SP 800-90B Non-Proprietary Public Use Document

Senetas CPU Jitter Entropy Source

Firmware Version: 3.4.1

Senetas Security
312 Kings Way
South Melbourne, Victoria 3205

Document Version 1.5
October 27, 2023

Revision History

Version	Change
1.0	Initial draft
1.1	Second draft
1.2	First Version for CMVP submission.
1.3	Revised for CCOM1 comments.
1.4	Revised for additional CMVP and vendor comments.
1.5	Added Intel Xeon D-2145NT (Skylake) processor.

Table of Contents

1 Description

CPU Jitter Entropy Source is a non-physical entropy source and makes no IID claim and thus meets all the requirements for non-IID compliance. The report is for CPU Jitter version 3.4.1. Table 1 summarizes the Senetas platforms and their respective processors for which testing was performed.

2 Security Boundary

The boundary of the CPU Jitter implementation is shown in Figure 1. CPU Jitter gets its entropy from the time deltas between repeated hash/memory collection operations. The CPU Jitter output is used seed an SP 800-90A compliant DRBG (which is outside of the boundary of the entropy source). The security boundary of the CPU Jitter implementation contains the source code files provided as part of the software delivery. The source code contains API calls which are used by modules requesting entropy from CPU Jitter.

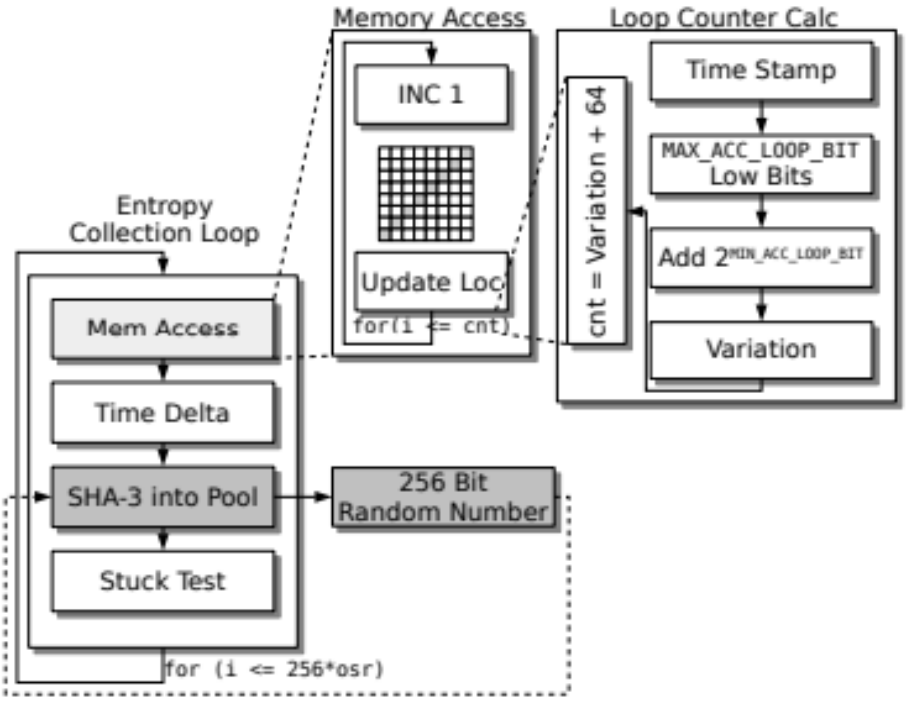

Figure 1 - Entropy Source Diagram

3 Operating Environments and Conditions

Table 1 summarizes the operating conditions for each of the tested platforms.

