NIST SP 800 90 Series
Brief Overview and Updates

Meltem Sönmez Turan, NIST

April 27, 2021

NIST National Institute of Standards and Technology
U.S. Department of Commerce

ITL INFORMATION TECHNOLOGY LABORATORY

Aim of the talk

- Brief overview of SP 800 90B
- General concepts, requirements, and validation process
- Planned updates

Cryptographic random numbers

Security of cryptographic protocols relies of strong random bits.

- Cryptographic keys, IVs, nonce, mask, challenges in C/R protocols etc.

Assumption: The bits are generated uniformly at random and are unpredictable.

Practical problems in real-world applications

- Heninger et al. (2012) performed a network survey of TLS and SSH servers, and collected certificates and recovered RSA and DSA keys, due to low entropy during key generation.
- Bernstein et al. (2013) studied the Taiwan's national "Citizen Digital Certificate" database and efficiently factored 184 distinct RSA keys due to low-quality hardware RNG.

- Heninger et al., *Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*, 21st USENIX Security Symposium, 2012.
- Bernstein et al., *Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild*. ASIACRYPT 2013

Challenges in random number generation

Designing RBGs is challenging:

- Finding a robust randomness source and correctly extracting randomness
- Difficult to know how unpredictable the outputs are (i.e., estimating entropy)
- Difficult to statistically model the process
- Difficult to understand the effects of outside parameters and environmental conditions (e.g., humidity, temperature) on the source

Validating RBGs is challenging.

- Expert knowledge on the noise sources
- Difficult to verify some of the claims
- Practical constraints (e.g., time, budget)

NIST Standards and Guidelines for Randomness

NIST SP 800-90 Series aims to improve the overall quality of the generators by specifying design principles (do not to provide full solutions for constructing RBGs)

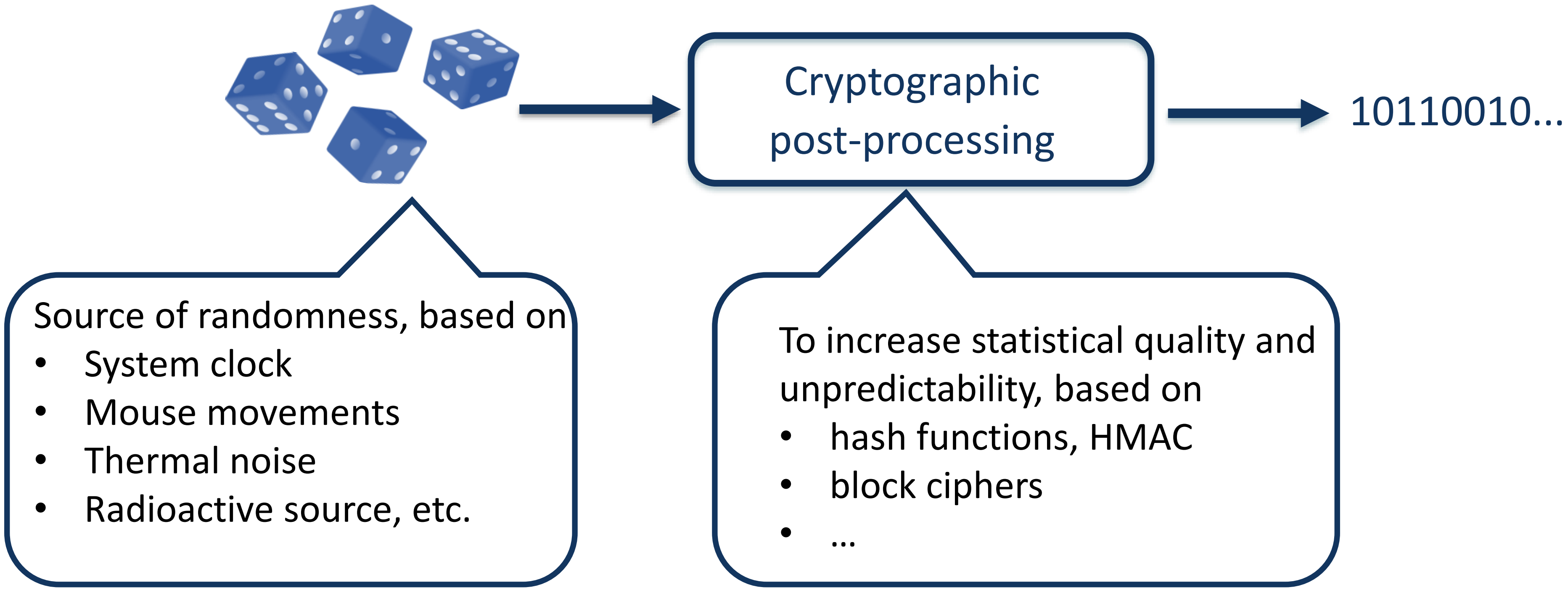
- SP 800-90A *Recommendation for RNG using Deterministic Random Bit Generators* (June 2015, previous versions dated June 2006, January 2012).

- SP 800-90B *Recommendation for the Entropy Sources Used for Random Bit Generation* (Jan. 2018, previous versions dated August 2012, Jan. 2016)

- SP 800-90C *Recommendation for Random Bit Generator (RBG) Constructions* (April 2016, previous versions dated August 2012) working on the new version.

NIST SP 800-22 *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010, previous version dated 2008) will be revised, comments are welcome.*

Cryptographic random number generators



Cryptographic post-processing

10110010...

Source of randomness, based on

- System clock
- Mouse movements
- Thermal noise
- Radioactive source, etc.

To increase statistical quality and unpredictability, based on

- hash functions, HMAC
- block ciphers
- …

Entropy

Entropy is measure of randomness and uncertainty.

- Information theory: a measure of the amount of information contained in a message.

Different mathematical formulations based on the *probability distribution* of the random variables.

Let *X* be a discrete random variable that takes values from a finite set $x_1,\ldots, x_n$ with probabilities $p_1,\ldots, p_n$.

