

Chainguard

Chainguard Kernel CPU Jitter Entropy Source version 1.0.0-r0

SP 800-90B Non-Proprietary Public Use Document

Document Version: 1.1

Document Date: 2026-06-27

Chainguard, Inc.

810 7th St S

Kirkland, WA 98033

www.chainguard.dev

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

© 2026 Chainguard, Inc. / atsec information security corporation

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

Table of Contents

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

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 Description

The Chainguard Kernel CPU Jitter Entropy Source version 1.0.0-r0 is a non-physical, software-based entropy source. Its purpose is to feed 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.

The entropy source is validated as conformant to SP 800-90B (2018) and FIPS 140-3 IG D.J (November 5, 2021), D.K (April 16, 2026), D.O (September 02, 2025).

The entropy source was tested on the operational environments listed in Table 1. 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: Operational environment and version.

Manufacturer	Model	Operational Environment and Version	Processor
Amazon	EC2 m7g.metal	Chainguard OS 20230214 with Linux 6.12	Amazon Neoverse V1 AWS Graviton3
Amazon	EC2 m7g.metal	Chainguard OS 20230214 with Linux 6.18	Amazon Neoverse V1 AWS Graviton3
Intel	EC2 m7i.metal-24xl	Chainguard OS 20230214 with Linux 6.12	Intel Golden Cove Xeon Platinum 8488C (Sapphire Rapids)
Intel	EC2 m7i.metal-24xl	Chainguard OS 20230214 with Linux 6.18	Intel Golden Cove Xeon Platinum 8488C (Sapphire Rapids)

Chainguard OS 20230214 is a host OS image.

2 Security Boundary

The boundary for this non-physical, software-based entropy source is the three pre-compiled object files jitterentropy.o, sha3_jitterentropy.o, and jitterentropy-optimized.o and the jitterentropy-shim.o object file that is re-compiled for each platform in Table 1.

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

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.

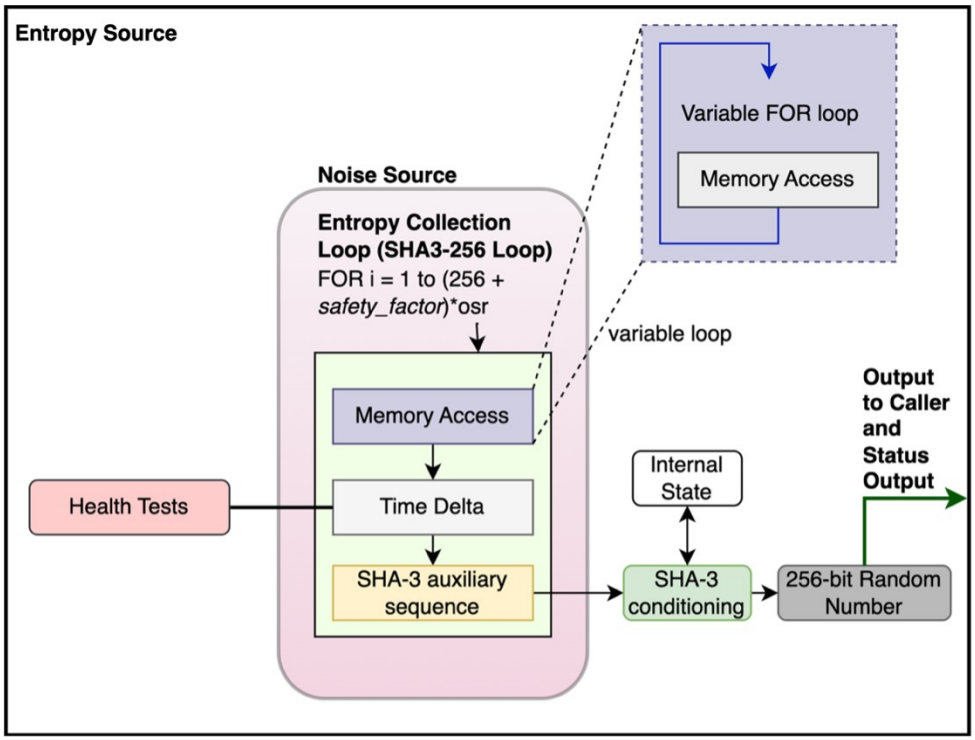


Figure 1: CPU Jitter 1.0.0-r0 Design

The noise source is implemented by collecting and accumulating variances of the execution time of a defined set of instructions. The time jitter of the execution time is measured over 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 output of the noise source is conditioned by a SHA3-256 conditioning component.

