AMD Random Number Generator

6/27/17

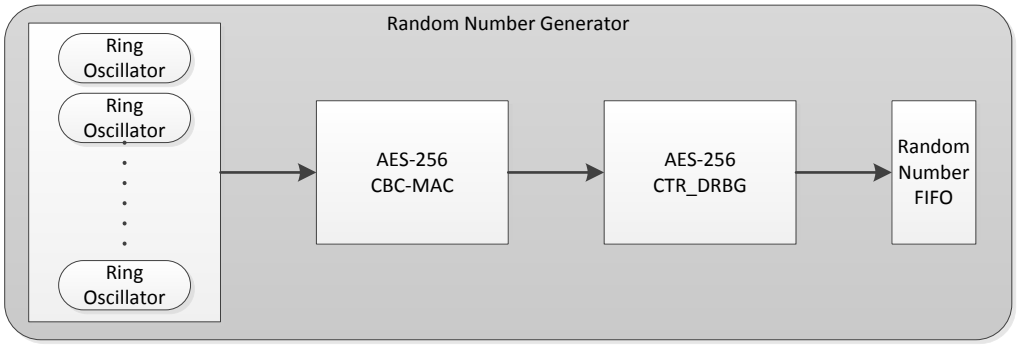
Introduction

This document describes the Random Number Generator (RNG) design used in the Cryptographic Co-Processor (CCP) 5.0 hardware included in the AMD RYZEN and EPYC processors. The RNG is designed to produce fast, high quality random numbers for use by the AMD Secure Processor as well as x86 software.

The RNG is designed to be compliant with the NIST SP 800-90 documents and uses constructs defined in those documents. Links to these documents can be found in the references section.

Architecture

The high-level RNG architecture is shown below:



There are 3 major components to the RNG, the noise source (ring oscillators), the entropy conditioner (CBC-MAC) and the deterministic random bit generator (DRBG).

Noise Source

The RNG uses 16 separate ring oscillator chains as a noise source. Each chain consists of a different prime number of inverters in a free-running configuration which capture natural clock jitter. The smallest chain consists of 3 inverters while the largest has 59. During each cycle of RNG operation, the 16 ring oscillators are sampled generating 16 bits of noise.

Entropy Conditioner

The 16-bits of ring oscillator noise is fed into the entropy conditioner which gathers multiple noise samples over time to use in generating the entropy needed by the RNG design. The RNG architecture is built assuming that each bit of noise output has a min-entropy of 0.5 bits, meaning that each 16-bit sample may have as little as 8 bits of entropy. This conservative estimation allows for some amount of bias to be present in the ring oscillators without affecting the secure operation of the rest of the RNG.

During operation, 512 total bits of noise samples are collected and fed into an AES-256 CBC-MAC construct as specified in NIST SP 800-90B section 6.4.2. According to NIST's guidance, this construct produces 128-bits of "full entropy" since the input string was considered to have 256 (2*128) bits of assessed entropy. This entire process is repeated 3 times to generate a 384-bit seed to be used by the CTR_DRBG.

After the initial 384-bit seed has been created, the entropy conditioner continues to run and accumulate ring oscillator samples into the 512-bit sample register with an XOR operation so new seed values can be ready when needed by the CTR_DRBG.

Entropy Health Checks

As stated earlier, the RNG architecture assumes that each bit of noise sample has a min-entropy of 0.5 bits. The hardware includes two hardware health checks that run continuously to ensure that this assumption stays valid and detect any anomalies in the noise source.

The **Repetition Count** health check detects if the same 16-bit noise sample is repeated across multiple sequential cycles. The test fails if this condition occurs for more than a specific number of cycles.

The **Adaptive Proportion** health check detects if the same 16-bit noise sample occurs multiple times within a fixed window of 4096 samples. The test fails if this happens more than a specific number of times in one window.

The specific cutoff values for these checks are configured by the AMD Secure Processor at runtime and are determined based on the assessed entropy rate (8 bits per 16-bit sample) as well as the acceptable false-positive probability (default is the NIST recommended value of 2^{-30}). For more details on these checks, see NIST SP 800-90B section 6.5.1.2.