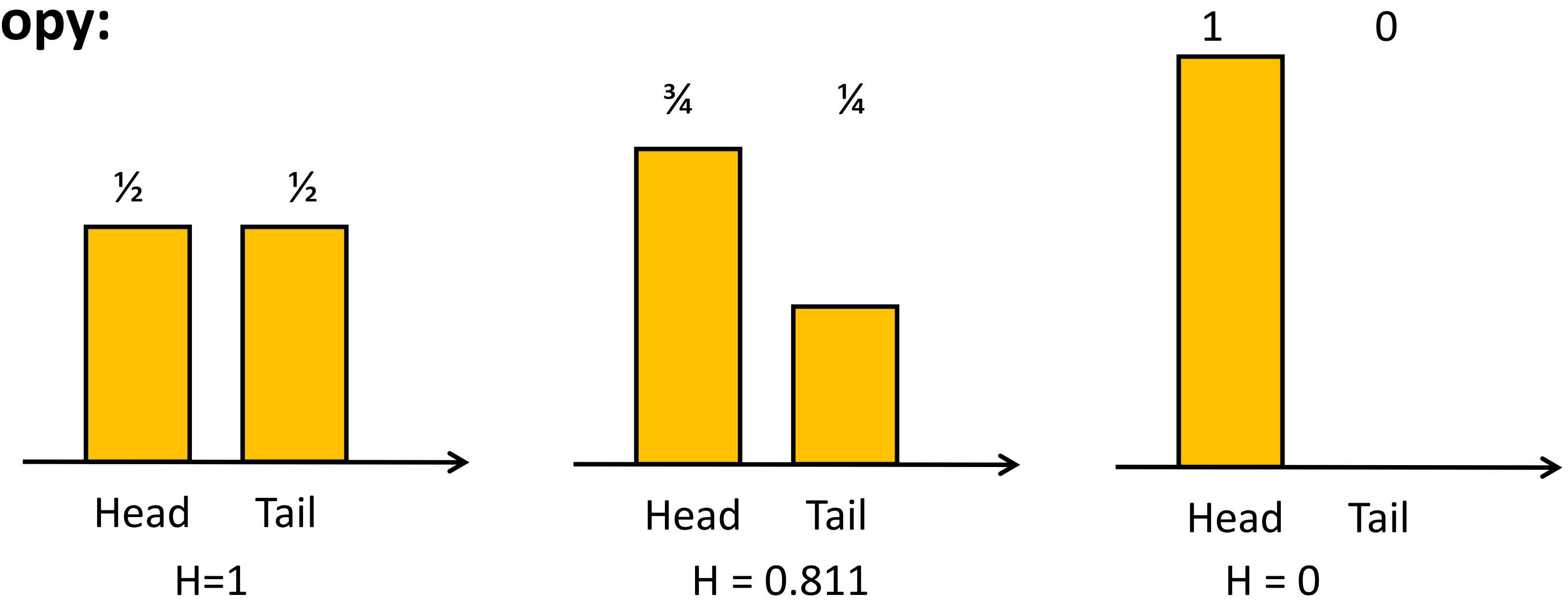
Shannon (compression): $H = -\sum p(x) \log p(x)$,

Min-entropy (guessing entropy): $H = -\log(\max p(x))$

Renyi entropy (collision entropy): $H = \frac{1}{1-\alpha} \log \sum p(x)^{\alpha}$

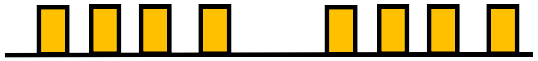
Example: Flipping a coin

Shannon entropy:

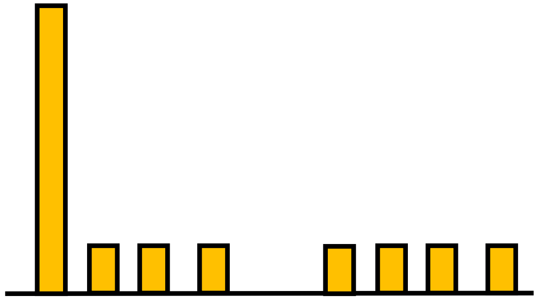


Shannon's entropy reaches its maximum value when each outcome is equally probable. After tossing a fair coin, we learn one bit of information. Flipping the coin *k* times provides *k* bit of information.

Example: Unpredictability of keys



Equally likely 2^{128} keys



All-zero key with pr. 1/2, and 2^{128} -1 equally likely keys

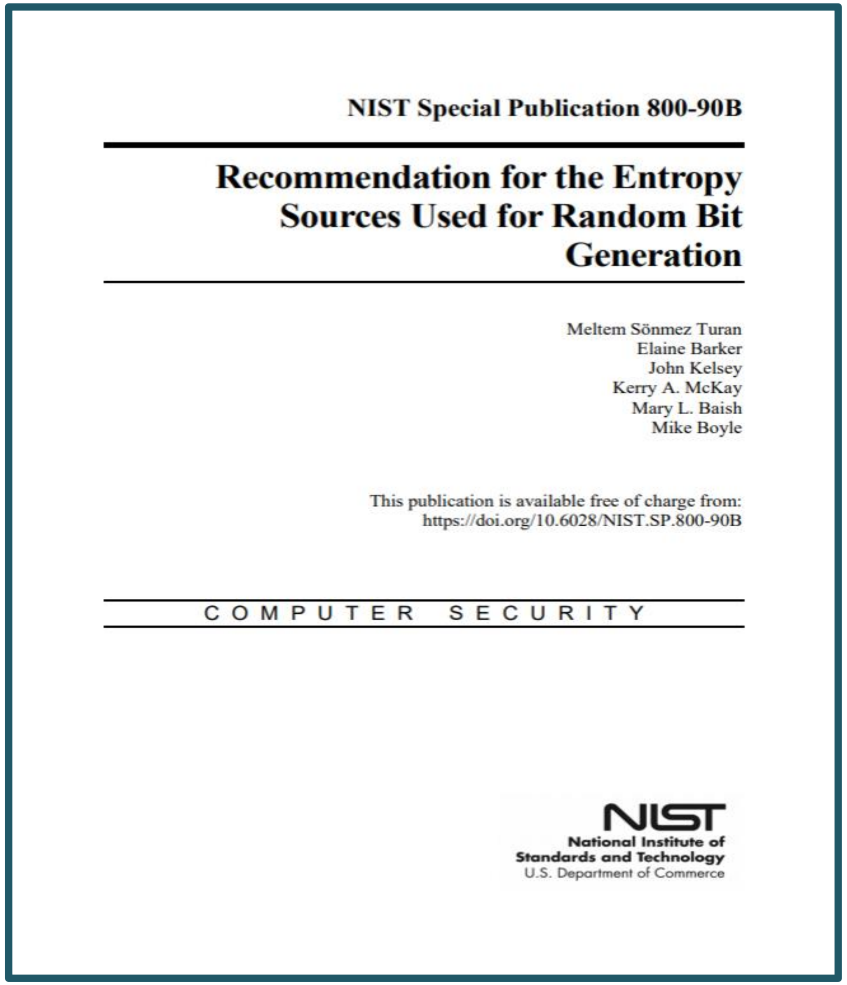
Shannon entropy H = 128 bit
Min-entropy H = 128 bit

Shannon entropy H = 65 bit
Min-entropy H = 1 bit

With probability ½, the key may be guessed on the first try!

