update: April 16, 2026). In particular, compliance with IG D.J (Last Modified Date: November 5, 2021) and IG D.K (Last Modified Date: April 16, 2026) is claimed.

The entropy source was tested on the operational environments listed in Table 2. The noise source was tested under the assumption that its output is non-IID. This entropy source is within a multi-chip standalone embodiment.

Table 1: Entropy source and operational environment summary.

Characteristic	Description
Vendor	Red Hat, LLC
Entropy source name	Red Hat Enterprise Linux Kernel CPU Time Jitter RNG Entropy Source
Standard, IGs	SP800-90B (2018) IG D.J (2021-11-05), IG D.K (2026-04-16), IG D.O (2021-05-04)
Entropy category	Non-Physical (NP)
Entropy source version	3.4.0
Embodiment and type	Multi-Chip Standalone

Table 2: Operational environment and version.

OE#	Manufacturer	Model	Operating System	Processor
1	Dell	PowerEdge R440	Red Hat Enterprise Linux 10	Intel® Xeon® Silver 4216
2	IBM	z16 3931-A01	Red Hat Enterprise Linux 10 on PR/SM Driver Level 51 with Bundle Level H36	IBM® Telum™
3	IBM	IBM 9080-HEX	Red Hat Enterprise Linux 10 on PowerVM FW1040.00 with VIOS 3.1.3.00	IBM POWER10

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2 Security Boundary

The boundary for this non-physical, software-based entropy source is the executable binary file. It is compiled from the C code that implements it (i.e., kernel source code).

Figure 1 depicts the overall design of the entropy source and its core operations.

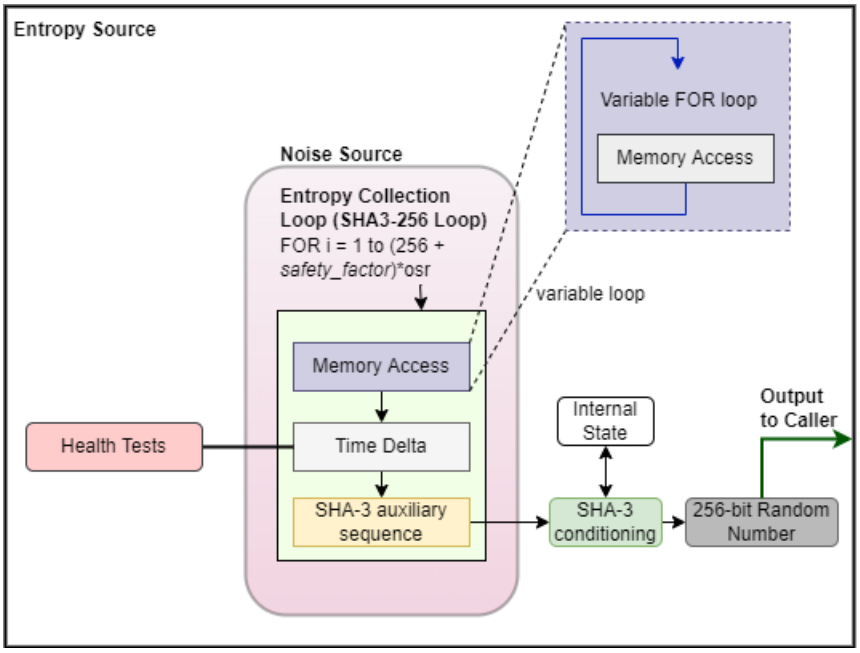


Figure 1: CPU Jitter 3.4.0 Design

The noise source is implemented by collecting and accumulating time variances of variable memory accesses and variances in the execution time of a defined set of instructions, which includes an implementation of SHA3-256. The time variances, in the form of time deltas, are collected and conditioned to 256-bits by the SHA3-256 vetted conditioning function.

The output of this conditioning function is 256 bits of full entropy.

If the Repetition Count Test (RCT) or the Adaptive Proportion Test (APT) health tests fail, the noise data is discarded, the entropy source halts without outputting any data, and a failure code is returned to the caller. If the health test failure is permanent, the kernel which contains this entropy source will panic.

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 3 below.

Table 3: Operating Conditions for each Operational Environment

OE #	Manufacturer / Model	Temperature Range	Voltage Range	Humidity Range	Clock Speed	Cache Sizes
1	Dell PowerEdge R440	10C° – 35C°	100 – 240 Vac	10 to 80%	2.1 GHz	L1d: 16x32 KB L1i: 16x32 KB L2: 1x16 MB L3: 22 MB

© 2026 Red Hat, LLC / atsec information security corporation

This document can be reproduced and distributed only whole and intact, including this copyright notice.

OE #	Manufacturer / Model	Temperature Range	Voltage Range	Humidity Range	Clock Speed	Cache Sizes
2	IBM z16 3931-A01	5C° – 40C°	200 – 240 V	8 to 85%	5.2 GHz	L1d: 6x128 KB L1i: 6x128 KB L2: 6x32 MB L3: 256 MB
3	IBM 9080-HEX	5C° - 40C°	200 - 208 / 220 – 240 Vac	8 to 85%	3.4 GHz	L1d: 2x64 KB L1i: 2x96 KB L2: 2x2 MB L3: 2x8 MB

4 Configuration Settings

The following noise source parameters are configured in the kernel configuration file:

CONFIG_CRYPTO_JITTERENTROPY_MEMSIZE_2 (default: set)

CONFIG_CRYPTO_JITTERENTROPY_MEMSIZE_128 (default: not set)

CONFIG_CRYPTO_JITTERENTROPY_MEMSIZE_1024 (default: not set)

CONFIG_CRYPTO_JITTERENTROPY_MEMSIZE_8192 (default: not set)

CONFIG_CRYPTO_JITTERENTROPY_MEMORY_BLOCKS (default: 64)

CONFIG_CRYPTO_JITTERENTROPY_MEMORY_BLOCKSIZE (default: 32)

CONFIG_CRYPTO_JITTERENTROPY_OSR (default: 3)

None of these values should be altered. Any changes to these parameters would require a recompilation of the Linux kernel image, which results in the validated entropy source described in the document not being accessed.

