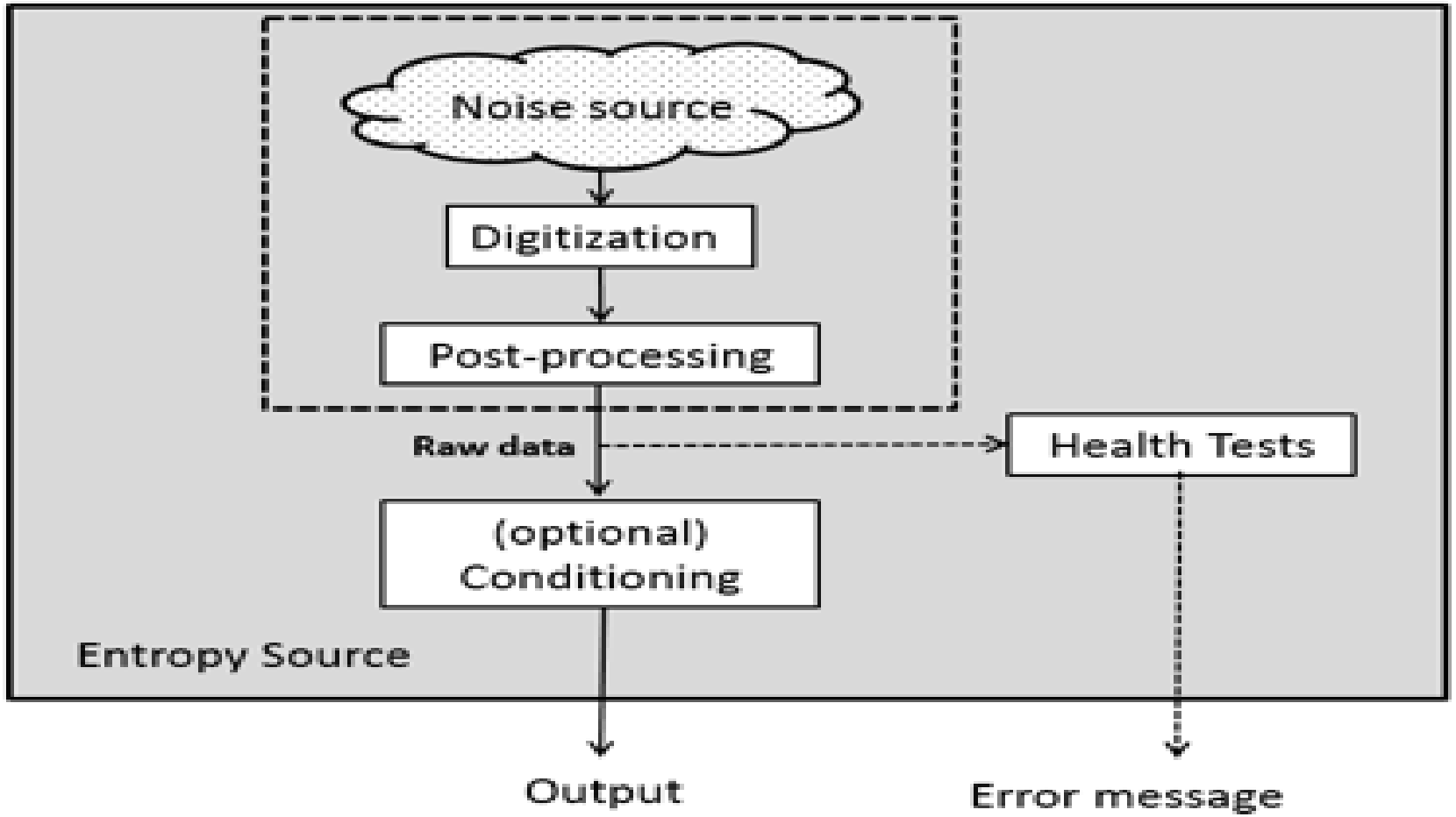
SP 800-90B Overview*

John Kelsey, NIST, May 2016



* Revised to correct some errors discovered after the presentation was given. Updated/new slides are starred.

Preliminaries

- SP 800-90 Series, 90B, and RBGs
- Entropy Sources
- Min-Entropy
- History of the Document

What's SP 800-90? What's 90B?

An RBG is used as a source of random bits for cryptographic or other purposes.

- The SP 800-90 series is about how to build *random bit generators*
- SP 800-90A describes deterministic mechanisms, called DRBGs
- **SP 800-90B describes entropy sources**
- SP 800-90C describes how to put them together to get RBGs

90B is about how to build, test, and validate entropy sources.

What is an Entropy Source?

- SP 800-90B is about how to build an *entropy source.*
- Entropy sources are used in SP 800-90 to provide seed material for DRBG mechanisms, among other things.
- An entropy source provides:
  - Bits or bitstings of fixed length
  - with a guaranteed minimum entropy per output
  - with continuous health testing to ensure the source keeps working

An entropy source is a black box that provides you entropy (unpredictable bits with a known amount of entropy) on demand.

So, What's Entropy Then?

- Entropy (specifically, min-entropy) is how we quantify unpredictability.
- If there's no way to guess X with more than a 2^{-h} probability of being right, then X has h bits of entropy.
- For a distribution with probabilities $p_1, p_2, \ldots, p_k$:

$$h = -\lg(\max(p_1, p_2, \ldots, p_k))$$

- Sometimes, we use the word "entropy" to mean unpredictability.
- Sometimes, we use it to mean the amount of unpredictability, in terms of min-entropy.

SP800-90B History

- Previous version published August 2012
- RNG workshop in 2012
- Snowden / Dual EC blowup
- Big changes:
  - Reworked the IID tests
  - Sanity checks → Additional estimates
  - Big improvements of tests of non-iid data
  - Updated and improved health tests
  - Restart Tests
  - Post-processing
  - Vetted conditioning functions moved here from 90C