Overview of NIST SP 800 90B

- Provides an entropy source definition and a model.
- Specifies design principles for entropy source components.
- Lists requirements for the entropy source, including interactions between components, parameter selections.
- Specifies black-box entropy estimation techniques.
- Describes the validation process



NIST Special Publication 800-90B

Recommendation for the Entropy Sources Used for Random Bit Generation

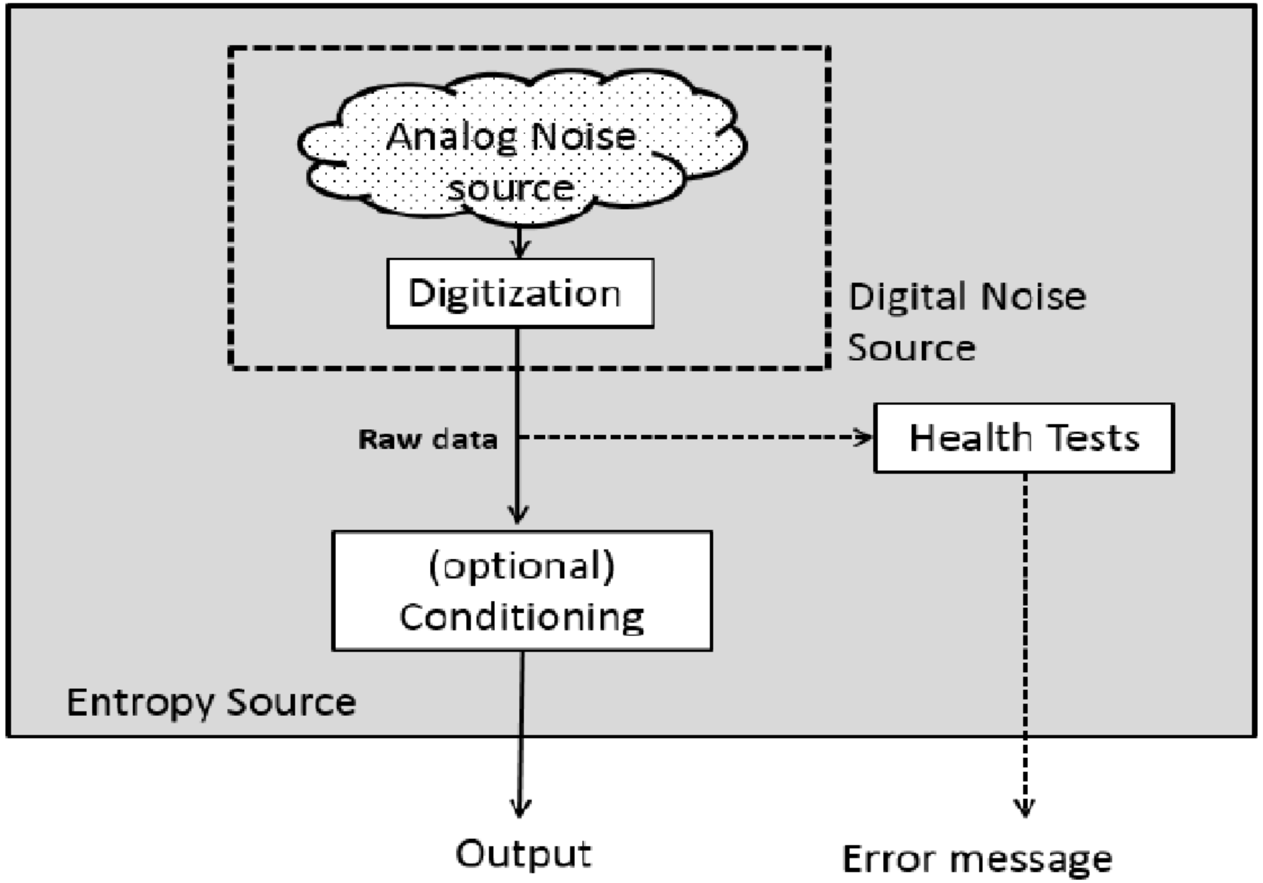
Meltem Sönmez Turan
Elaine Barker
John Kelsey
Kerry A. McKay
Mary L. Baish
Mike Boyle

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-90B

COMPUTER SECURITY

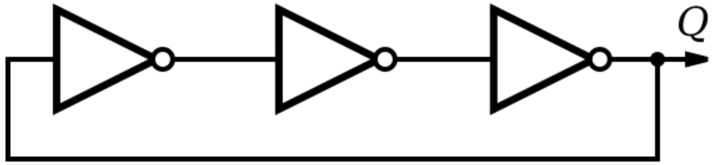
NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Entropy Source Model



Noise Source

A noise source provides bits or bitstrings of fixed length having approximately same amount of entropy per output.



Physical noise sources

- Dedicated hardware (e.g., ring oscillators)
- Behaves similar for all copies of the RNG.
- Accurate modelling is suitable.

Non-Physical noise sources

- Uses system data/human interaction (e.g., hard drive access times, mouse movements)
- Different behavior depending on the platform
- Does not allow accurate modelling.

Health Tests

Noise sources can be fragile and can be affected by the external conditions. Health tests are required. They aim to detect deviations from intended behavior of the source during operation.

Types: Start-up, continuous and on-demand tests

Pre-defined health tests:

Adaptive proportion – detect when one value becomes much more common

Repetitive count tests – detect when the source gets stuck on one output

User-defined tests tailored to the failure modes of the source

Conditioning Component

Noise source outputs may be statistically biased (e.g., biased coin), or correlated. Conditioning component processes the noise source outputs to increase statistical quality of the outputs, or entropy rate.

90B specifies set of vetted conditioning components:

- Keyed: HMAC with any approved hash function, CMAC and CBC-MAC
- Unkeyed: Approved hash functions, hash_df (specified in 90A), Block_cipher_df (specified in 90A)

Vendors can use other conditioning components.

Conditioning is optional.

Conceptual Interfaces

90B describes three conceptual interfaces that can be used to interact with entropy source:

- GetEntropy inputs requested amount of entropy, and outputs the string that provides the requested entropy.
- GetNoise inputs requested number of samples from the noise source and receives the noise source outputs.
- HealthTest inputs the type of the tests to be performed.

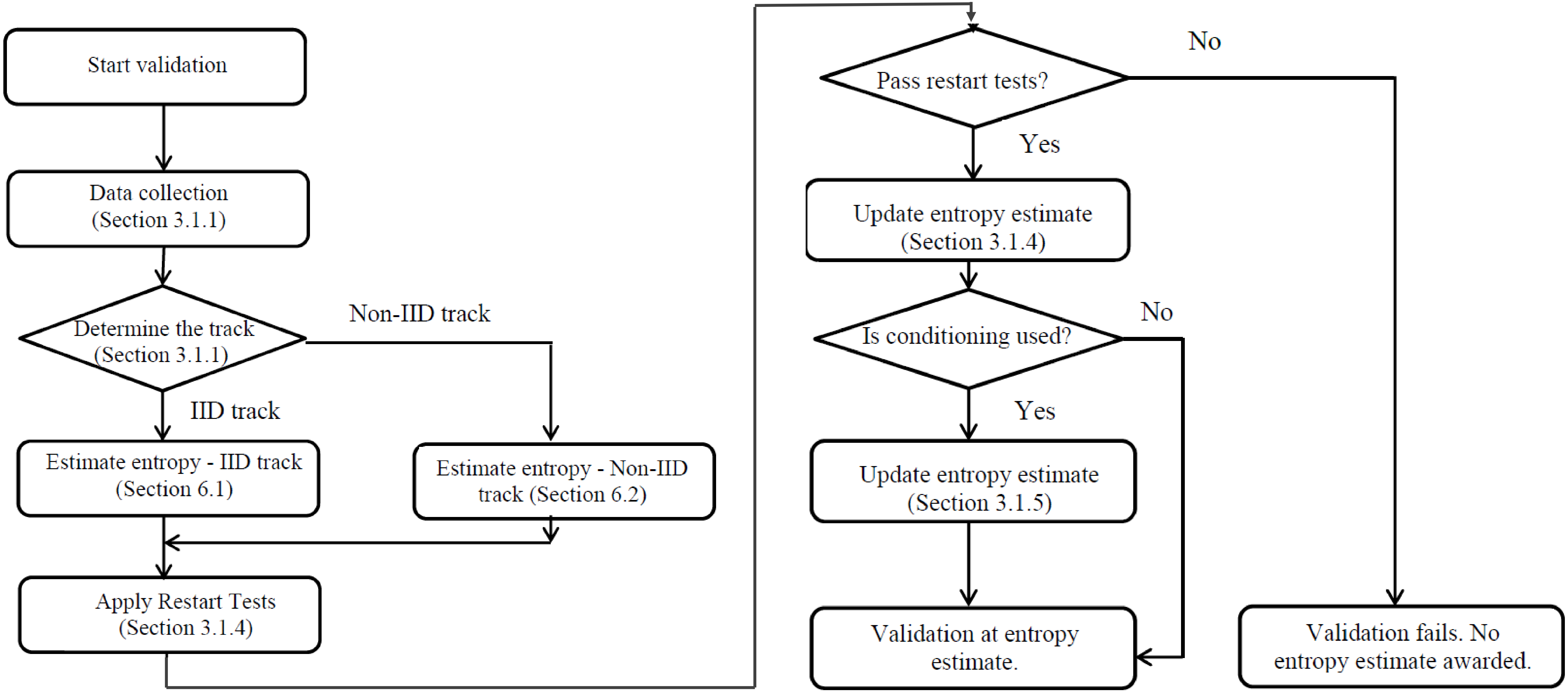
Basic requirements

- Documentation on the entire design of the entropy source including interactions between components, parameter selections
- Justification for why the noise source can be relied upon to produce bits with entropy
- Requirements on the noise source, conditioning component (correctness of the implementations etc.), health tests
- Requirements on data collection
- Range of operating conditions
- Entropy estimate from the submitter
- IID justification, if applicable
- etc.

Validation Process

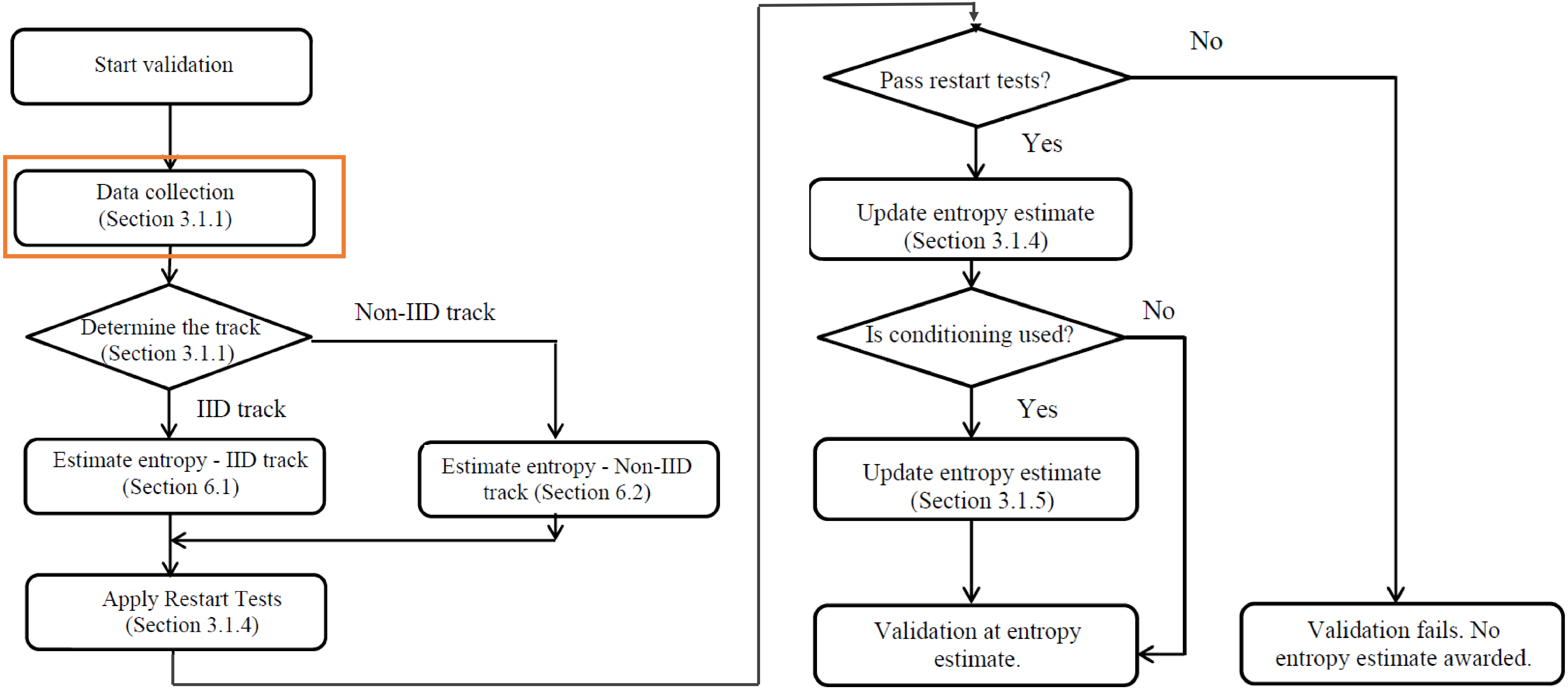
- Entropy source validation is necessary in order to obtain assurance that all relevant requirements are met.
- 90B provides requirements for validating an entropy source at a specified entropy rate.
- Validation consists of testing by an NVLAP-accredited laboratory against the requirements of 90B, followed by a review of the results by CAVP and CMVP.

Validating Entropy Sources



Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

Validating Entropy Sources


Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

Data Collection

Entropy estimation requires entropy source and noise source outputs.

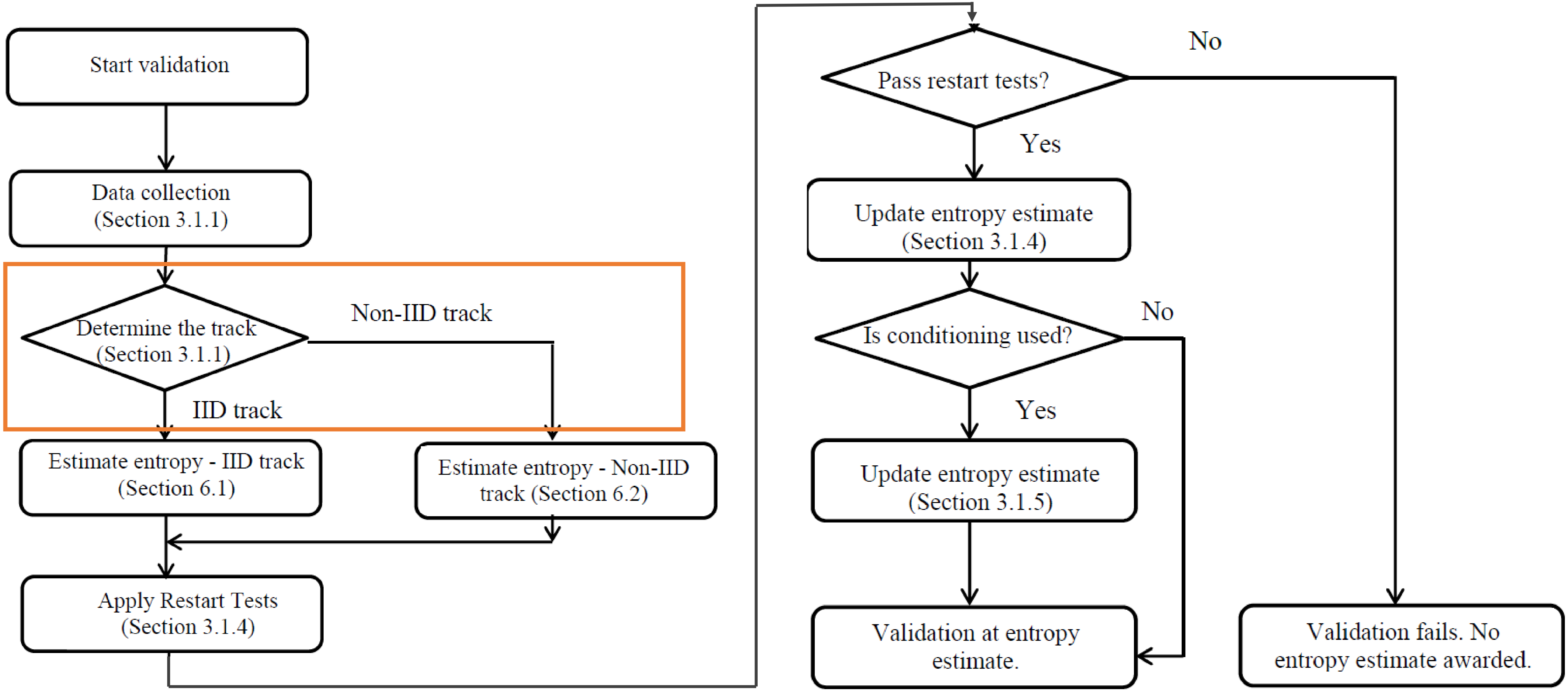
Two datasets from the noise source :

- Sequential dataset: 1,000,000 successive noise source outputs
- Restart dataset: 1,000 restarts, 1,000 samples from each restart (Restart = reboot, hard reset etc.)

One dataset for non-vetted conditioned outputs (if applicable):

- Conditioner dataset: 1,000,000 successive samples from conditioner

Validating Entropy Sources


Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

Determining the Track

IID: independent and identically distributed

- each sample has the same probability distribution as every other sample, and
- all samples are mutually independent.

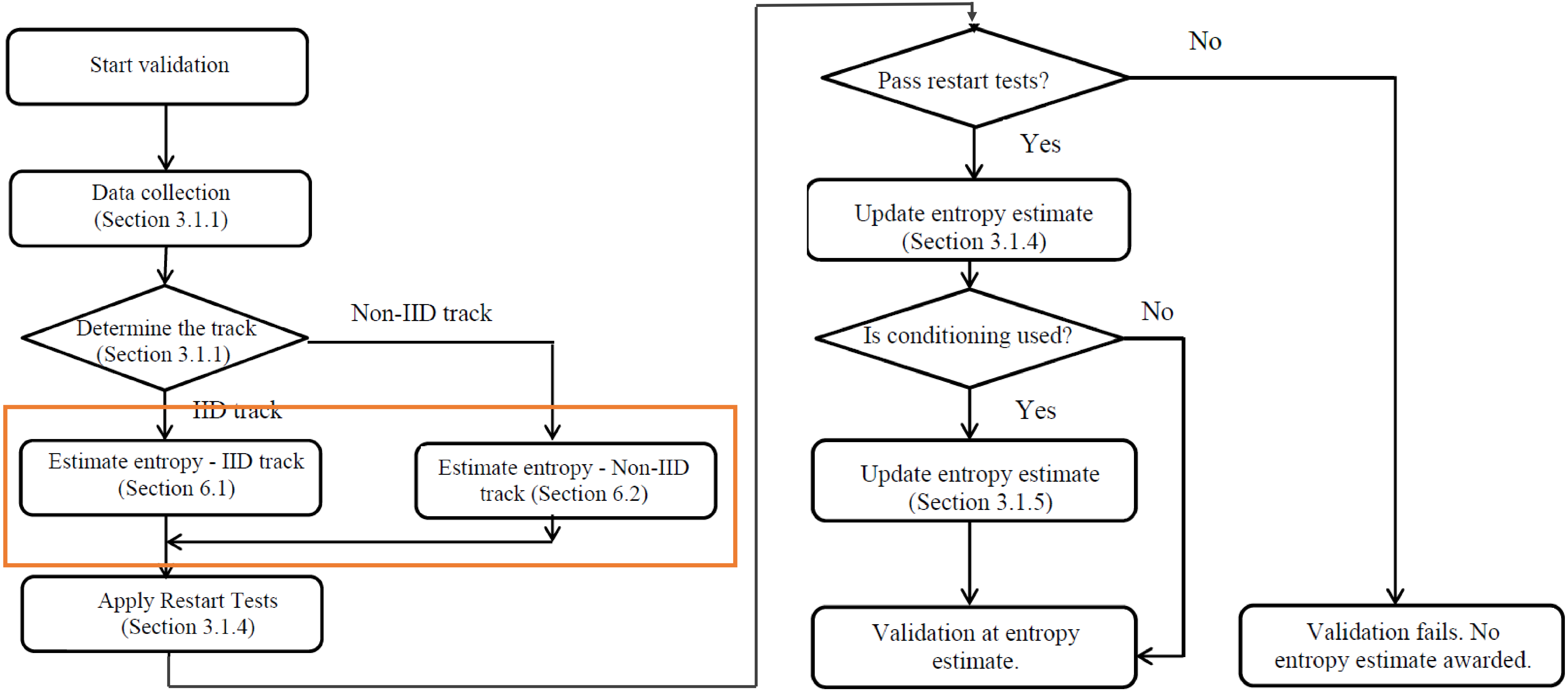
Two tracks: IID track and non-IID track

IID track is used only when both conditions are satisfied:

1. The submitter makes an IID claim on the noise source, based on the submitter’s analysis of the design. The submitter shall provide rationale for the IID claim.

2. The input datasets (sequential, row, column and conditional) are tested using the statistical tests (permutation and chi-square) to verify the IID assumption, and the IID assumption is verified (i.e., there is no evidence that the data is not IID).

Validating Entropy Sources


Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

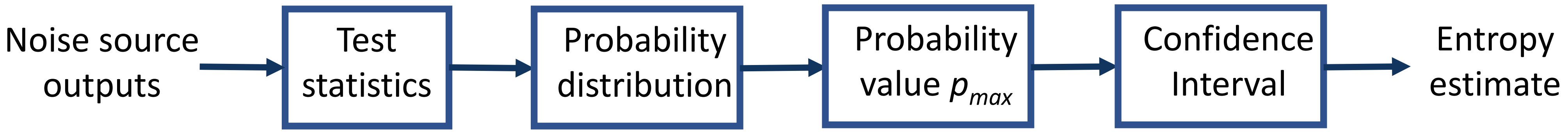
Estimating entropy

Main question: How unpredictable are the entropy source outputs?

- Min-entropy is used to measure unpredictability.

Entropy estimation:

- Black-box approach, various estimation methods based on different statistical models (i.e., Markov models), and predictors



Noise source outputs → Test statistics → Probability distribution → Probability value p_{max} → Confidence Interval → Entropy estimate

Two tracks:

- For IID track: based on the pr. of the most common value
- For non-IID track: multiple estimation methods are used.

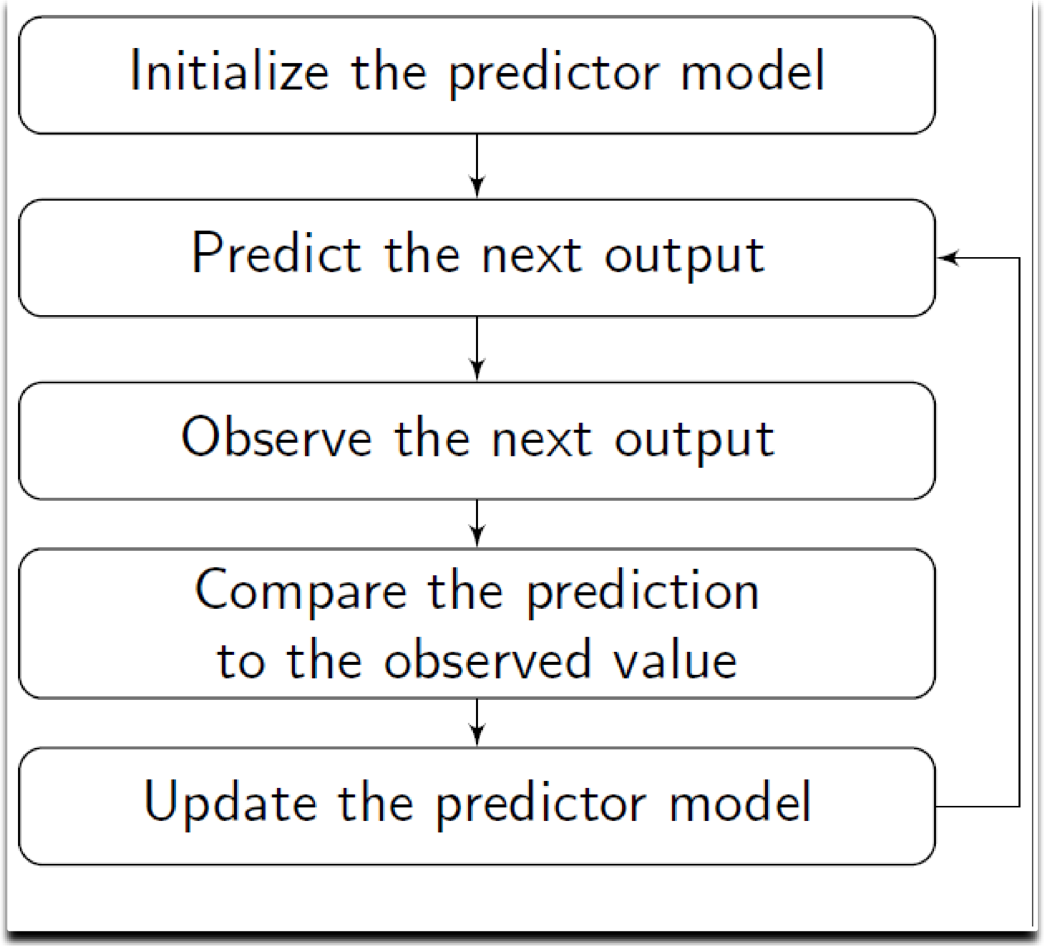
Estimating entropy using predictors

Predictors behave more like an attacker, who observes source outputs and makes guesses based on previous observations.

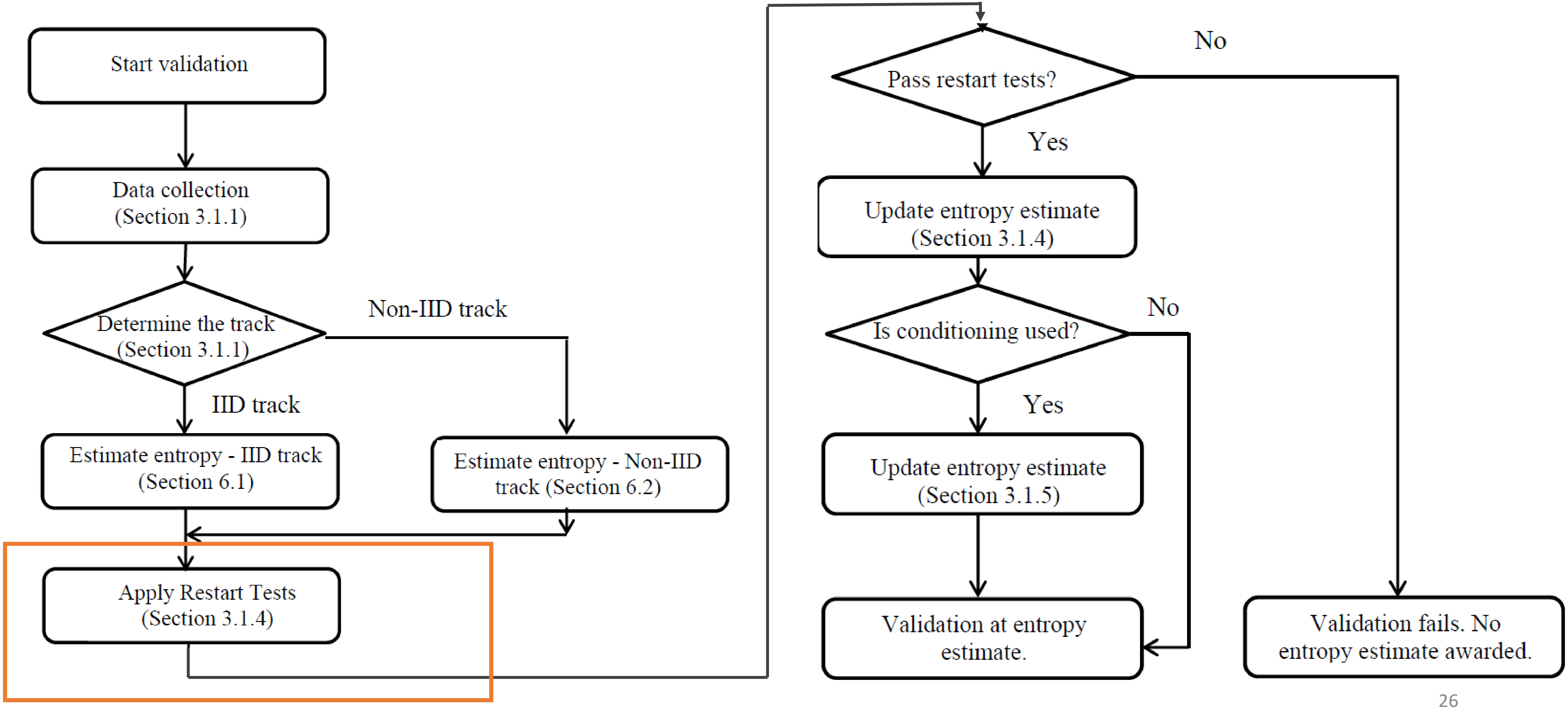
Generic predictors: Multi Most Common in Window Prediction, Lag Prediction, MultiMMC Prediction, LZ78Y Prediction

Global predictability: Number of correct predictions/Total number of predictions

Local predictability: Length of the longest run of correct predictions


Initialize the predictor model
Predict the next output
Observe the next output
Compare the prediction to the observed value
Update the predictor model

