SP 800-90B Non-Proprietary Public Use Document

Document Version: 1.1

Date: January 7, 2023

Entropy Source: JEnt-MemOnly

KeyPair Consulting Inc.
987 Osos Street
San Luis Obispo, CA 93401
United States

keypair.us
info@keypair.us
+1 (805) 316-5024

References

Ref	Title	Date	Author
[90B]	NIST SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation	10-Jan-2018	NIST CT
[140IG]	Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program	7-Oct-2022	CMVP
[ESVMM]	Entropy Source Validation, program Management Manual (under development)	TBD	ESV
[Hill 2022]	*JEnt (MemOnly) and JEnt Analysis Approaches*. CMUF Entropy Working Group.	18-Oct-2022	Joshua E. Hill
[JEnt-MemOnly]	*Jitterentropy library* with MemOnly updates. https://github.com/joshuaehill/jitterentropy-library/tree/MemOnly	21-Nov-2022	Various
[Müller 2022]	CPU Time Jitter Based Non-Physical True Random Number Generator	01-Jul-2022	S.Müller Chronox.de

Table of Contents

1 Description

This Public Use Document accompanies the KeyPair Consulting entropy source validation of the MemOnly branch of the open-source jitterentropy-library, hereafter referred to as "JEnt-MemOnly". JEnt-MemOnly is a non-physical entropy source intended to generate entropy input for instantiation and reseed of an SP 800-90A compliant DRBG.

To incorporate the JEnt-MemOnly validation into a FIPS 140-3 module submission, please contact KeyPair Consulting.

The JEnt-MemOnly design is depicted in Figure 1, with design documented in [Müller 2022, Section 3]. A summary of MemOnly branch code changes and an accompanying analysis process is provided in [Hill 2022].

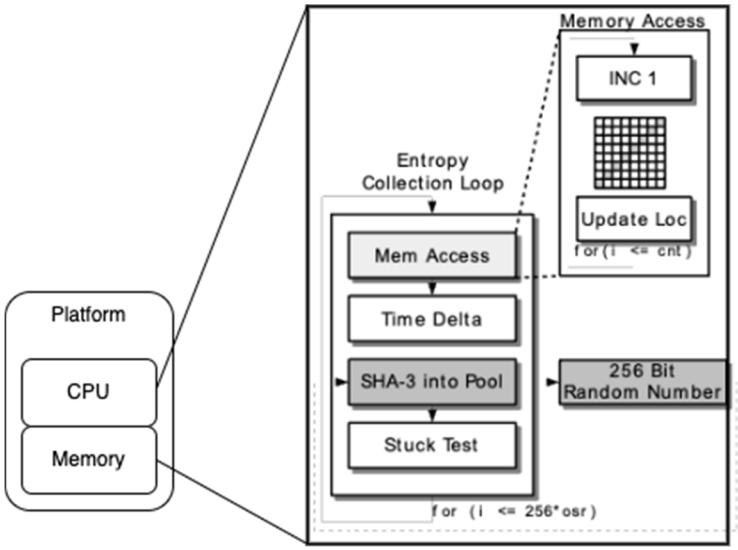

Figure 1: Entropy Source (Adapted from [Müller 2022, Figure 3.1])

The JEnt-MemOnly design includes elements that map to the conceptual components contained within an SP 800-90B entropy source:

- The Memory Access (MemAccess) primary noise source (memory timing).
- An additional noise source (Overall Timing).
- Health tests (depicted as the "Stuck Test" block in Figure 1).
- A conditioning algorithm (SHA-3).

In the original jitterentropy-library design, noise from both the Overall Timing source as well as the MemAccess noise source contributes to the estimation of entropy at runtime and in the assessment process. The MemOnly branch of design used in JEnt-MemOnly provided modifications to:

- Use the timing of the memory MemAccess as the sole source of entropy that is credited in the analysis. The Overall Timing is now treated as an additional noise source but is not credited in the analysis.
- Limit the primary noise source output (and thus the entropy estimation) to a selected sub-distribution.
- Provide configurable decimation (discarding) of samples credited in the analysis and health testing – see [Hill 2022] for additional detail.
- Remove unused and deprecated code.

The use of JEnt-MemOnly is accompanied by a process to determine a defensible entropy estimate along with the corresponding runtime parameters. Briefly, the process:

- Identifies an allocated memory size (per platform) required to force sufficient proportion of cache misses.
- Identifies the sub-distribution (minimum and maximum time delta) associated with memory I/O.
- Applies the *Essentially IID* analysis approach to non-zero JENT_MEMORY_DEPTH_EXP configurations.
  or
- Applies the *Selected Sub-distribution Empirical* analysis approach for JENT_MEMORY_DEPTH_EXP=0.

This Public Use Document is constructed to support validation of multiple platforms within a general architecture. While the performance of JEnt-MemOnly is highly variable across variations in CPU and memory configuration, the MemOnly branch implementation and the selected sub-distribution entropy analysis approach permit a consistent Entropy Analysis Report and associated validation listing. The design of JEnt-MemOnly permits compile time configuration (here required to be consistent with the Operating Conditions and Configuration settings described below) that results in the entropy source providing full-entropy conditioned output. Consequently:

- Table 2 cites evaluated platforms.
- Table 3 provides tested operating conditions for each evaluated platform.
- Table 4 specifies platform specific configuration elements for each evaluated platform.

Table 1: Evaluated Entropy Source Specification

Identifier	Details
Entropy Source Name	JEnt-MemOnly
Software Version	[JEnt-MemOnly] tagged as v3.4.1-MemOnly-1.
Entropy Category	Non-physical (NP)
Entropy Estimation Track (per SP 800-90B §3.1.2)	Non-IID

Table 2: Evaluated Platforms

Evaluated Platform(s)	CPU	RAM size	RAM configuration
JMO-EP1	Intel Xeon 6252, Stepping 7 (2.1 GHz – 3.7 GHz clock)	384 GB	6 x Crucial CT64G4LFQ4266[1]

The entropy assessment findings and the validation described by this Public Use Document are specific to the particular part/version and entropy-relevant parameters given in the tables above and do not apply more broadly. If the entropy-relevant parameters are different than those described in this report, then additional modeling and statistical testing are required. Any part/version or configuration other than that described above is not covered by this validation.

2 Security Boundary

The JEnt-MemOnly entropy source in the context of a software or firmware solution is depicted in Figure 1.

3 Operating Conditions

The testing and operation of the JEnt-MemOnly entropy source is valid across full range of operations for the operational environment, as the mechanism of entropy is not expected to vary across conditions.

[1] Note this part has fixed timing, so the memory size, voltage, buffering, internal clock rate, prefetch delay, and memory timings are all fixed through use of this part.

4 Configuration Settings

Entropy relevant parameters are listed in Table 3 (parameters common across platforms) and Table 4 (parameters specific to a platform and an associated operating and analysis configuration).

Table 3: Entropy-Relevant Parameters (common across validated platforms)

Parameter	Meaning	Type	Value
JENT_CONF_ENABLE_INTERNAL_TIMER	Allow the pthread-based timer.	Macro	Not Defined
JENT_HEALTH_LAG_PREDICTOR	Use the Lag-predictor-based health test.	Macro	Defined
JENT_LAG_WINDOW_SIZE	Size of the Lag-predictor-based health test window.	Macro	131072
JENT_LAG_HISTORY_SIZE	Size of the Lag-predictor-based history.	Macro	8
JENT_APT_WINDOW_SIZE	APT Window size.	Macro	512
JENT_HASHLOOP_EXP	Exponent of number of hash loops (i.e., there is 1 hash loop per output).	Macro	0
JENT_MEMACCESSLOOP_EXP	Exponent of number of MemAccess loops (i.e., there is 1 MemAccess loop per output).	Macro	0
JENT_DIST_WINDOW	Size of the window for the Distribution Health Test.	Macro	10000
JENT_POWERUP_TESTLOOPCOUNT	A lower bound for the number of values to test on initialization (JENT_POWERUP_TESTLOOPCOUNT + CLEARCACHE are performed).	Macro	1024
CLEARCACHE	An additional number of values to perform to set branch prediction and caches.	Macro	100
ENTROPY_SAFETY_FACTOR	The number of extra bits of min entropy required to make a "full entropy" claim. Used only in FIPS mode.	Macro	64
fips_enabled	Flag that controls error mode reporting and use of the ENTROPY_SAFETY_FACTOR. Setting the force_fips flag for a JEnt-MemOnly instance forces this flag to be set to 1.	Variable	1
enable_notime	Use the constructed pthread timer.	Variable	0
Optimizer Setting	The compiler optimizer setting has a substantial impact on the resulting distribution.[2]	Compiler Flag	-O0

Table 4: Entropy-Relevant Parameters (platform specific)

Platform	DIST_MIN	DIST_MAX	MEM_SIZE	MEM_DEPTH	OSR
JMO-EP1 (configuration 1)	105	185	28	**0**	1
JMO-EP1 (configuration 2)	105	185	28	**11**	1

Platform specific parameters:

DIST_MIN (JENT_DISTRIBUTION_MIN macro): lower bound for the selected memory sub-distribution.
DIST_MAX (JENT_DISTRIBUTION_MAX macro): Upper bound for the selected memory sub-distribution.
MEM_SIZE (JENT_MEMORY_SIZE_EXP macro): Exponent of selected memory size (i.e., $2^{\text{MEM\_SIZE}}$ bytes will be used).

[2] The impact of the optimizer on this library is extensively discussed on the library's GitHub repository Issue #21.

MEM_DEPTH (`JENT_MEMORY_DEPTH_EXP` macro): Exponent of selected memory depth (used for decimation).
OSR (`osr` variable): Oversample Rate. The code presumes that the lower bound for the min entropy is 1/osr.

5 Physical Security Mechanisms

The JEnt-MemOnly entropy source operates within the physical protections of the associated processor.

The JEnt-MemOnly design may be incorporated into modules intended for CMVP validation of compliance to FIPS 140-3 and the related suite of security assurance standards and specifications. Any additional physical security protections are outside the scope of this entropy source design and implementation.

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` or `jent_entropy_init_ex` functions, which invoke lower-level functions that include 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 a JEnt instance, establishes the value of osr for this instance, and allows the user to request various behaviors by passing in the parameters listed in Table 2 and Table 3.

The JEnt-MemOnly instance can then be used to request entropy using the `jent_read_entropy` function. This function repeatedly produces blocks of 256 bits of conditioned data using the `jent_random_data` function until at least the amount of requested data has been produced.

The `jent_random_data` function first makes an initialization call to `jent_measure_jitter` to establish an initial value for the time, and then iteratively calls `jent_measure_jitter` in a loop until (256+ ENTROPY_SAFETY_FACTOR)*osr non-stuck delta noise source outputs have been input into the conditioner.

7 Min-Entropy Rate

There are interfaces that allow a user or cryptographic module developer to sample both the raw data and conditioned data for this entropy source.

When blocks of 1-million consecutive raw data samples are tested using the SP 800-90B non-IID tests, the results are expected to conform to an assessment distribution. Table 5 provides the observed assessment range for the assessed noise sources.

Table 5: Observed Raw Test Results (platform specific)

			Raw Data Assessment	
Platform	MEM_SIZE	MEM_DEPTH	Minimum	Maximum
JMO-EP1 (configuration 1)	28	0	3.28	3.48
JMO-EP1 (configuration 2)	28	11	3.52	3.56

In accordance with the Entropy Analysis Report, the conditioned output of this entropy source is full-entropy; that is, the analysis supports the claim of 256 bits of min entropy per 256-bit conditioned output block, provided the platform specific conditions and configuration settings described above are met.

These blocks are the output of a vetted conditioning function and can be subdivided further (as per SP 800-90B). If these blocks are subdivided, then every byte from the block can be treated as possessing at least 8 bits of min entropy, and the min entropy of any truncated sub-portion of the 256-bit output block is linearly scaled with the length of the retained sub-portion.

If this entropy source is used to seed a compliant DRBG, then the seeding requirements summarized in Table 6 must be met.

Table 6. Seeding Requirements for Security Strengths

DRBG Security Strength (bits)	Bytes Required (Nonce Provided)	256-bit Blocks Required (Nonce Provided)	Bytes Required (Random Nonce)	256-bit Blocks Required (Random Nonce)
112	14	1	21	1
128	16	1	24	1
192	24	1	36	2
256	32	1	48	2

8 Health Tests

All health tests are essentially continuous health tests, and are tested within the `jent_stuck` function. This function performs a modified version of the Repetition Count Test (RCT), an Adaptative Proportion Test (APT), a Lag Health Test, and a Distribution Health Test.

The Lag Health Test is performed by attempting to predict the current symbol using the prior 8 symbols. If the most successful lag (delay) becomes too successful in predicting the current output, or if the test is globally more successful than expected, then the Lag Health Test fails.

The Distribution Health Test operates on data prior to the selection of a sub-distribution or any decimation. If the proportion of pre-raw data that is output is too low, then the Distribution Health Test fails.

When the `fips_mode` flag is set, calls to the `jent_health_failure` function return with the current `health_failure` flag state. If `fips_mode` is not set, then this function always indicates that no failure has occurred. The `health_failure` flag indicates a persistent error for the JEnt instance used, and this flag cannot be reset. For an instance in FIPS mode, it is only possible to continue using the library for entropy production if a new JEnt instance is created.

The targeted cutoff parameters for the APT, RCT and Lag Predictor Health Test are dependent on the setting of osr. In the FIPS mode, only data that passes all health tests can be integrated into the conditioner.

All of the start-up tests (the full set of continuous health tests[3]) take the first 1024 consecutive samples. The samples used by the start-up health test are discarded. Some of the architectural health tests are run on the first 1124 samples.

On-demand testing is performed by allocating a new JEnt handle, which triggers the start-up tests. The samples used by the on-demand health test (effectively the start-up health test) are discarded.

There is no mechanism to clear an error state short of re-instantiating a new entropy source.

[3] Note that the Distribution Health Test requires a window size of at least 10,000 "pre-raw" samples to fill its test window after the platform's "delta GCD" is set. This amount of data is not available in start-up testing, so the Distribution Health Test cannot fail during start-up health testing.

9 Maintenance

The JEnt-MemOnly entropy source does not require maintenance.

10 Required Testing

If a user of this entropy source or cryptographic module developer wanted to verify that their instance of this source is working correctly, then they can use this procedure.

A raw data sample can be generated using the `jitterentropy-hashtime` tool in the `tests/raw-entropy/recording_userspace` directory.

The jitterentropy-hashtime program is compiled using the command:

```
CFLAGS="-DJENT_MEMORY_DEPTH_EXP=MEM_DEPTH -DJENT_MEMORY_SIZE_EXP=MEM_SIZE -DJENT_DISTRIBUTION_MIN=DIST_MIN
-DJENT_DISTRIBUTION_MAX=DIST_MAX" make -f Makefile.hashtime
```

Raw data can be extracted using the command:

```
./jitterentropy-hashtime 1000000 1 jent-sample
```

This creates the file `jent-sample-0001-sd.bin` which contains 8-bit raw samples that can be directly tested using the NIST tool using the command:

```
ea_non_iid -vv jent-sample-0001-sd.bin
```

If the resulting assessed entropy is greater than or equal to the “Minimum Assessment” and less than or equal to the “Maximum Assessment” values noted in Table 5, then this result is evidence that the entropy source is behaving in a way that is consistent with the assessed entropy source. Any assessed entropy greater than $\frac{1}{osr}$ bits of min entropy per symbol would support the entropy source’s “full entropy” claim for the output conditioned data.