DINAMO Cyber Security Company

DINAMO Hardware Security Module CPU Jitter

6.9.0.0

SP 800-90B Non-Proprietary Public Use Document

Document Version: 0.5

Prepared for:

DINAMO
Cyber Security Company

DINAMO Cyber Security Company
United Nations Avenue, 14401
ED. TARUMA, Sao Paulo
Brazil

Phone: +55 11 3304 3120
www.dinamonetworks.com

Prepared by:

Corsec

Corsec Security, Inc.
12600 Fair Lakes Circle
Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
www.corsec.com

Revision History

Version	Modification Date	Modified By	Description of Changes
0.1	2026-01-28	Austin Crocker	Initial draft.
0.2	2026-04-06	Austin Crocker	Addressed Lab Comments: • Clarified section 10
0.3	2026-06-30	Austin Crocker	Addressed Lab Comments: • Changed use of internal timer in Table 1 • Added JENT_CACHE_SHIFT_BITS parameter to Table 2 • Changed JENT_DISABLE_INTERNAL_TIMER flag in Table 3 to Used
0.4	2026-07-28	Austin Crocker	Addressed Lab Comments: • Changed entropy source name to align with CAVP certificate Addressed DINAMO Comments: • Updated the company name Additional Changes: • Updated FIPS IG Reference date
0.5	2026-09-09	Austin Crocker	Addressed CMVP comments.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

Table of Contents

1. Description 4
2. Security Boundary 5
3. Operating Conditions 6
4. Configuration Settings 7
5. Physical Security Mechanisms 9
6. Conceptual Interfaces 10
7. Min-Entropy Rate 11
8. Health Tests 12
9. Maintenance 13
10. User Verification 14

List of Tables

Table 1 – Operating Conditions 6
Table 2 – Configuration Settings 7
Table 3 – CPU Jitter Configuration Flags (for jent_entropy_init_ex and jent_entropy_collector_alloc) 7
Table 4 – Entropy Source Interfaces 10

List of Figures

Figure 1 – Entropy Collection Operation 5

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

1. Description

The DINAMO Hardware Security Module CPU Jitter is a non-physical (NP) entropy source that meets the standards of NIST SP 800-90B (version January 2018) and FIPS 140-3 IGs D.J, D.K, and D.O (version August 19, 2026). This report is for DINAMO Hardware Security Module CPU Jitter version 6.9.0.0 based on the user-space jitterentropy-library v3.6.2.

The noise generation of this entropy source is based on the tiny, unpredictable variations in the execution time of a given set of instructions (this set of instructions is separated into a SHA3-256 component and a memory access component). The execution time 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 measurement of the jitter is performed over the execution time of the SHA3-256 and the execution time of memory access, after which the variances are collected and injected into the SHA3-maintained entropy pool.

The entropy source was tested on the operating environments listed in Table 1. It makes no IID claim and thus meets all the requirements for non-IID compliance.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

2. Security Boundary

The security boundary of the entropy source is the compiled binary of the entropy source of the C code that implements it and any memory allocated to it by the underlying operational environment. This entropy source provides API calls which are to be used by entities using the DINAMO Hardware Security Module CPU Jitter.

Figure 1 below illustrates the entropy collection process.