If a health check does fail during operation, the RNG is designed to stop producing random values and an interrupt will be sent to the AMD Secure Processor for error handling.

DRBG

The RNG uses an AES-256 CTR_DRBG construct as specified in NIST SP 800-90A section 10.2.1 in order to produce fast, high quality random outputs. The DRBG is seeded using 384-bits of entropy from the entropy conditioner and produces random values which are subsequently stored in a FIFO buffer. This buffer enables fast burst reads of random numbers when needed.

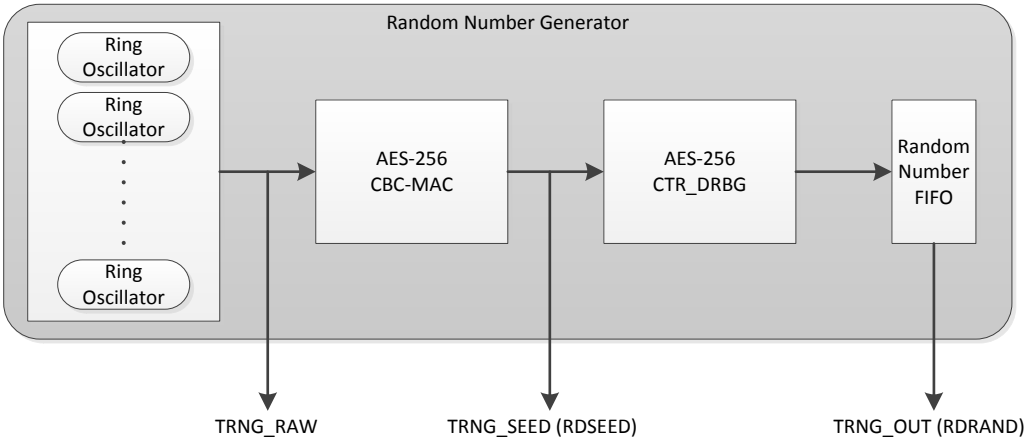
The DRBG is designed not to generate more than 2048 32-bit random values from an individual seed. Upon reaching this limit, the DRBG will stall until it is able to re-seed using new entropy from the entropy conditioner. The DRBG will also attempt to re-seed itself prior to this point if idle.

Software Visibility

The RNG includes 3 read-only output registers which provide visibility at different parts of the RNG flow. These registers are accessible through the AMD Secure Processor MMIO space for x86 software, and in some cases are also available through x86 user-level instructions:

MMIO Space Register	Description	Equivalent x86 instruction
TRNG_OUT	Contains a 32-bit random output value from the CTR_DRBG	RDRAND
TRNG_SEED	Contains 32 bits of conditioned entropy from the CBC-MAC	RDSEED
TRNG_RAW	Contains a 16 bit noise sample	n/a

The visibility points are also shown in the figure below:



Note that any values that are returned to software via the TRNG_RAW or TRNG_SEED registers are immediately discarded and not used in the rest of the RNG hardware. All 3 registers are built so a different value is returned automatically on every read.

The x86 RDRAND and RDSEED instructions can be used by user-level software to retrieve 16, 32, or 64-bits of high quality random number or conditioned entropy respectively. The TRNG_RAW register can be used by software wishing to perform more detailed statistical analysis of the ring oscillator output but users should be aware that by definition the raw noise sample outputs are not perfectly random.

Conclusion

The RNG design described here consists of three key parts, a ring oscillator based noise source, an AES CBC-MAC based entropy conditioner and an AES CTR_DRBG component for final output generation. The RNG is designed per the NIST SP 800-90 documents and it uses algorithms directly from these documents for the conditioner, DRBG, and entropy health checks. The result is a design that produces fast, high quality random numbers using well-established and standardized constructs.

References

NIST SP 800-90A: http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

NIST SP 800-90B (first draft): http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf

NIST SP 800-90C: http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf