AWS-LC CPU Time Jitter RNG Entropy Source version 3.4.0

SP 800-90B Non-Proprietary Public Use Document

Document Version 1.0

Document Date: 2023-07-05

Prepared by:
atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of Contents

1 Description

The AWS-LC CPU Time Jitter RNG version 3.4.0 is a non-physical entropy source. 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 uses an external timer provided by the operational environment on which the entropy source was tested. The tested operational environments are listed in Table 1. The noise source was tested under the assumption that its output is non-IID.

Table 1: Operational environment and version.

Manufacturer	Model	Operational Environment and Version	Processor
Amazon Web Services (AWS)	c5.metal	Amazon Linux 2	Intel(R) Xeon(R) Platinum 8259CL
	c5.metal	Amazon Linux 2	Intel(R) Xeon(R) Platinum 8275CL
		Amazon Linux 2023	
		Ubuntu 22.04	
	c7g.metal	Amazon Linux 2	AWS Graviton3
		Amazon Linux 2023	
		Ubuntu 22.04	

2 Security Boundary

The entropy source executes as a statically linked library on the general-purpose operating systems listed in Table 1. The security boundary for this non-physical, software-based entropy source is the library in which it resides.

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

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 accumulated and mapped down to 256-bits by the SHA3-256 vetted conditioning function which outputs 256 bits of full entropy.

If the Repetition Count Test (RCT), Adaptive Proportion Test (APT), or Lag Predictor Test 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.

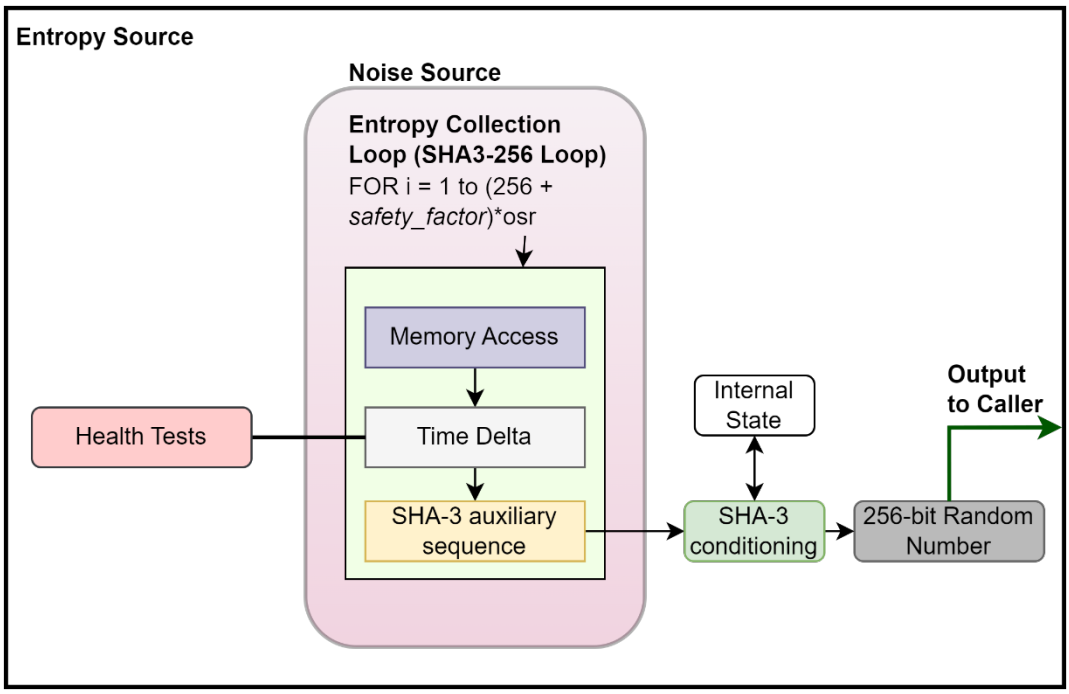

Figure 1: CPU Jitter 3.4.0 Design

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
AWS c5.metal	0C° – 45C°	12V	60%	2.5 GHz	L1d: 24x32 KB L1i: 24x32 KB L2: 24x1 MB L3: 35.75 MB
				3.0 GHz	L1d: 48x32 KB L1i: 48x32 KB L2: 48x1 MB L3: 2x35.75 MB
AWS c7g.metal	0C° – 45C°	12V	60%	2.6 GHz	L1d: 64x64 KB L1i: 64x64 KB L2: 64x1 MB L3: 32 MB

4 Configuration Settings

The AWS-LC CPU Time Jitter RNG version 3.4.0 obtains time jitter noise from the CPU external timer. If the CPU does not support a high-resolution timer, the entropy source will not function. No configuration settings are supported.