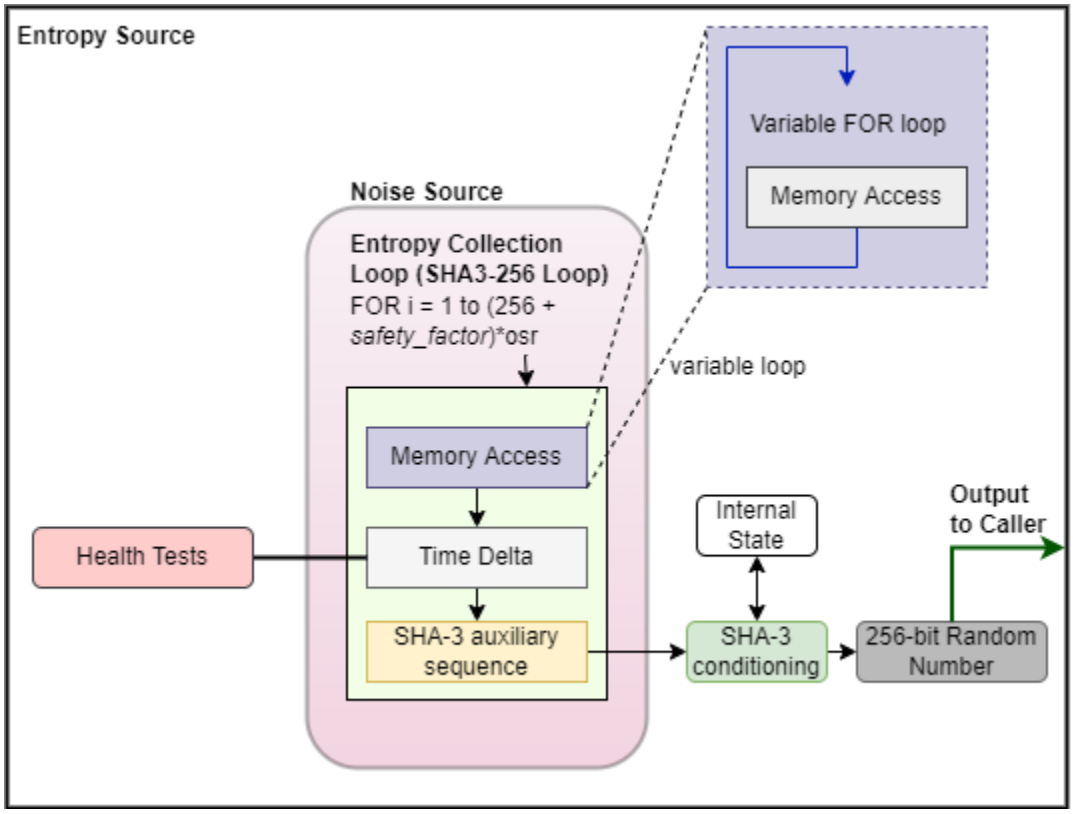


Figure 1 – Entropy Collection Operation

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

3. Operating Conditions

Table 1 summarizes the operating conditions for each of the tested operational environments (OEs).

Table 1 – Operating Conditions

Identifier	Operating System	Processor	Software	Operating Conditions
OE 1	Oracle Linux 9.5	Intel Xeon Silver 4210 (Cascade Lake) • CPU Clock Speed: 2.20 GHz • L1i Cache: 32KB x 10 • L1d Cache:32 KB x 10 • L2 Cache: 1 MB x 20 • L3 Cache: 13.75 MB x 2	• Health Testing reporting and use of safety factor: fips_enabled • safety_factor = 64 • Oversampling rate: osr = 3 • Use of internal timer: Disabled • Optimizer flag: -O0	• Temperature range: 10°C to 85°C • Core Voltage (Voltage range): 0.6V to 1.1V

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

4. Configuration Settings

Table 2 summarizes the configuration settings used by the DINAMO Hardware Security Module CPU Jitter. These settings must be preserved to comply with the ESV certificate.

Table 2 – Configuration Settings

Parameter	Default Value	Location
JENT_CONF_DISABLE_LOOP_SHUFFLE	Defined	jitterentropy.h
JENT_HEALTH_LAG_PREDICTOR	Defined	jitterentropy.h
JENT_RANDOM_MEMACCESS	Defined	jitterentropy.h
ENTROPY_SAFETY_FACTOR	64	jitterentropy.h
JENT_MIN_OSR	3	jitterentropy.h
JENT_MEMORY_SIZE	2^{17}	jitterentropy.h
JENT_MEMORY_BITS	17	jitterentropy.h
JENT_MEMORY_BLOCKS	512	jitterentropy.h
JENT_MEMORY_BLOCKSIZE	128	jitterentropy.h
JENT_MEMORY_ACCESSLOOPS	128	jitterentropy.h
JENT_CONF_ENABLE_INTERNAL_TIMER	Defined	Makefile
JENT_APT_WINDOW_SIZE	512	jitterentropy.h
JENT_LAG_WINDOW_SIZE	2^{17}	jitterentropy.h
JENT_LAG_HISTORY_SIZE	8	jitterentropy.h
JENT_POWERUP_TESTLOOPCOUNT	1024	jitterentropy-base.c
CLEARCACHE	100	jitterentropy-base.c
MIN_HASH_LOOP	0	jitterentropy-noise.c
MIN_ACC_LOOP_BIT	0	jitterentropy-noise.c
JENT_CACHE_SHIFT_BITS	0	jitterentropy.h

Table 3 – CPU Jitter Configuration Flags (for jent_entropy_init_ex and jent_entropy_collector_alloc)

Parameter	Status	Argument Name	Meaning
JENT_DISABLE_MEMORY_ACCESS	Not Used	flags	Disable the memory access timing.
JENT_FORCE_INTERNAL_TIMER	Not Used	flags	Force use of pthreads-based internal timer, rather than the underlying hardware timer.
JENT_DISABLE_INTERNAL_TIMER	Used	flags	Prevent the use of the pthreads-based internal timer.
JENT_FORCE_FIPS	Used	flags	Force fips_enabled to be set (enabling health test reporting and use of the safety factor).
JENT_MAX_MEMSIZE_*	Not Used	flags	Allows for run-time selection of the memory region size.
Osr	0	osr	Sets the internal osr rate to max(osr, JENT_MIN_OSR).

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

5. Physical Security Mechanisms

The DINAMO Hardware Security Module CPU Jitter is a software-based noise source. As such, the DINAMO Hardware Security Module CPU Jitter may be used on a variety of devices with differing security levels. The physical security mechanisms will be provided by the module security policy.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

6. Conceptual Interfaces

The entropy source is depicted in Figure 1. At a high level, the entropy source is initialized by running the `jent_entropy_init_ex` function. This function invokes lower-level functions that trigger the required power-on health tests. After the library has been initialized, new instances can be allocated using the `jent_entropy_collector_alloc` function. This function allocates the memory necessary for an instance, establishes the value of osr for this instance, and forces certain behaviors required in this validation. The Jitter RNG instance can then be used to request entropy using the `jent_read_entropy` function.