Components of an Entropy Source

Noise Source

Where the entropy comes from

Post-processing

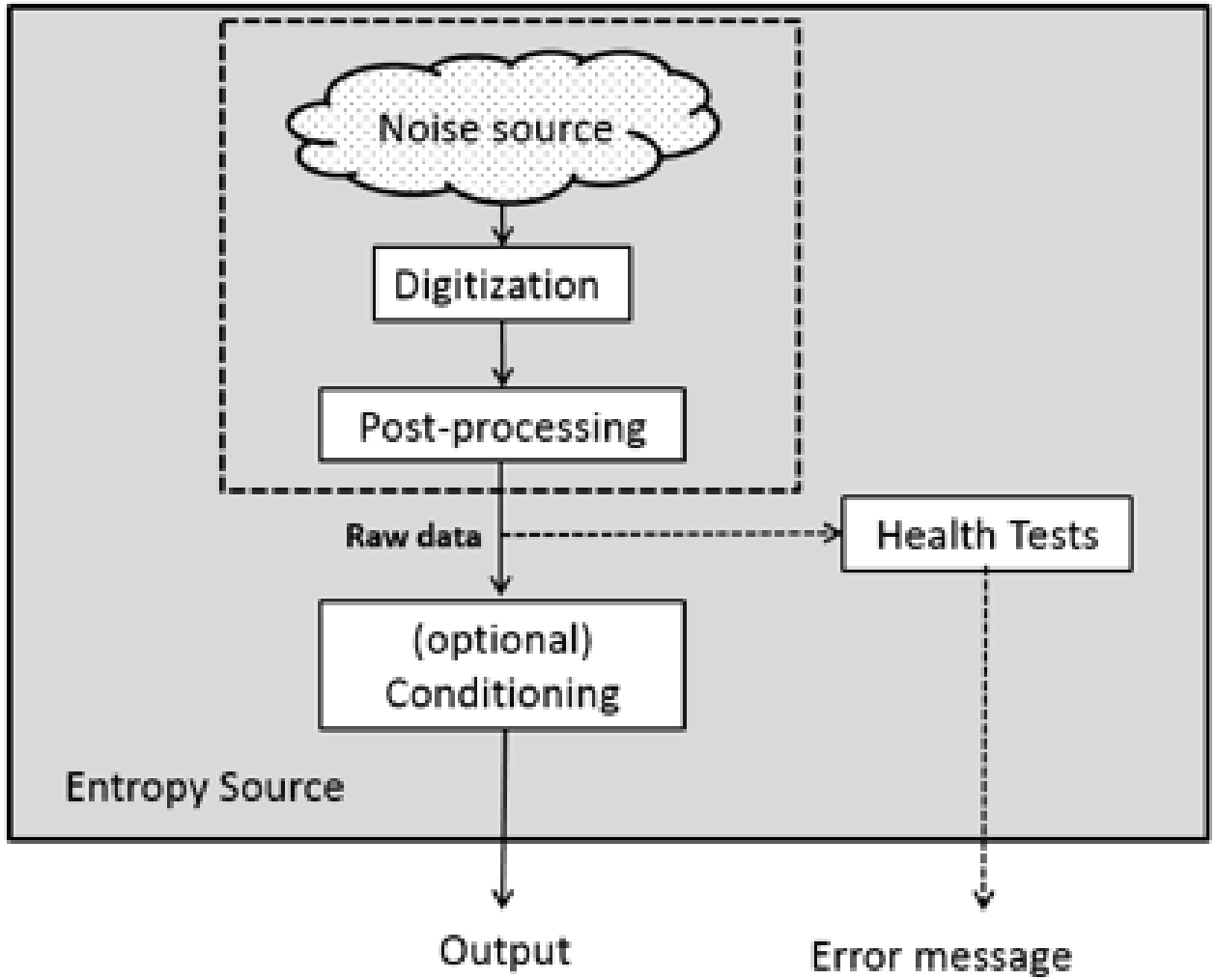
Optional minimal processing of noise source outputs before they are used.

Health tests

Verify the noise source is still working correctly

Conditioning

Optional processing of noise source outputs before output.



Noise Source

- A noise source provides bitstrings with some inherent unpredictability.
- Nondeterministic mechanism + sampling + digitization = noise source.
- **Every entropy source must have a noise source.**
- Common examples: ring oscillators, interrupt timings.

- The noise source is the core of an entropy source—it's the thing that actually provides the unpredictability

Post-Processing

- Post-processing is minimal processing done to noise-source outputs before they are used or tested.
- **Post-processing is optional**—not all entropy sources use it.
- There are a small set of allowed post-processing algorithms:
  - Von Neumann Unbiasing
  - Linear Masking
  - Run Counting

Health Tests

- Health tests attempt to detect major failures in the noise source.
- **They are REQUIRED** for all entropy sources.
- Health testing splits into:
  - Startup testing = one-time test done before any outputs are used
  - Continuous testing = testing done in the background during normal operation
- 90B defines two health tests
- Designers may use those, or provide their own tests and demonstrate that their tests detect the same failures.

Conditioning

- Conditioning processes the noise source outputs to increase their entropy/bit before they are output from the entropy source.
- **Conditioning is optional**—not all entropy sources incorporate a conditioning component.
- 90B specifies a set of six “vetted” conditioning functions based on cryptographic mechanisms.
- Vendors may also design or select their own conditioning functions.
- 90B describes the *entropy accounting* used for conditioning functions.

Components Wrapup

Noise Source

Where the entropy comes from

Post-processing

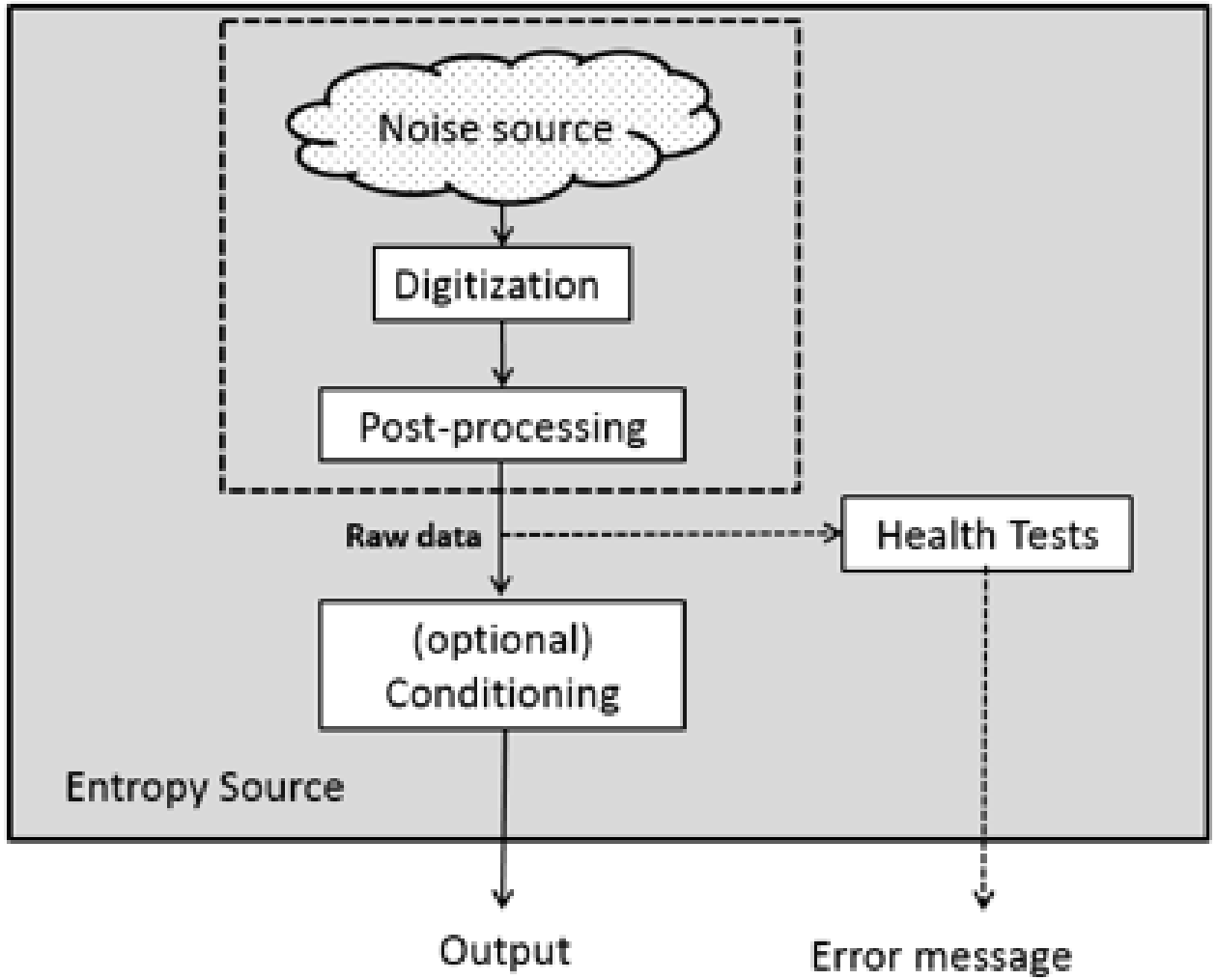
Optional minimal processing of noise source outputs before they are used.

Health tests

Verify the noise source is still working correctly

Conditioning

Optional processing of noise source outputs before output.



The Validation Process

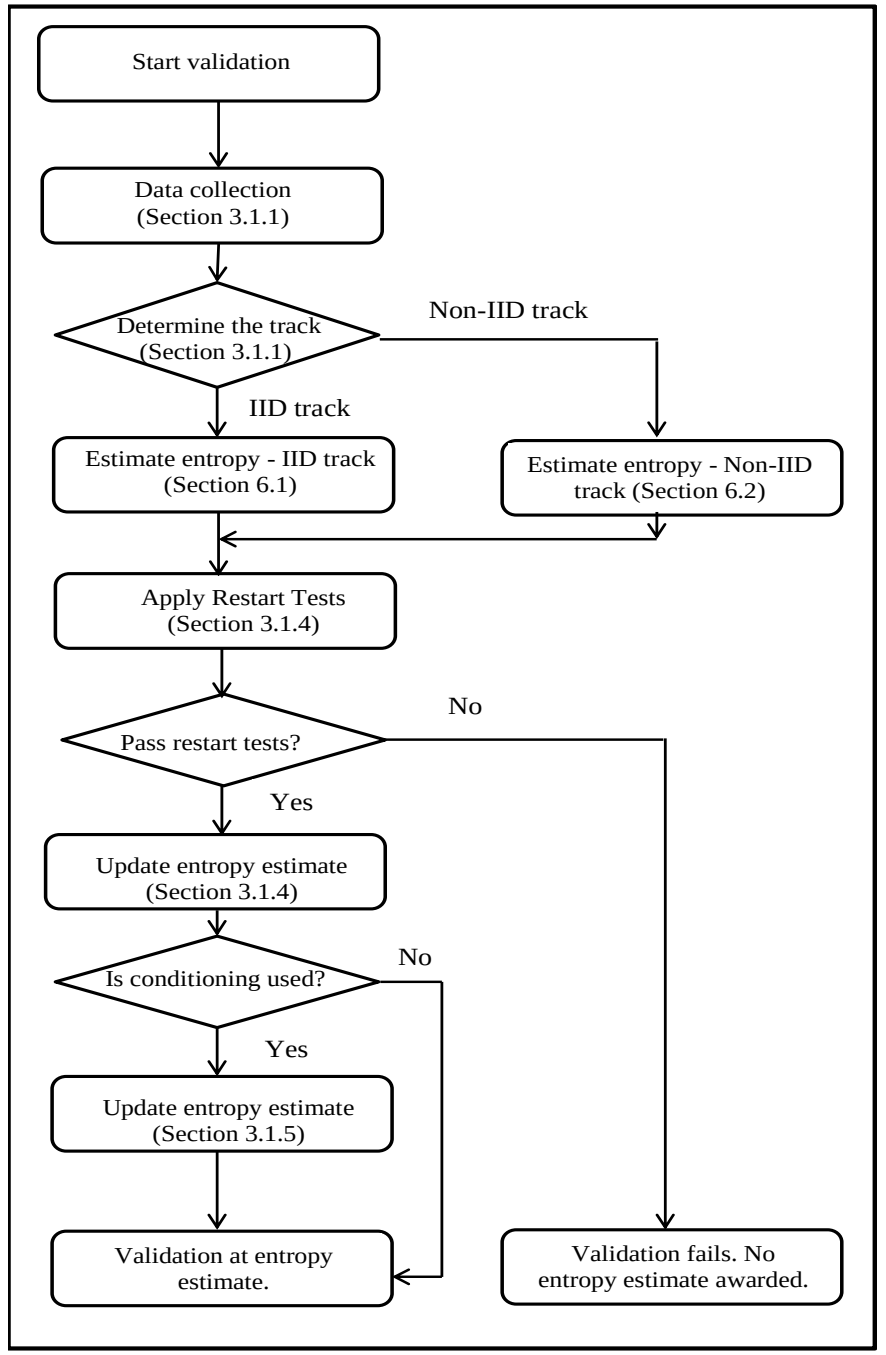
- Documentation
- Estimating entropy
- Design requirements

Documentation Requirements

- Full description of how the source works
- Justification for why noise source has entropy
- Entropy estimate and whether source might be iid
- Description of post-processing and conditioning, if used
- Description of alternative health tests, if used
- Description of how data collection was done

Entropy Estimation: Testing the Noise Source

- **Data collection**
- IID vs non-IID
- Entropy estimation
- Restart tests



Data Collection

- Entropy estimation requires lots of data from raw noise source, and possibly some data from conditioned outputs.
- Ideally, device has a defined way to get test access to raw noise source bits.
- If not, submitter may need to do data collection.
  - Submitter must justify why his method for data collection won't alter behavior of source.
  - Must also describe exactly how data collection was done.

