SP 800-90B Non-Proprietary Public Use Document for CPU Time Jitter RNG

Entropy Source

Version 3.4.0

***Document Version* 1.2**

Last update: 2023-03-10

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

1. Description

The Kernel 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.

The noise source was tested under the assumption that its output is non-IID.

The entropy source was tested on the operational environments listed in Table 1. The operating system in each instance is BIG IP 16.1.3.1.

F5 Model / Virtual Environment	Processor
VIPRION B2250	Intel® Xeon® E5-2658v2
VIPRION B4450	Intel® Xeon® E5-2658v3
BIG-IP i4600, BIG-IP i4800	Intel® Xeon® D-1518
BIG-IP i5600, BIG-IP i5800, BIG-IP i5820-DF	Intel® Xeon® E5-1630v4
BIG-IP i7600, BIG-IP i7800, BIG-IP i7820-DF	Intel® Xeon® E5-1650v4
BIG-IP i10600, BIG-IP i10800	Intel® Xeon® E5-1660v4
BIG-IP i11600-DS, BIG-IP i11800-DS	Intel® Xeon® E5-2695v4
BIG-IP i15600, BIG-IP i15800 BIG-IP i15820-DF	Intel® Xeon® E5-2680v4
Hyper-V	Intel®Xeon® E5-2660v3
VMware	Intel®Xeon® E5-2670
KVM	Intel®Xeon®E5-2690v4
Hyper-V	Intel®Xeon® Silver 4309Y

Table 1: Operational Environments for CPU Jitter RNG 3.4.0

2. Security Boundary

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

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.

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

If the Repetition Count Test (RCT) or the Adaptive Proportional 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 (typically causing a kernel panic, since this is a kernel space non-physical entropy source).

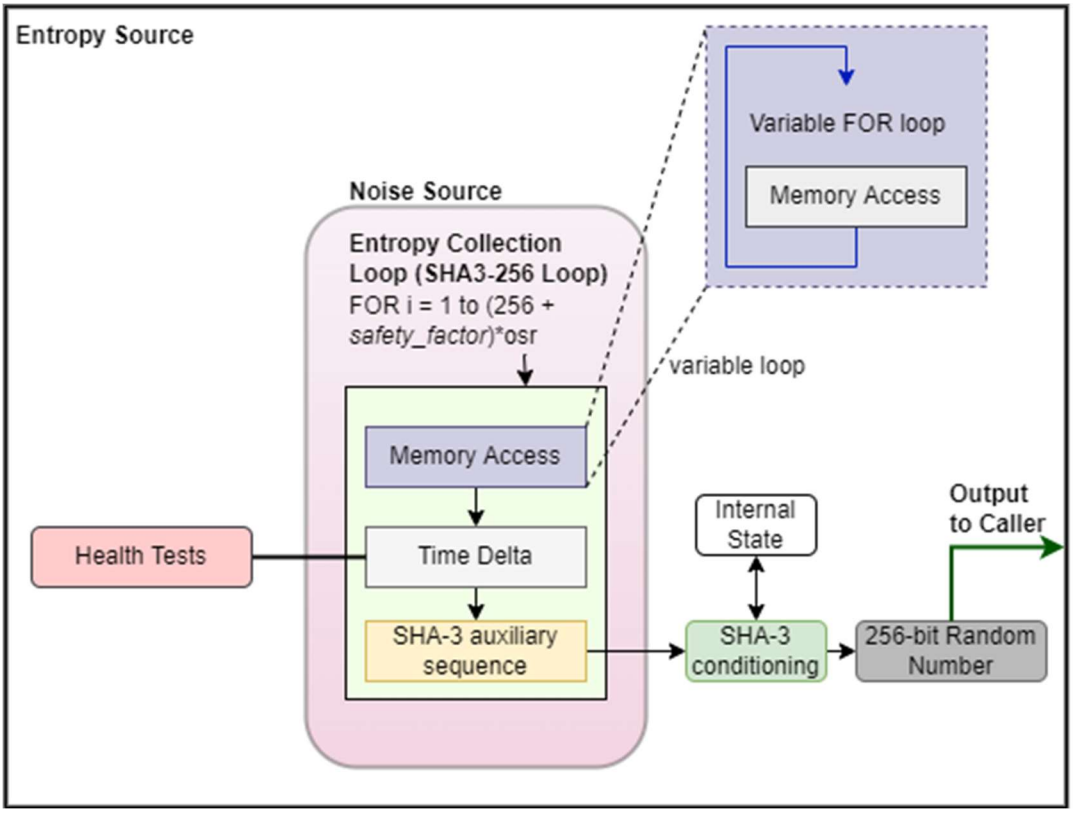

Figure 1: Security Boundary of the Entropy Source

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.