5 Physical Security Mechanisms

The noise source is non-physical. The physical security mechanisms only apply to the hardware component of the operational environment in which the entropy source is installed, and thus the entropy source inherits those mechanisms.

6 Conceptual Interfaces

The entropy source provides the following interfaces:

- jent_kcapi_init(): Initializes the entropy source context, including the configuration of the CPU timer or the internal timer. This function returns 0 if the entropy source context is initialized successfully.
- 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. This interface corresponds to the GetEntropy() and HealthTest() conceptual interfaces from SP 800-90B. This function returns 0 if the entropy was generated successfully.
- jent_measure_jitter(): Obtains raw noise data for testing purposes. This interface corresponds to the GetNoise() conceptual interface from SP 800-90B. This function returns the result of the stuck test.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

- jent_kcapi_cleanup(): zeroizes and frees the given entropy collector instance.

7 Min-Entropy Rate

The entropy source provides 256 bits of entropy per 256-bit conditioned output block (i.e., full entropy).

8 Health Tests

The entropy source implements the following categories of health tests:

- Start-up health tests.
- Continuous health tests.
- On-demand health tests.

The continuous health tests implement a Repetition Count Test (RCT), an Adaptive Proportion Test (APT), and a developer-defined Stuck Test.

As allowed by Section 4.3 of SP 800-90B, the entropy source defines two types of health test failures for the RCT and the APT: intermittent failures and permanent failures.

An intermittent failure is characterized by a false positive probability $\alpha_{\mathrm{i}} = 2^{-30}$, which lies within the recommended range of $2^{-20} \leq \alpha \leq 2^{-40}$. When an intermittent failure is detected, the CPU Jitter RNG is automatically reset (which includes clearing the entropy pool and resetting the conditioning component), and the caller is notified of this failure.

Permanent failures are characterized by a false positive probability of $\alpha_{\mathrm{p}} = 2^{-60}$. When a permanent failure is detected, the CPU Jitter RNG is also reset, but the Linux kernel that contains this entropy source immediately enters the error state. In practice, this results in a kernel panic.

8.1 Start-up Health Tests

Start-up tests conduct the same set and parameters of the continuous health tests on 1024 time deltas. The data is discarded after the start-up tests have completed successfully. Any health test failure during the start-up tests will always be treated as a permanent failure.

8.2 Continuous Health Tests

8.2.1 Repetition Count Test

The Repetition Count Test conforms to SP 800-90B section 4.4.1.

- $H = 1/3$ bit of entropy per 64-bit time delta.
- Intermittent failure alpha value $\alpha_i = 2^{-30}$.
- Permanent failure alpha value $\alpha_p = 2^{-60}$.
- Intermittent failure cutoff value $C_i = 91$.
- Permanent failure cutoff value $C_p = 181$.

8.2.2 Adaption Proportion Test

The Adaptive Proportion test conforms to SP 800-90B section 4.4.2.

- $W = 512$.

© 2026 Red Hat, LLC / atsec information security corporation

This document can be reproduced and distributed only whole and intact, including this copyright notice.

- $H = 1/3$ bit of entropy per 64-bit time delta.
- Intermittent failure alpha value $\alpha_i = 2^{-30}$.
- Permanent failure alpha value $\alpha_p = 2^{-60}$.
- Intermittent failure cutoff value $C_i = 459$.
- Permanent failure cutoff value $C_p = 479$.

8.2.3 Stuck Test

The stuck test computes the first, second and third discrete derivatives of the time delta (the noise sample). If any of these derivatives are zero, then the received time delta is considered stuck. In this case the noise sample is not used as input to the conditioning function, and another sample needs to be fetched. The stuck test then triggers the RCT for further processing, such that the stuck test itself does not trigger a health test failure. The second derivative is in fact the RCT itself.

8.3 On-Demand Health Tests

On-demand health tests of the noise source may be performed by rebooting the operational environment, which results in the immediate execution of the start-up tests. The data used for the on-demand health tests is discarded after successful completion.

8.4 Error Codes

When entropy is requested from the jent_kcapi_random() function, the return code indicates the status of the entropy source. A return code of -EAGAIN indicates an intermittent health test failure, whereas a return code of -EFAULT indicates a permanent health test failure. The following error codes are defined for jent_read_entropy():

- -1 entropy_collector is NULL or the generation failed.
- -2 Intermittent health failure.
- -3 Permanent health failure.

9 Maintenance

There are no maintenance requirements as this is a software-based entropy source.

10 User Verification

To test the entropy source, raw data samples must be collected using both timer options using a test harness that is capable of accessing the jent_measure_jitter() noise interface from the entropy source. The test harness and accessory tools must be supplied by the vendor.

Raw noise data samples consisting of at least 1,000,000 bits 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 expected min-entropy rate must approach the one in Section 7.

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.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

The operating environment information can be verified as follows:

- Operating system: **cat /etc/os-release** command
- Processor name and model: **lscpu** command
- Hardware: **dmidecode** command

11 Vendor Permissions and Relationship

The ESV certificate is “Reuse restricted to vendor”. Parties other than the vendor may only use the certificate with written and signed permission from the vendor’s point of contact (as indicated on the ESV certificate).

This document can be reproduced and distributed only whole and intact, including this copyright notice.