What Data is Collected?

Two datasets from raw noise source samples—always required:

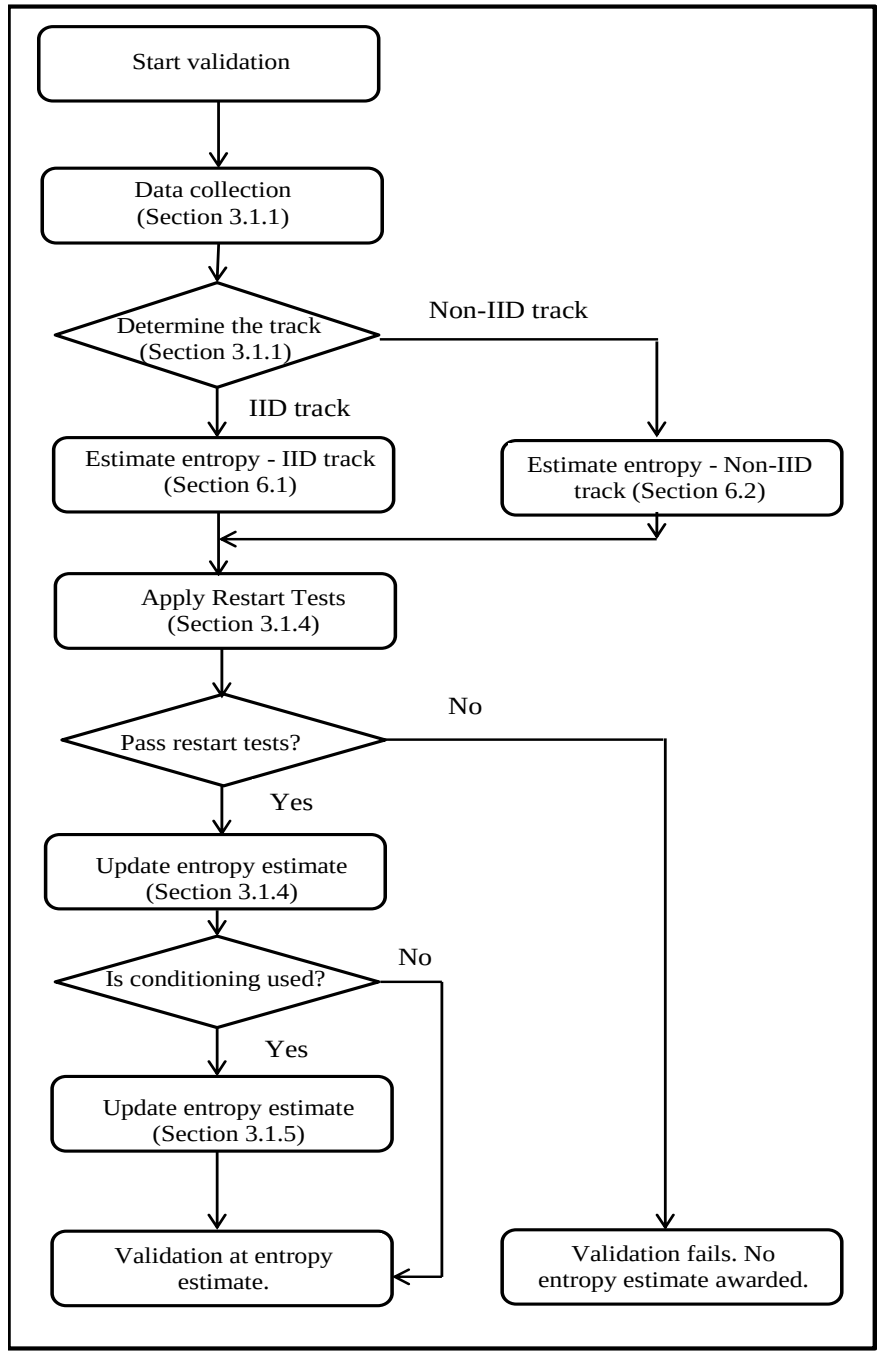
- Sequential dataset: 1,000,000 successive samples from noise source
- Restart dataset: 1,000 restarts, 1,000 samples from each restart
  - Restart = power cycle, hard reset, reboot

One dataset from conditioned outputs—sometimes required

- Conditioner dataset: 1,000,000 successive samples from conditioner
- *Only required if design uses a non-vetted conditioning function.*

Entropy Estimation: Testing the Noise Source

- Data collection from noise source (optionally with post-processing)
- **IID vs non-IID**
- Entropy estimation
- Restart tests



Determining the Track (IID or Non-IID)