The noise source and the SHA3-256 conditioning component are compiled into a single kernel object file, jitterentropy_rng.ko, that is built into the kernel at compile time. The entropy source runs in kernel space.

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.

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 2 below.

Table 2: Operating Conditions for each Operational Environment

Manufacturer / Model	Temperature	Voltage	Humidity	Clock Speed	Cache Sizes
EC2 m7g.metal	0°C-45°C	12V	60%	2.6 GHz	L1d: 64 kB per core (64 cores) L1i: 64 kB per core (64 cores) L2: 1 MB per core

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Manufacturer / Model	Temperature	Voltage	Humidity	Clock Speed	Cache Sizes
					(64 cores) L3: 32 MB (shared across all cores)
EC2 m7i.metal-24xl	0°C-45°C	12V	60%	0.80-3.80 GHz	L1d: 48 kB per core (48 cores) L1i: 32 kB per core (48 cores) L2: 2 MB per core (48 cores) L3: 105 MB (shared across all cores)

4 Configuration Settings

No configuration settings are required.

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.
- getrandom(): System call interface that requests conditioned entropy from the entropy source. Internally, it invokes jent_kcapi_random() which implements the GetEntropy() and HealthTest() conceptual interfaces from SP 800-90B. The call blocks if insufficient entropy is available and returns the number of bytes placed in the buffer on success.
- jent_measure_jitter(): Obtains raw noise data for testing purposes. This interface corresponds to the GetNoise() conceptual interface from SP 800-90B.
- 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 continuous health tests:

- Start-up health tests
- Continuous health tests

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.

- On-demand health tests

The continuous health tests implement a Repetition Count Test (RCT), an Adaptive Proportion Tests (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.

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.

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 health tests conduct the Adaptive Proportion Test and the Repetition Count Test over the course 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, which results in a kernel panic.

8.2 Continuous Health Tests

8.2.1 Repetition Count Test

The Repetition Count Test conforms to SP800-90B, Section 4.4.1.

- Assessed min-entropy of noise source $H = 1/3$ bit of entropy per 64-bit time delta.
- Intermittent RCT failure alpha value $\alpha = 2^{-30}$.
- Permanent RCT failure alpha value $\alpha = 2^{-60}$.
- Intermittent RCT failure cutoff value $C = 90$.
- Permanent RCT failure cutoff value $C = 180$.

8.2.2 Adaptive Proportion Test

The Adaptive Proportion Tests conforms to SP800-90B, Section 4.4.2.

- Window size $W = 512$.
- Intermittent APT failure alpha value $\alpha = 2^{-30}$.
- Permanent APT failure alpha value $\alpha = 2^{-60}$.
- Intermittent APT failure cutoff value $C = 459$.
- Permanent APT failure cutoff value $C = 479$.

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 SHA3-256 conditioning function, and another sample needs to be fetched (the entropy value is not counted). 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.

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.

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 through 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_kcapi_random():

- -3: Permanent health test failure.
- -2: Intermittent health test failure.
- -1: All other errors.

9 Maintenance

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

10 User Verification

To identify the entropy source, the following command can be run:

```
$ cat /etc/os-release
```

The output of the above command should be confirmed to contain “Chainguard OS 20230214”.

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

Raw noise data consisting of at least 1,000,000 64-bit samples truncated to 8 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.

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).

© 2026 Chainguard, Inc. / atsec information security corporation
This document can be reproduced and distributed only whole and intact, including this copyright notice.