

Userspace CPU Time Jitter RNG Entropy Source version 3.4.0

SP 800-90B Non-Proprietary Public Use Document

Document Version: 1.0

Document Date: 2026-09-14

SAP SE

Dietmar-Hopp-Allee 16

Walldorf, Baden-Wuerttemberg 69190

www.sap.com

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

© 2026 SAP SE / 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 3
3 Operating Conditions 4
4 Configuration Settings 5
5 Physical Security Mechanisms 5
6 Conceptual Interfaces 5
7 Min-Entropy Rate 5
8 Health Tests 5
  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 6
  8.3 On-Demand Health Tests 6
  8.4 Error Codes 6
9 Maintenance 7
10 User Verification 7
11 Vendor Permissions and Relationship 7

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

1 Description

The Userspace CPU Time Jitter RNG Entropy Source version 3.4.0 is a non-physical entropy source that is implemented in the kernel binary. Its purpose is to feed user space applications. 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	SAP SE
Entropy source name	Userspace CPU Time Jitter RNG Entropy Source
Standard, IGs	SP800-90B (2018) IG D.J (2021-11-05), IG D.K (2023-03-17), 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: Tested operational environments.

OE#	Operating System	Hardware	Processor	Cache Sizes
1	Garden Linux 1877	EC2 c8i.metal-48xl bare metal	Intel® Xeon® 6975P-C Granite Rapids	L1d: 96x48 KiB L1i: 96x64 KiB L2: 96x2 MiB L3: 1x480 MB
2	Garden Linux 1877	EC2 c8g.metal-24xl bare metal	Amazon Web Services (AWS) Graviton4	L1d: 96x64 KiB L1i: 96x64 KiB L2: 96x2 MiB L3: 1x36 MiB

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). The noise

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

source, the SHA3-256 conditioning component, and the HMAC-SHA-512 DRBG conditioning component are implemented as part of the kernel executable.

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

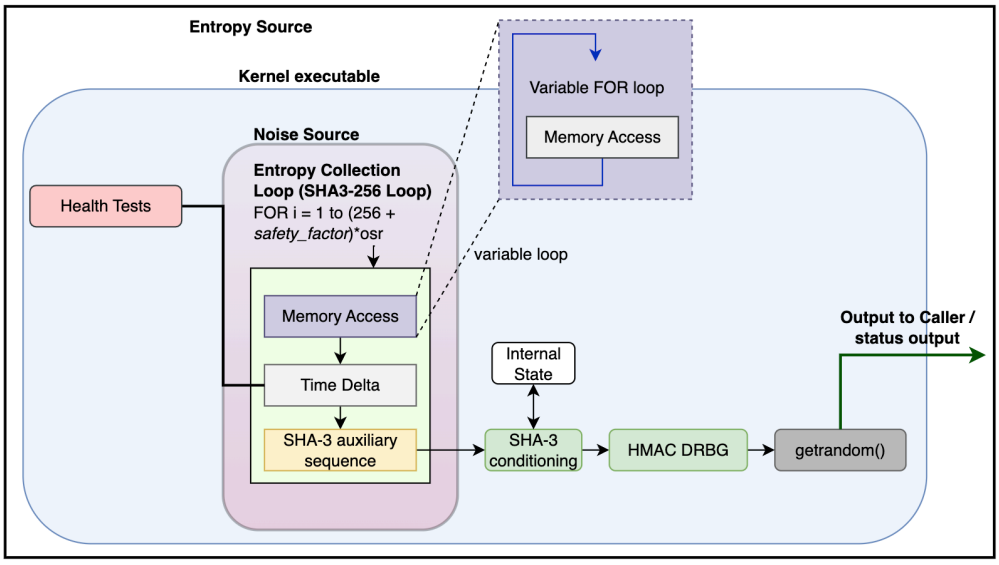


Figure 1: CPU Jitter 3.4.0 Design

The noise source is implemented by collecting and accumulating 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 noise source is processed through a chain of two conditioning components:

- a SHA3-256 function that works as the first conditioning component, using the noise source as input.
- a NIST SP 800-90Ar1-compliant HMAC-SHA-512 DRBG that works as the second conditioning component, using the SHA3-256 output as input.

User space applications shall interface with the kernel relying on the getrandom() syscall only with the GRND_RANDOM flag, as described in Section 4.

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#	Hardware	Temperature	Voltage	Humidity	Clock Speed
1	EC2 c8i.metal-48xl bare metal	-20C° – 60C°	12V	60%	2.7GHz – 3.9GHz
2	EC2 c8g.metal-24xl bare metal	-20C° – 60C°	12V	60%	2.8 GHZ

© 2026 SAP SE / atsec information security corporation

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

4 Configuration Settings

The caller shall use the getrandom system call with the GRND_RANDOM flag set. When this flag is set, the DRBG conditioning component complies with the requirements described in FIPS 140-3 IG D.K, Resolution 5.

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:

- getrandom() with the GRND_RANDOM flag set: Obtains conditioned entropy for the caller and provides health test status. 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 interface from SP800-90B.
- jent_measure_jitter(): Obtains raw noise data for testing purposes. This interface corresponds to the GetNoise() conceptual interface from SP800-90B.

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.

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.

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 $\alpha_i = 2^{-30}$, 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 $\alpha_p = 2^{-60}$. 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.

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

The continuous-health tests are applied to each new time delta obtained from the noise source.

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.

- $W = 512$.
- $H = 1/3$ 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 getrandom() function, the return code indicates the status of the entropy source. A return code of -EAGAIN indicates an intermittent health test

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

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.

The operating environment information can be verified as follows:

- Operating system: **cat /etc/os-release** command
- Processor: **lscpu** command
- Hardware (including EC2 instance type): **dmidecode** command

11 Vendor Permissions and Relationship

The ESV certificate is “Open for Reuse”.

© 2026 SAP SE / atsec information security corporation

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