IID = Independent and Identically Distributed

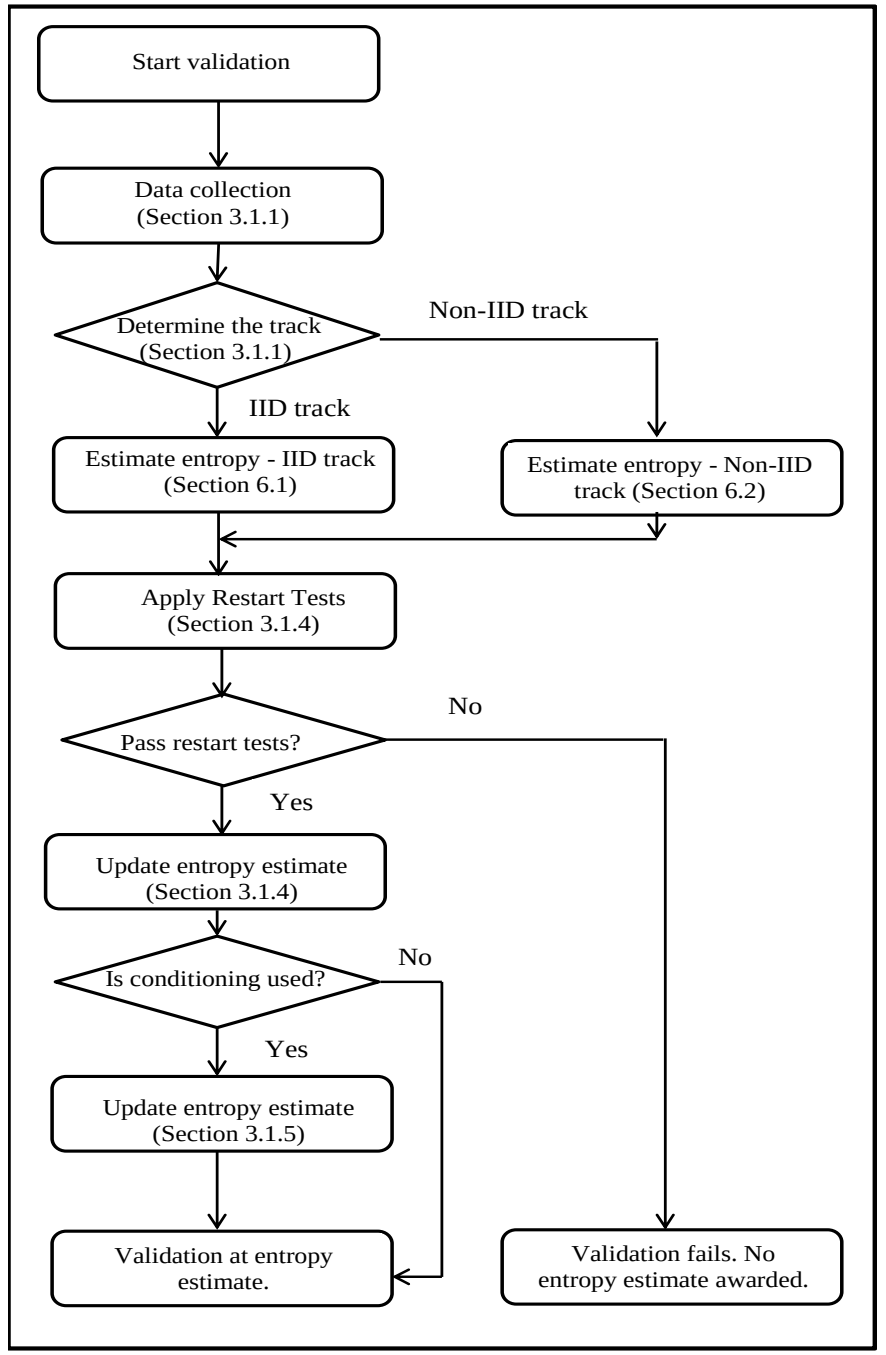
- IID means each sample independent of all others, independent of position in sequence of samples

How do we determine if source is iid?

- If designer says source is NOT iid, then it's NEVER evaluated as iid
- Otherwise, run a bunch of statistical tests to try to prove that source is NOT iid
- If we can't disprove it, we assume source is iid for entropy estimation

Entropy Estimation: Testing the Noise Source

- Data collection from noise source (optionally with post-processing)
- IID vs non-IID
- **Entropy estimation**
- Restart tests



Entropy Estimation*

IID Case:

- Estimate entropy by counting most common value and doing some very simple statistics.
- *Presentation on this later today.*

Non-IID Case:

- Apply many different entropy estimators against sequential dataset.
- Take minimum of all resulting estimates.
- *Presentation on this later today.*

Sample implementations of entropy estimation can be found at:

https://github.com/usnistgov/SP800-90B_EntropyAssessment

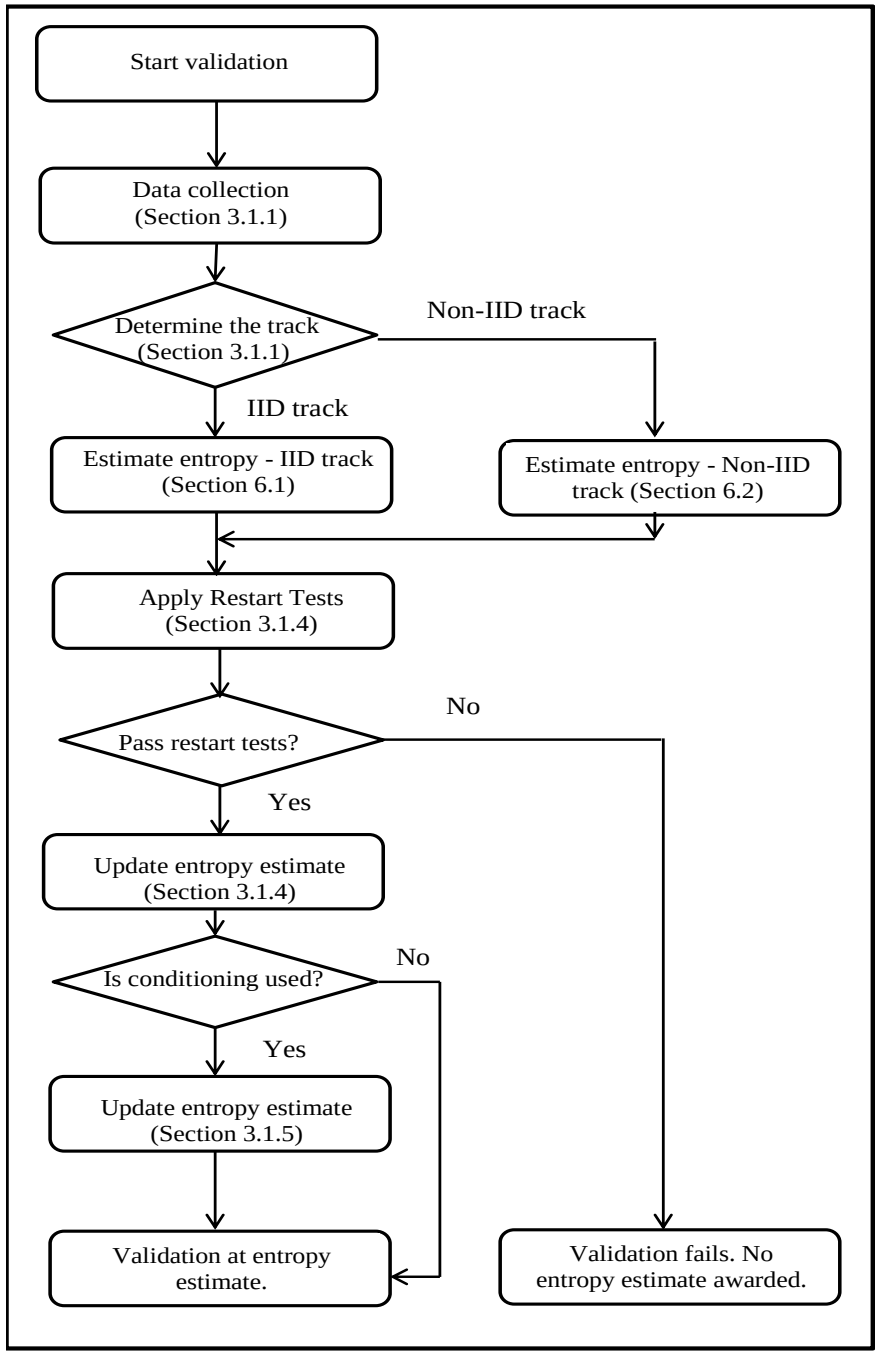
Entropy Estimation—Combining Estimates*:

- H[original]
  - We do entropy estimation (iid or noniid) on the sequential dataset.
- H[binary]
  - If the sequential dataset is not binary, we convert it to binary and do a second estimate on the first 1,000,000 bits that result.
- H[submitter]
  - The submitter had to estimate the entropy in his documentation package.
- H[I] = min(H[original,H[binary],H[submitter])
  - Just take the smallest estimate of the three (or two, if it’s binary data)

Entropy Estimation: Testing the Noise Source

- Data collection from noise source (optionally with post-processing)
- IID vs non-IID
- Entropy estimation
- **Restart tests**

* This is leaving out some details. See the full 90B draft for the full picture.



What Are Restart Tests?

- Some noise sources look a lot worse *across restarts*.
  - Restart = power cycle, hard reset, reboot
- Example: Repeated RSA prime factors
- Restart testing attempts to detect predictability that only becomes apparent when examining many sequences generated by a source across restarts.

Restart Datasets

Restart dataset is a 1,000 x 1,000 matrix of samples

- Each row is 1,000 successive samples, taken after the source restarted
- Column K is the Kth sample from each restart.

Use the matrix to generate two additional datasets:

- Row dataset = concatenate the rows together to make a 1,000,000 sample sequence
- Column dataset = concatenate the columns together to make a 1,000,000 sample sequence

Restart Testing*

- We start with entropy estimate H[I] from previous step.
- Use sequential entropy estimation techniques on row and column dataset:
  - results -> H[r], H[c]
- Two chances for the source to fail validation:
- Most Common Value in the Rows and Columns:
  - Use H[I] to compute an upper bound on number of times most common value should appear in a column or row.
  - If any row or column exceeds upper bound, then **the source fails validation**.
- Is H[r] or H[c] *too* low?
  - If min(H[r],H[c]) < H[I]/2, then **the source fails validation**.

Entropy Estimation Wrapup*

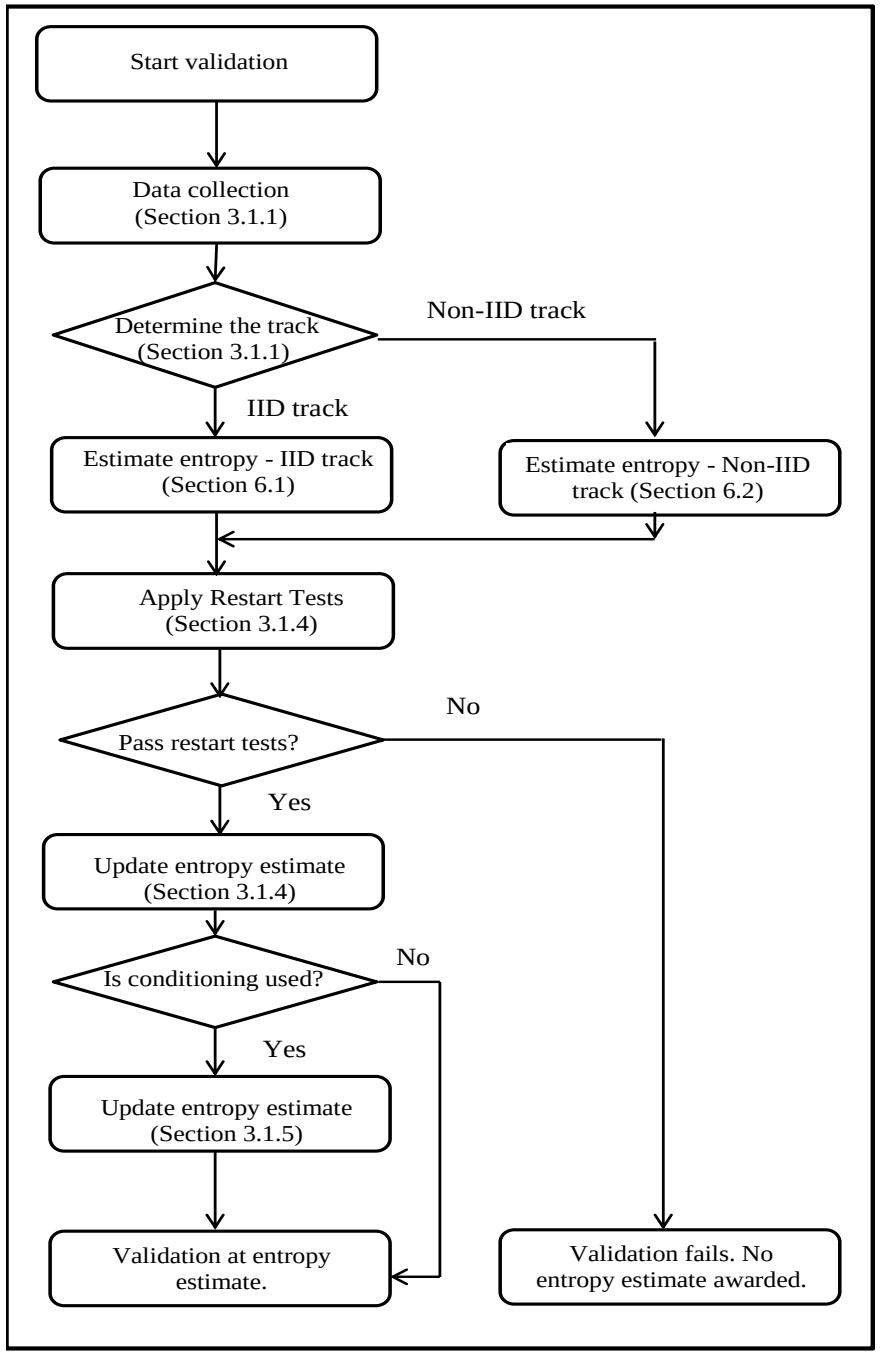
- H[original] = original entropy estimate from sequential dataset
- H[submitter] = submitter's original entropy estimate
- H[binary] = estimate from binary version of data (sometimes)
- H[r] = row dataset entropy estimate
- H[c] = column dataset entropy estimate

- All combined to give us the final estimate:

H[final] = min(H[original],H[submitter],H[binary],H[r],H[c])

Entropy Estimation: Wrapup*

- Collect the data
- IID or non-IID
- Sequential dataset -> h[sequential]
- Restart testing
  - May fail the source
  - May lower the estimate
- h[final] = minimum of all estimates
  - sequential, submitter, binary, row, column,
- Either FAIL source, or we have entropy estimate **for noise source**.