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_entropy_init(): Initializes the entropy source context, including the configuration of the CPU timer or the internal timer.
- jent_read_entropy(): 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() 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.
- jent_entropy_collector_free(): zeroizes and frees the given entropy collector instance.

7 Min-Entropy Rate

The noise source provides an entropy rate for each time delta of $H_{submitter}$ = 1/3 bits.

The entropy source collects 960 time deltas of 64 bits each from the noise source in order to input to the SHA3-256 vetted conditioning component. This corresponds to 320 bits of entropy. In other words, the entropy source uses an oversampling rate of three samples in order to collect one bit of entropy; the 320 bits of input entropy considers the expected output entropy of 256 bits and a safety factor of 64 bits of additional entropy defined in FIPS 140-3 IG D.K, Resolution 19.

This design ensures that the entropy source can provide full entropy, that is, 256 bits of entropy in its output.

The oversampling rate (osr) and safety factor are fixed values defined in the implementation and cannot be altered.

8 Health Tests

The entropy source implements the following continuous health tests:

- Repetition Count Test conforming to SP 800-90B section 4.4.1.
  - $H = 1/3$ bits of entropy per 64-bit sample.
  - Alpha value $\alpha = 2^{-30}$.
  - Cutoff value $C = 31$.
- Adaptive Proportion test conforming to SP 800-90B section 4.4.2.
  - $W = 512$

- $H = 1/3$ bits of entropy per 64-bit sample
- Alpha value $\alpha = 2^{-30}$.
- Cutoff value $C = 458$.

- Stuck (Non-Permanent) Test: The stuck test computes the first, second and third discrete derivatives of the time value that will be processed by SHA3-256. If any of these derivatives are zero, then the received time delta is considered stuck. In this case the input state to SHA3-256 is not updated, and the entropy value is not counted. The stuck test then triggers the RCT for further processing. The second derivative is in fact the RCT itself.
- Lag Predictor Test: The goal of this test is to detect a failure mode in which the outputs may become mostly deterministic. In essence, this test constructs a scoreboard and tracks the number of times that a subpredictor was correct. The subpredictor that scored the most correct predictions is used to predict the next value of a series. The lag predictor test is configured in this entropy source with the following parameters:
  - $\alpha = 2^{-22}$
  - Window size $W = 131072$
  - Lag history size $L = 8$
  - Global cutoff $= \mathrm{InverseBinomalCDF} = \mathrm{CRITBINOM}\left(W - L, 2^{-1/osr}, 1 - \alpha\right)$
  - Local cutoff = 111

The continuous health tests are applied to each new sample obtained from the noise source. Whenever an RCT, APT, or Lag Predictor Test failure is detected during operation, the noise data is discarded, the entropy source halts without outputting any data, and a failure code is immediately returned to the caller. The stuck test is considered non-permanent, as positive stuck tests will be registered but will not immediately halt the entropy source.

Startup tests conduct the same set and parameters of the continuous health tests on 1024 samples of noise data. The data is discarded after the startup tests have completed successfully.

On-demand health tests of the noise source may be performed by re-initializing the entropy source, which results in the immediate execution of the start-up tests. Similar to the startup tests, the data used for the on-demand health tests is discarded after successful completion.

The following error codes are defined for jent_read_entropy():

- -1 entropy_collector is NULL
- -2 RCT failed
- -3 APT failed
- -4 The timer cannot be initialized
- -5 Lag Predictor Test failed

9 Maintenance

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

10 Required Testing

To test the entropy source, raw data samples must be collected using a test harness that is capable of accessing the jent_hash_time() noise interface from the entropy source. The test harness and accessory tools 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_hash_time() 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.

In order to collect samples output from the vetted SHA3-256 conditioning component, a test harness that is capable of accessing the jent_read_entropy() interface is required.