Platform	Processor	Clock Speed	Cache	Platform Temperature	Platform Voltage (AC)
CN6100	Intel ATOM Z530	2.00GHz	L1 = 24 KiB L2 = 512 KiB L3 = N/A	0°C – 40°C	90V – 264V
Dell VEP4600	Intel Xeon D-2145NT (Skylake)	1.90 GHz	L1 = 32 KiB L2 = 1 MiB L3 =1.375 MiB	0°C – 90°C	90V – 264V

Table 1 - Operating Environment and Conditions

4 Configuration Settings

Table 3 summarizes the configuration settings used by the Senetas implementation of CPU Jitter. These settings must be preserved to comply with the CPU Jitter ESV certificate.

Parameter	Value	Description
JENT_RANDOM_MEMACCESS	Enabled	Enables random memory access using bit versus blocks
JENT_MEMORY_BITS	17	Number of bits in a memory block (i.e., the memory block size in bits is 2^JENT_MEMORY_BITS)
JENT_MEMORY_SIZE	128 KiB	Memory block size
JENT_MEMORY_ACCESSLOOPS	128	Memory access loops
JENT_MIN_OSR	3	Set in code by JENT_CONF_DISABLE_LOOP_SHUFFLE (Enabled: 3 (default), Disabled: 1)
JENT_CONF_DISABLE_LOOP_SHUFFLE	Enabled	Disables pseudo-random looping e.g., lower boundary entropy equals upper boundary entropy.

Table 2 - Configuration Settings

5 Physical Security Mechanisms

The Senetas CPU Jitter Entropy Source is enclosed entirely within the module's cryptographic boundary which is protected by tamper evident seals, anti-probing barriers and electronic tamper detection and is assessed for FIPS level 3 Physical Security compliance.

6 Min-Entropy Rate

The Senetas CPU Jitter implementation generates an output that is considered to have full entropy. A request for 256 bits of entropy results in 256 bits of entropy per output sample, or full entropy.

7 Health Tests

Per the SP 800-90B requirements, health tests are run when CPU Jitter starts, and are then run continuously while it is operating. All tests check for persistent failures based on their respective "cutoff" values, which represent expected error thresholds.

CPU Jitter implements four types of health tests:

- Stuck Test
- Repetition Count Test (RCT)
- Adaptive Proportion Test (APT)
- Lag Predictor Test

All health test failures are considered permanent failures. If one is triggered, the current instance of the CPU Jitter Entropy Source will always remain in error state. The documentation of the API call jent_read_entropy (3) explains that the caller can only clear this error state by deallocating the CPU Jitter Entropy Source instance followed by an allocation of a new CPU Jitter Entropy Source instance to reset the noise source.

8 Maintenance

There are no maintenance requirements for the CPU Jitter library.

9 Required Testing

The entropy report contains the results of the testing Intel did on their raw and restart data. Raw data was collected by running the invoke_testing.sh script included in the CPU Jitter package. This script gathers 1,000,000 samples of 8-bits of raw time stamp data. It also gathers 1000 samples of 8-bits of raw data after restarting CPU Jitter each time, thus eventually gathering 1,000,000 samples of raw data for the restart tests.

Raw Data Analysis

Each raw data sample consists of one time stamp, which is 64 bits long. The CPU Jitter design states that the source can deliver full entropy if and only if the min-entropy is at least 1/osr (osr=3) bits of entropy per time stamp. This CPU Jitter Entropy Source implementation uses an oversampling rate of 3.

Restart Data Analysis

The CPU Jitter Entropy does not have a stochastic model, but an $H_{submitter}$ estimate of 1 was determined using analysis as described in SP 800-90B Section 3.2.2, Requirement 3. The CPU Jitter Entropy Source implementation uses an oversampling rate of 3. For that reason, the initial entropy estimate is set to 0.333 (1/3) for the purpose of running the NIST restart tests.

For the restart tests, the raw entropy data is collected for 1,000 CPU Jitter Entropy Source instances allocated sequentially. That means, for one collection of raw entropy, one CPU Jitter Entropy Source instance is allocated. After the conclusion of the data gathering it is deallocated and a new CPU Jitter Entropy Source instance is allocated for the next restart test round.