So, What's the Problem With All This?

- No set of general-purpose statistical tests can measure the entropy per sample in an arbitrary sequence of values.
- Right way to build a noise source is:
  - Design your noise source
  - Understand it
  - Model it
  - Use your model to estimate its entropy
  - Run general-purpose tests on outputs as a sanity check.
- We require design documentation and an entropy estimate from designer to support this...
- ...but we're limited in what resources we can demand for validation testing, and what expertise we can require from labs.

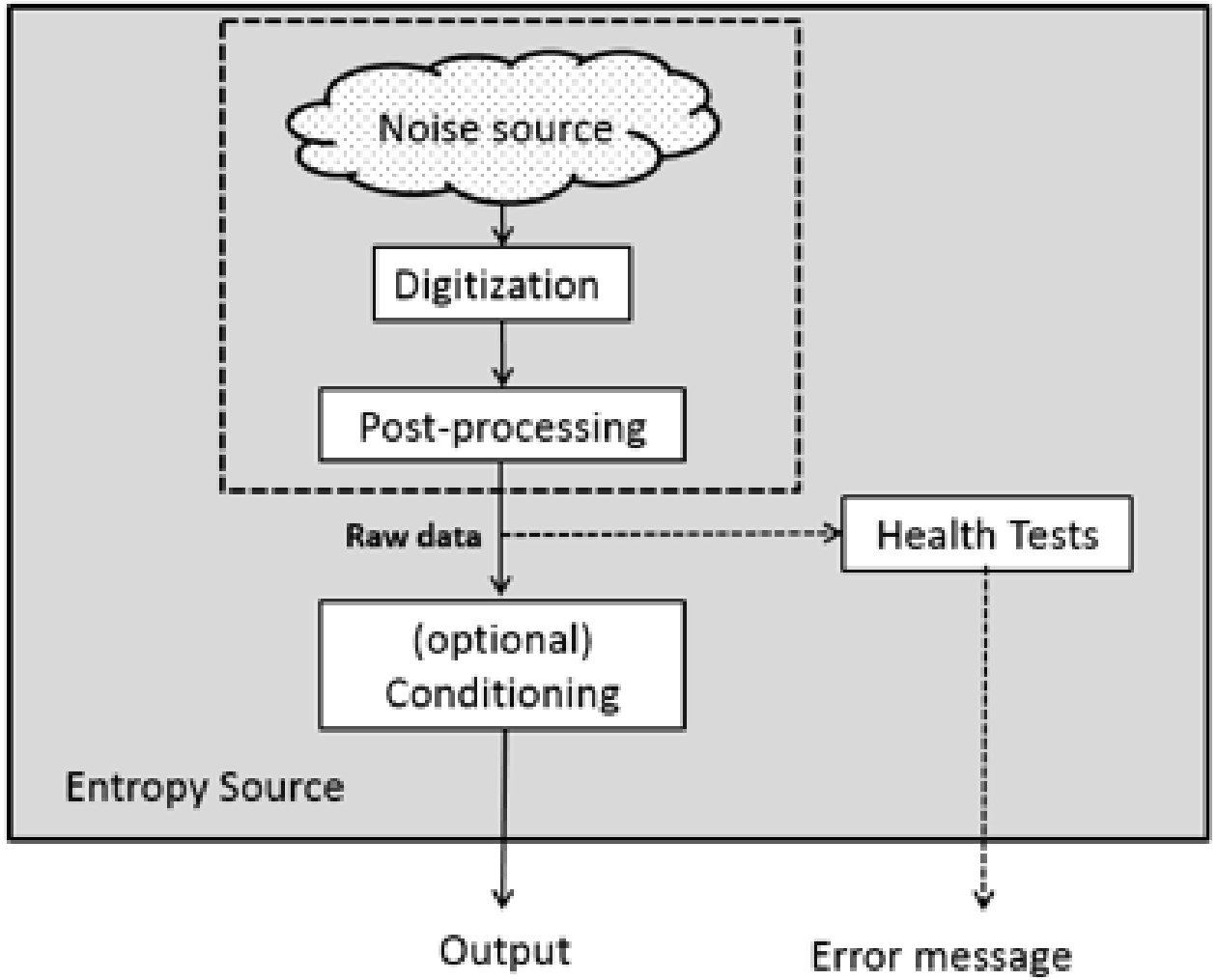
This approach is the best we know how to do given existing constraints.

Design Requirements

Noise source is validated by examining documentation and entropy estimation.

We still need to validate:

post-processing
health tests
conditioning.



Post-Processing

- There are three algorithms for post-processing permitted:
- Von Neuman's Method
  - Common method for unbiasing a biased but iid source
  - Note: applying it to a source with some dependencies can make things worse
  - 00 or 11 -> no output; 01 -> 1 bit output, 10 -> 0 bit output
- Linear Filtering
  - Apply any linear function to w-bit (non-overlapping) blocks to produce output
- Length of Runs
  - Count the lengths of runs
  - 000001111000110 -> 5,4,3,2,1
- Documentation specifies which (if any) post-processing function is used.

Conditioning

- There are six vetted conditioning functions based on cryptographic functions
  - HMAC, CMAC, CBC-MAC, any approved hash function, hash-df, bc-df
  - These are all algorithms, so the implementation can be validated.
- Designers may also use any conditioning function they like.
  - Documentation must specify the conditioning function.
  - Additional data collection and testing used for these functions.
- Validation has to verify the entropy accounting.

- *See presentation later today on conditioning functions.*

Health Tests: Big Picture

Noise sources can be fragile. Thus, all entropy sources MUST have health tests.

- Health tests examine the **raw noise source outputs**, NOT conditioned outputs.

- Tests must have minimal performance impact.
  - False positive rate is important

- 90B has two recommended health tests
- Designers can use ours or make up their own
  - with some extra requirements

Why is False Positive Rate Important?

- Health tests run continuously
  - They process **every raw noise source output**.
- Many sources can produce millions of outputs per hour
  - A 10^{-6} false positive rate means > one failure per hour!
- False positive rate needs to be tuned to make sure expected number of failures of healthy noise sources is acceptably small!
- This is a tradeoff—the lower the false positive rate, the larger the failure of the noise source that won't be detected!
- We think values between 2^{-40} and 2^{-50} are usually reasonable.