The interface inputs and outputs are as follows:

Table 4 – Entropy Source Interfaces

SP 800-90B Interface	API	Inputs / Outputs
GetEntropy	jent_read_entropy()	• Inputs: ○ struct rand_data *ec ○ char *data. ○ size_t len • Outputs: ○ Non-negative integer number of bytes written to the buffer, or ○ A negative integer value to indicate that an error occurred.
GetNoise	jent_entropy_collector_alloc()	• Inputs: ○ Integer osr ○ Integer flags • Inputs: ○ A (non-NULL) pointer to the entropy source instance state, or ○ A NULL pointer to indicate that the entropy source allocation has failed.
HealthTest	jent_entropy_init_ex()	• Inputs: ○ Integer osr ○ Integer flags • Outputs: ○ The integer 0 when initialization is successful and start-up health tests have passed, or ○ A negative value to indicate that initialization has failed.

Additional Instructions: The flags and osr parameters must be set as specified in Table 3 for the relevant OE. The caller must test the return status for every function invocation. Use of the `jent_entropy_init()` and `jent_read_entropy_safe()` functions is not allowed within this validation.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

7. Min-Entropy Rate

The DINAMO Hardware Security Module CPU Jitter implementation generates an output that considered to have full entropy per IG D.K Resolution 19. The entropy source uses SHA3-256, which is a vetted conditioning component providing 256 bits of entropy per request of 256 bits, or full entropy.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

8. Health Tests

The DINAMO Hardware Security Module CPU Jitter implements the following health tests:

- Stuck Test
- Repetition Count Test
- Adaptive Proportion Test
- Lag Predictor Test

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.

On-demand health tests are performed by deallocating the current Jitter RNG instance and reallocating a new instance. The startup health tests are then re-executed and the samples used for those tests are discarded.

The following error codes can occur:

- -1 Entropy_collector is NULL – the entropy source will need to be reinstantiated.
- -2 RCT test fails – the entropy source will need to be reinstantiated.
- -3 APT test fails – the entropy source will need to be reinstantiated.
- -4 The timer cannot be initialized – the entropy source will need to be reinstantiated.
- -5 Lag predictor test fails – the entropy source will need to be reinstantiated.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

9. Maintenance

There are no maintenance requirements for the DINAMO Hardware Security Module CPU Jitter entropy source.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

10. User Verification

Since the entropy source is embedded in firmware, the user can retrieve the firmware version using the DINAMO API using the following steps:

1. From the **Main Screen**, click **Device Management**.
2. Click **HSM Information**.
3. The operator will see the following:

Firmware:	6.9.0.0
Hardware Profile:	6.055.3.06.RevX.U

The raw noise data was collected from the tested operational environments listed in Table 1 at their normal operating conditions using the invoke_testing.sh script, written by Stephan Müller in the jitterentropy library - recording userspace github. This script first gathers 1,000,000 consecutive samples of raw data. It then gathers 1,000,000 samples of raw data in groups of 1,000 samples and restarts the DINAMO Hardware Security Module CPU Jitter instance after each group of data for the restart tests, following the restart procedure specified in SP 800-90B. Then, all collected samples from both the raw noise data and the restart data were mapped down to their eight least significant bits using the extractbits tool in the Theseus distribution using a bitmask of 0x000000FF.

The raw noise data can be collected or user verification from the operational environment listed in Table 1 at its normal operating conditions using the invoke_testing.sh script, written by Stephan Müller in [Jitter Library].

This script makes and invokes the jitterentropy-hashtime.c script to

1. First gather 1,000,000 consecutive samples of raw data using the command below:

```
jitterentropy-hashtime 1000000 1 $ ../results-measurements/jent-raw-noise
```

2. It then gathers 1,000,000 samples of raw data in groups of 1,000 samples, restarting noise source (following the restart procedure specified in SP 800-90B) after each 1,000-sample data group using the command below:

```
jitterentropy-hashtime 1000 1000 ../results-measurements/jent-raw-noise-restart
```

Then, all collected samples from both the raw noise data and the restart data were mapped down to their highest entropy producing bits using a bitmask of 0x000000FF; the `extractbits` tool in the Theseus distribution was the tool used in our reports.

The raw noise data was then processed by the SP 800-90B tool to obtain a minimum entropy rate which must provide enough entropy to the conditioner to output full entropy as stated in section 7.

The restart noise data was processed by the SP 800-90B tool, where the following conditions must be met:

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.

- The sanity test applied to the noise restart data must pass.
- The minimum of the row-wise and column-wise entropy rate shall not be less than half of $H_I = \min\left(\frac{1}{osr}, H_{original}, 8 * H_{bitstring}\right)$ for the OE and its osr defined in Table 1.

©2026 DINAMO Cyber Security Company
This document may be freely reproduced and distributed whole and intact including this copyright notice.