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

Version 2.2.0

Document Version: 1.2

Document Date: 2023-02-03

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 2.2.0 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 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.

The CPU Jitter RNG runs in kernel space on the SUSE Linux Enterprise Server 15 SP4 operating system in the following operational environments.

Table 1: Operational environment and version

Manufacturer	Model	Operational Environment and Version	Processor
Supermicro	Super Server X11DDW-L	SUSE Linux Enterprise Server 15 SP4 Kernel	Intel(R) Xeon(R) Silver 4215R
GIGABYTE	R181-Z90-00	SUSE Linux Enterprise Server 15 SP4 Kernel	AMD EPYC(TM) 7371
GIGABYTE	G242-P32-QZ	SUSE Linux Enterprise Server 15 SP4 Kernel	Ampere(R) Altra(R) Q80-30
IBM	z15 Model T01	SUSE Linux Enterprise Server 15 SP4 Kernel	IBM z15
IBM	IBM 9080-HEX	SUSE Linux Enterprise Server 15 SP4 Kernel	IBM Power10

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.

The noise source is implemented by collecting and accumulating variances of the execution time of a defined set of instructions. The accumulation is done with the assistance of a Linear-Feedback Shift Register (LFSR) function that works as the conditioning component of the entropy source, as well as the entropy pool that is used to accumulate the acquired entropy. The time jitter of the execution time is measured over the processing logic of the LFSR and functions that support the implementation.

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).

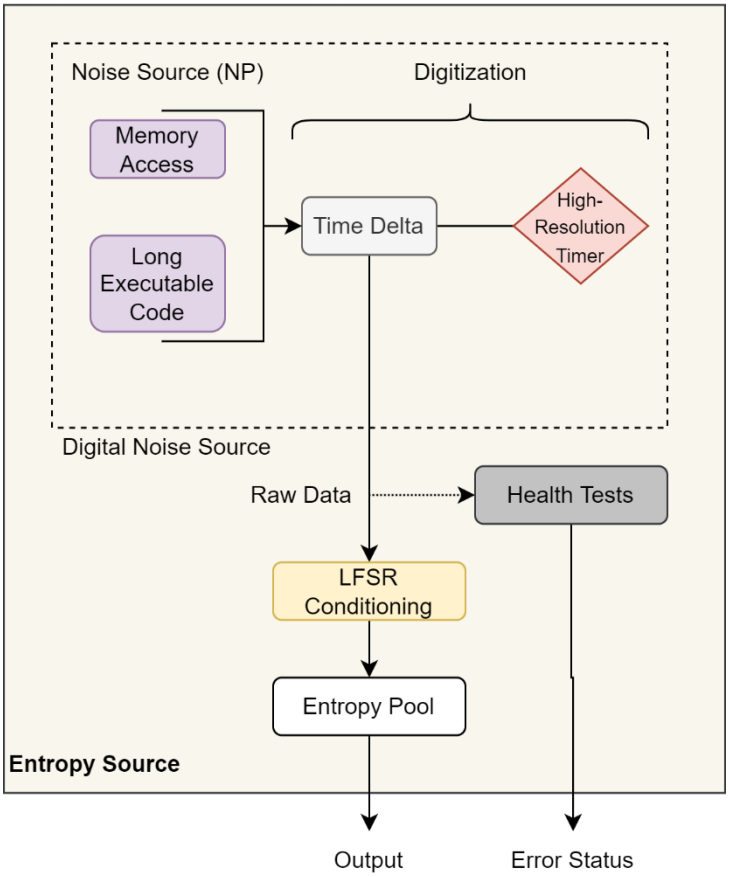

Figure 1: Security boundary of the entropy source.

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.

Table 2 - Operating Conditions

Manufacturer / Model	Temperature	Voltage	Humidity	Clock Speed	Cache Sizes
Supermicro Super Server X11DDW-L	10C° – 35C°	+12 V	8% - 90%	3.2 GHz	L1: 8x32 KB L2: 8x1 MB L3: 11 MB
GIGABYTE R181-Z90-00	10C° – 35C°	100-240 V	20% - 95%	3.1 GHz	L1: 16x64 KB or 16x32 KB L2: 16x512 KB L3:64 MB

Manufacturer / Model	Temperature	Voltage	Humidity	Clock Speed	Cache Sizes
GIGABYTE G242-P32-QZ	10C° – 35C°	100-240 V	8% - 80%	3.3 GHz	L1: 80x64 KB L2: 80x1 MB L3: 80x32 MB
IBM z/15 Model T01	5C° - 40C°	200-240 V	8% - 85%	5.2 GHz	L1: 128 KB L2: 4 MB L3: 256 MB
IBM 9080-HEX	4C° – 40C°	4.6k VA	8% – 85%	3.5 GHz	L1: 48+32 KB L2: 2 MB L3: 120 MB

4 Configuration Settings

There are no specific configuration settings for this entropy source except that the external CPU timer shall be used and the internal timer shall be disabled. There are configuration settings defined in jitterentropy.h, however, any changes to them would invalidate the entropy validation certificate:

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 */

#define JENT_MEMORY_BLOCKS 64

#define JENT_MEMORY_BLOCKSIZE 32

#define JENT_MEMORY_ACCESSLOOPS 128

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_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. The entropy gathering logic creates 64 bits per invocation. This interface corresponds to the GetEntropy() conceptual interface from SP800-90B.

- jent_lfsr_time(): 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 56.22 bits of entropy per each 64-bit output, that is an entropy rate of 0.8785 bits/bit.

After the entropy source output is obtained by the consuming application (e.g. cryptographic module), additional entropy can be obtained by the concatenation of each 64-bit output. The entropy source does not truncate or partition the data before it is returned to the caller.

A module that utilizes this entropy source shall consider the amount of entropy available as described here and list such values in its Security Policy.

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 the LFSR. If any of these derivatives are zero, then the received time delta is considered stuck. In this case the LFSR is still updated, but the entropy value is not counted and the output of the LFSR is not used for insertion into the entropy pool. The stuck test then triggers the RCT for further processing. The second derivative is in fact the RCT itself.

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 (for the kernel space, the failure typically causes a kernel panic). 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 complete 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 kernel space cannot be reloaded without rebooting the entire kernel. Similarly, the data used for the on-demand health tests are discarded after successful completion.

The following error codes can be returned by jent_read_entropy():

-1 entropy_collector is NULL

-2 RCT failed

-3 APT test failed

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_lfsr_time() noise interface from the entropy source. The test harness and accessory kernel tools must be supplied by the vendor.

Raw noise data samples consisting of at least 1,000,000 bytes 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_lfsr_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.

If conditioned entropy data is to be tested, then the jent_read_entropy() interface shall be accessed instead of jent_lfsr_time() interface.