Health Tests: Continuous vs Startup

- Continuous Tests
  - Going on all the time behind the scenes—noise source outputs continue to be used unless the test detects a problem
  - If the tests detect a problem, this is signaled somehow to the device or calling application. Normally, you'd expect to see an error message or a device failure here.
- Startup Tests
  - Run on a sequence of noise source samples at startup, before those samples may be used for anything.
  - If the tests detect a problem, then the device refuses to function.

Our Health Tests

- Repetition Count Test – Detect when the source gets “stuck” on one output for much longer than expected.
- Adaptive Proportion Test – Detect when one value becomes much more common in output than expected.

- Note that tests:
  - Require minimal resources
  - Outputs can be used as they are produced
  - Allow tunable false-positive rates

Repetition Count

The repetition count test is an updated version of the "stuck test."

- Let H = entropy estimate for noise source
- P[max] = 2^{-H} is maximum probability for any value.
- Probability of seeing T identical values in a row <= $P[max]^{(T-1)}$
- Choose T so that $P[max]^{(T-1)}$ is small enough that false positives aren't important (for example, 2^{-50}).
- Memory requirements = previous sample value, counter of how many repetitions were seen

Adaptive Proportion Test

The Adaptive Proportion Test tries to detect when one particular value has become much more common than expected.

- Choose window size (W): 512 for nonbinary sources, 1024 for binary
- Based on entropy/sample (H), W, and acceptable false positive rate, choose cutoff value C.
- Every W samples, the test restarts.
- We take the first sample in the window, and count how many times it appears in the whole window.
- If it appears C or more times, then we detect an error condition.

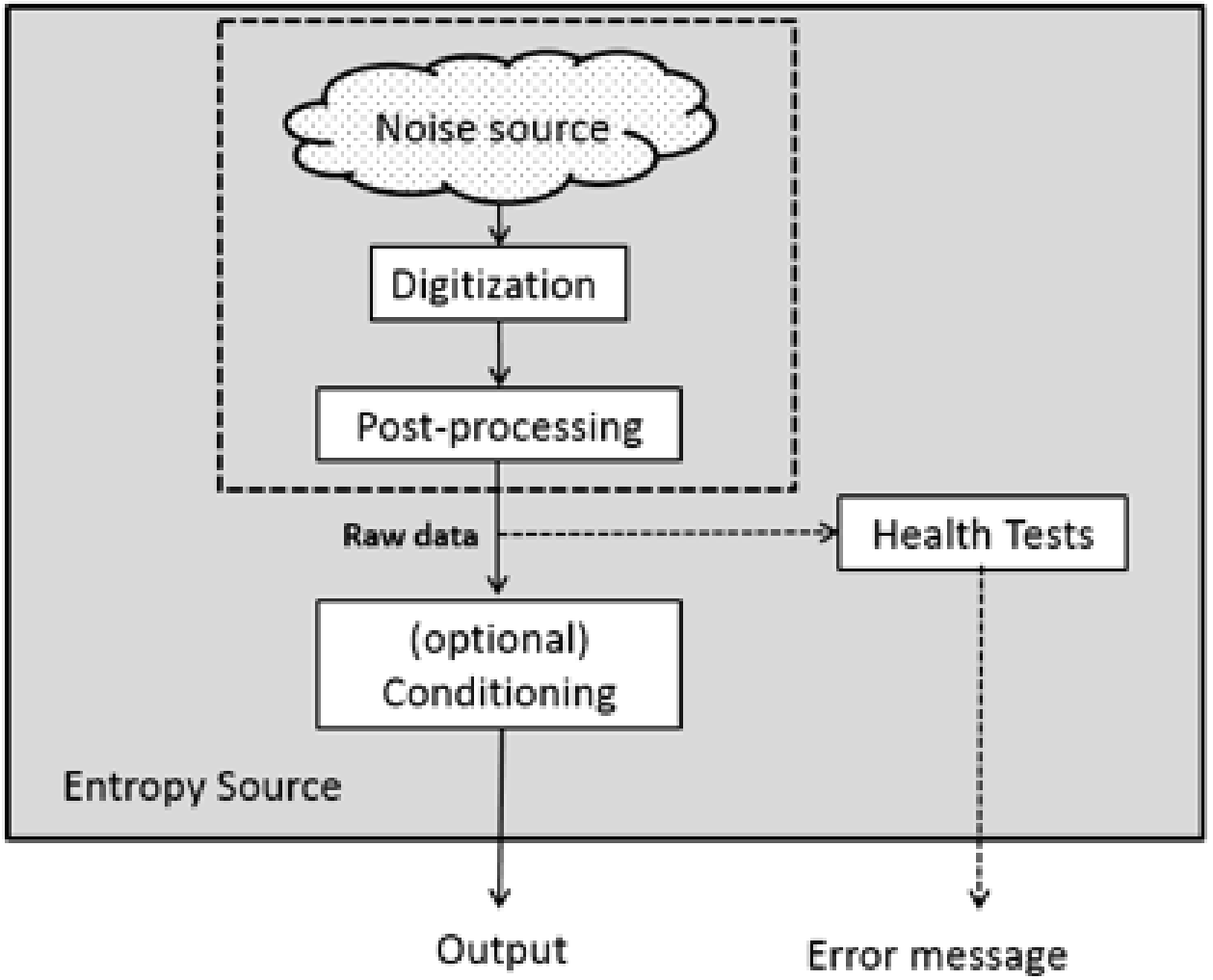
Alternative Tests

- Designers should understand their sources much better than we can, when writing this document.
- So we allow designers to specify their own continuous health tests, if they meet the requirements: (H = entropy/sample of noise source)
  - If a single value appears more than 100/H times in a row, the tests must detect this with high probability.
  - If a single value begins to appear with probability $P >= 2^{-H/2}$, the tests must detect this with acceptable probability.
- Designers can provide statistical simulations or some other proof that their tests meet these requirements.

Startup Testing

- Startup testing happens when the module starts up.
- The noise source generates a pool of at least 4096 samples, with the continuous health tests running. The samples CANNOT be used until the testing is complete.
- If no error is detected, then the module MAY use or discard those samples.
- The designers MAY include additional tests at startup.

Wrapup


Noise source
Digitization
Post-processing
Raw data
Health Tests
(optional)
Conditioning
Entropy Source
Output
Error message

Wrapup: Lots of New Stuff in SP 800-90B

- Documentation requirements
- Data collection requirements
- Entropy estimation
- Restart testing
- Post-processing
- Conditioning
- Health Tests

Wrapup: Open Questions

- Post-processing functions
  - Are they necessary?
  - What else should be included?
  - What should be removed?
- Testing strategy
  - How can we improve this, given constraints on cost of evaluation and lab resources?
- Health tests