4. Configuration Settings

The only configurable setting allows the choice of using the external CPU timer which exists on most modern CPUs or to disable this functionality and to use an internal timer provided by the CPU Jitter code. In this entropy source, only the external CPU timer is allowed. Therefore, this ESV certificate is only valid when the external CPU timer is used and the internal timer is disabled.

The timer settings are defined in the file jitterentropy.h:

#define JENT_CONF_ENABLE_INTERNAL_TIMER /* enables internal non-CPU timer. This setting shall be disabled */

#define JENT_FORCE_INTERNAL_TIMER (1<<3) /* forces the use of the internal timer. This setting shall be disabled */

#define JENT_DISABLE_INTERNAL_TIMER (1<<4) /* disable the potential use of the internal timer. This setting shall be enabled */

There are other settings in jitterentropy.h that can be changed but this would invalidate the ESV certificate. The following are some examples:

#define JENT_FORCE_FIPS (1<<5) /* Force FIPS compliant mode including full SP 800-90B compliance */

#define ENTROPY_SAFETY_FACTOR 64

#define JENT_MEMORY_BITS 17

#define JENT_MEMORY_SIZE (UINT32_C(1)<<JENT_MEMORY_BITS)

#define JENT_MEMORY_BLOCKS 512

#define JENT_MEMORY_BLOCKSIZE 128

#define JENT_MEMORY_ACCESSLOOPS 128

5. Physical Security Mechanisms

The noise source is non-physical. The physical security mechanisms 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_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

$H_{submitter}$ = 1/3 bit per 64 bits of time delta. The original noise sample size has a length of 64 bits, and it is further reduced to 8 bits for assessment and health testing purposes. The 64-bit time deltas are concatenated to gather sufficient entropy for the SHA3-256 vetted conditioning function.

The entropy source provides an output of 256 bits. This output provides 256 bits of entropy.

8. Health Tests

The entropy source implements the following continuous health tests:

- Repetition Count Test conforming to SP 800-90B section 4.4.1.
  - $H = 1$ bit of entropy per 8-bit sample.
  - alpha value of $\alpha = 2^{-30}$.
  - Cutoff value $C = 31$.
- Adaptive Proportion test conforming to SP 800-90B section 4.4.2.
  - $W = 512$
  - $H = 1$ bit of entropy per 8-bit sample
  - alpha value of $\alpha = 2^{-30}$.
  - Cutoff value $C = 325$.
- 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: $window\_size = 131072$
  - Lag history size: $lag\_history\_size = 8$
  - Global cutoff = $InverseBinomialCDF = CRITBINOM(n = window_{size} - lag_{history_{size}}; p = 2^{-\frac{1}{osr}}; 1 - \alpha)$
  - Local cutoff = 111

The continuous-health tests are applied to each new sample obtained from the noise source. Whenever a failure is detected during the health testing specifically for the RCT and APT, entropy data is not returned to the caller; instead, a failure code is returned to enable the caller to acknowledge the failure. The entropy source then halts and will refuse new requests for entropy. Upon return of the failure code, the caller shall attempt to reset or reboot the entropy source or return an error to its own operator. 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 rebooting the operational environment, which results in the immediate execution of the start-up tests. Typically, this entropy source designed for user space cannot be reloaded without restarting the executable. Similarly, the data used for the on-demand health tests are discarded after successful completion.

The following error codes are defined for jent_read_entropy():

-1 entropy_collector is NULL

-2 RCT failed

-3 APT test failed

-4 The timer cannot be initialized

-5 LAG failure.

9. Maintenance

There are no maintenance requirements as the entropy source is software based.

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_read_entropy() noise interface from the entropy source, and also the entropy output of the entropy source from user space. 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_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.