Non-physical entropy sources

29 April 2021

Tim Hall (NIST)

Non-physical noise sources

Sec 2.2.1 Noise Source:

> "Noise sources can be divided into two categories: Physical noise sources use dedicated hardware to generate randomness; whereas *Non-physical noise sources* use system data (such as output of Application Programming Interface (API) functions, Random Access Memory (RAM) data or system time) or human input (e.g., mouse movements) to generate randomness."

Appendix B – Glossary has similar description for a Non-physical non-deterministic random bit generator.

Non-physical noise and entropy sources

- Sometimes called “software” entropy sources
- Also sometimes called “found” sources
  - Source not designed for the purpose of producing entropy
- Popular type are those in OS kernel RNGs

Day 1 unanswered questions on non-physical sources (all were upvoted)

- *Modeling non-physical entropy sources* would be very useful to hear about. When will there be time for that?

- [Meltem's talk] mentioned using stochastic model, especially for physical sources, *what about SW based solution with high resolution timers*?

- *SW solution* [requires] mandatory stochastic models and demonstrations? Any approved reviews on SW based solution for RNG? *Let's say high resolution timer* solution ?

Day 1 and Day 2 unanswered questions on non-physical sources

- Is there *more leeway in the required modeling assurance for non-physical sources*? Can a SW solution (e.g., interrupt timings) model actually be convincing enough?

- Can *SW based interrupt events be considered a single noise source* per exception called out in IG 7.19 #11? Concern is that it will be challenging

- For software-based sources, much of the underlying infrastructure is likely similar between many implementations. Is there some room to draw equivalence?

Other Day 1 and Day 2 mentions of non-physical sources

- “Nonphysical sources are typically way too complex to model well.”

- A few additional comments and questions on using a *heuristic approach* in justification of claimed entropy in non-physical (software) sources.
  - “Can we just describe them in English?”

Looks like there are some themes here…

- Consider software interrupts sampling a high resolution clock cycle counter as a noise source and discuss it

- Background
  - Pre-review program and reviewers in general
    - Role of reviewers and validation program

Software interrupts as non-physical noise source

- Consider the following noise source: software interrupts serviced by the OS
  - Each SW interrupt samples high-precision clock [timestamp counter, clock cycle counter]
  - Low-order bits of the timestamp are used as noise source samples

Software interrupts as non-physical noise source (2)

Some preliminaries:

- The high resolution clock cycle (timestamp) counter is not the noise source

- For the purposes of analyzing this noise source, the operation of the clock cycle counter should be consider as being *essentially deterministic*

Software interrupts as non-physical noise source (3)

- How do we produce an acceptable justification for this noise source?
- What is an acceptable heuristic?
- Where is unpredictability?
  - Expect any entropy from distribution of SW interrupts themselves?
  - Is it the interaction of the two that produces the entropy?

My other observations on non-physical entropy sources

- Boundaries of the noise source and entropy source were not clearly defined *in the reports*

- Generally, OS entropy sources had difficulty fitting neatly in the boundaries in SP 800-90B Figure 1
  - e.g., combining multiple noise sources and/or entropy sources as input to health tests
  - Even though they clearly *identified* the primary noise source and distinguished additional noise sources and entropy sources

My other observations on non-physical entropy sources

- Many (most?) have a dependency on the underlying platform
- Many of the *additional* noise sources and entropy sources rely on system (device), network and user events.
- Comments on how noise source functions in a *virtualized environment* very helpful
  - Not currently required

Questions?

Conditioning components in SP 800-90B

29 April 2021

Tim Hall (NIST)

Outline

- What is a conditioning component?
- Vetted conditioning components
- Non-vetted conditioning components
- Conditioning components that are bijective functions

What is a conditioning component?

SP 800-90B Sec. 2.2.2: "The *optional conditioning component* is a deterministic function responsible for reducing bias and/or increasing the entropy rate of the resulting output bits (if necessary to obtain a target value)."

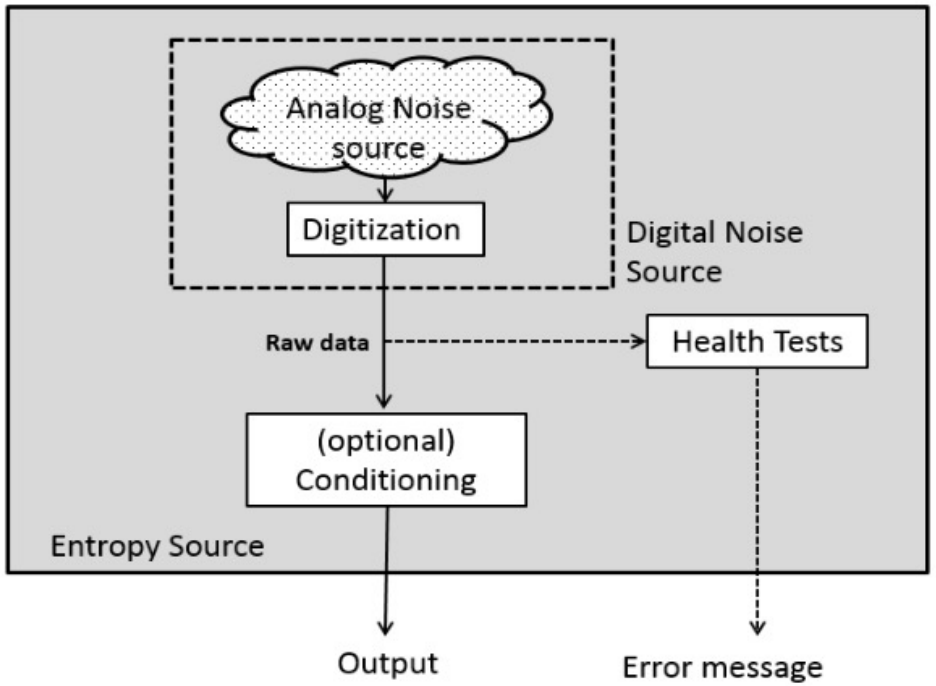

Figure 1 from SP 800-90B Sec. 2.2 "The Entropy Source Model"

What is a conditioning component?

Sec 3.1.5:

> "Noise source outputs are concatenated to construct n_{in}-bit input to the conditioning component. The entropy of the input, denoted h_{in}…is estimated to be w x h bits.
>
> Since the conditioning component is deterministic, the entropy of the output is at most h_{in}. However, the conditioning component may reduce the entropy of the output."

Vetted conditioning components

- Six defined in SP 800-90B
  - Three keyed functions:
    - HMAC (FIPS 198) with any approved hash function
    - CMAC (SP 800-38B) using AES block cipher
    - CBC-MAC using AES block cipher as defined in Appendix F
  - Three unkeyed functions
    - Any approved hash function in FIPS 180 or FIPS 202
    - Hash_df as specified in SP 800-90A using any approved hash function
    - Block_Cipher_df as specified in SP 800-90A

- Must have CAVP certificate

Vetted conditioning components

Sec. 3.1.6 Additional Noise Sources:

> "This Recommendation allows one to *concatenate the outputs of the additional noise sources to the primary noise source* to generate input to the conditioning component. *In such cases, vetted conditioning components* ***shall*** *be used*. No entropy is credited from the outputs of the additional noise sources."

Vetted conditioning components are these

Table 1 The narrowest internal width and output lengths of the vetted conditioning functions.

Conditioning Function	Narrowest Internal Width (nw)	Output Length (n_{out})
HMAC	hash-function output size	hash-function output size
CMAC	AES block size = 128	AES block size = 128
CBC-MAC	AES block size = 128	AES block size = 128
Hash Function	hash-function output size	hash-function output size
Hash_df	hash-function output size	hash-function output size
Block_Cipher_df	AES key size	AES key size

Non-vetted conditioning components are *everything else*

- Sec. 3.1.5.2 Using Non-vetted Conditioning Components

  "For non-vetted conditioning components, the entropy in the output depends on the entropy and size of the input (h*in* and n*in*), the size of the output (n*out*), and the size of the narrowest internal width (nw) and the *entropy of the conditioned sequential dataset* (as described in item 2 of Section 3.1.1), which *shall be computed using the methods described in either Section 6.1 (for IID data) or Section 6.2 (for non-IID data)*. Let the obtained entropy estimate per bit be h'.

  The output of the conditioning component (n_{out}) shall be treated as a binary string, for purposes of the entropy estimation."

Special case of non-vetted conditioning component

- One that implements a *bijective function*
  - FIPS 140-2 IG 7.19, Res. 9

"For Section 3.1.5, if the conditioning function can be shown to be bijective, then the vendor may claim that h_{out}=h_{in}"

Examples of conditioning components that is bijective?

Questions?