Validating Entropy Sources


Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

Restart Tests

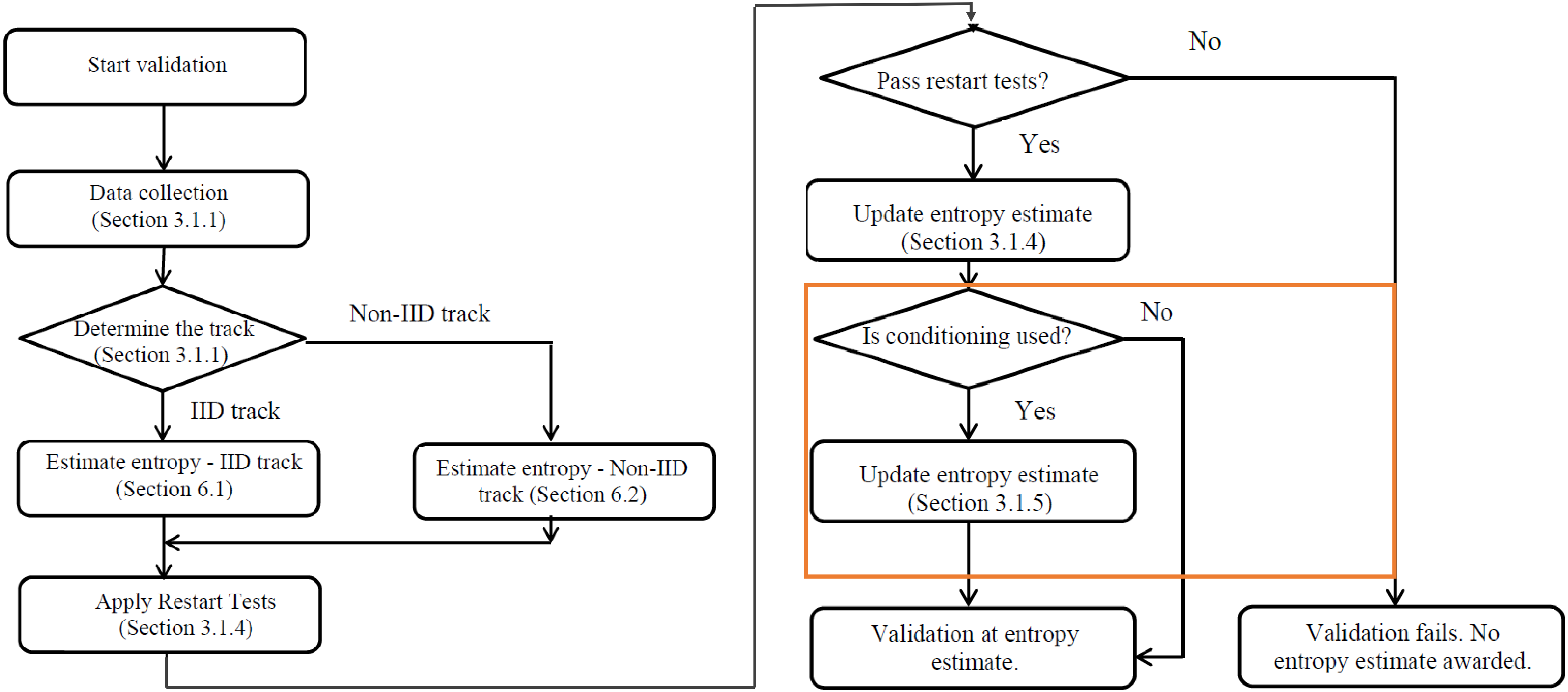
Noise sources may be predictable after restarts. (restart may be hard reset, reboot etc.)

Restart testing aims to detect predictability that only becomes apparent when examining many sequences generated by a source across restarts.

Restart dataset is a 1,000 x 1,000 matrix of samples: Row datasets and Column datasets

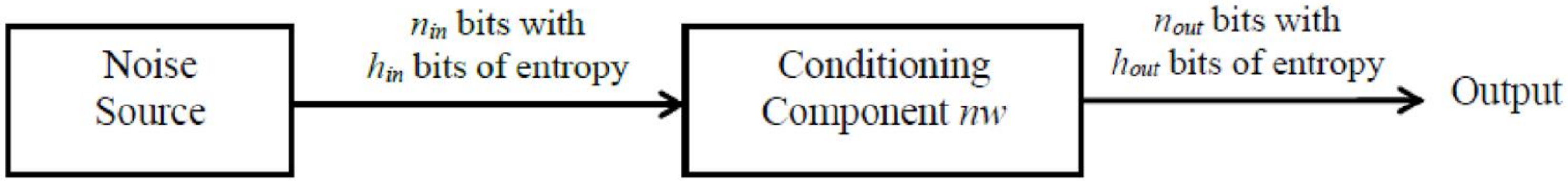
- Sanity test based on the frequency of the most common sample
- Entropy estimation using row and column datasets
- If the new estimate too low, validation fails.

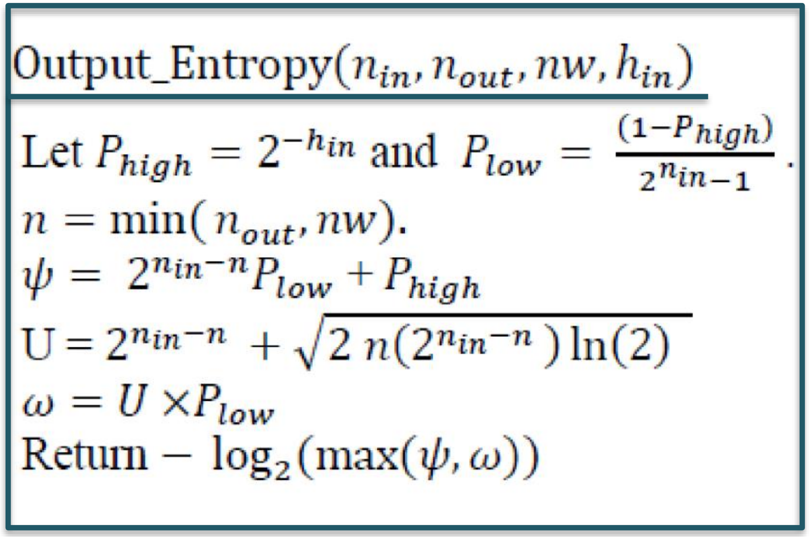
Validating Entropy Sources


Start validation
Data collection (Section 3.1.1)
Determine the track (Section 3.1.1)
Non-IID track
IID track
Estimate entropy - IID track (Section 6.1)
Estimate entropy - Non-IID track (Section 6.2)
Apply Restart Tests (Section 3.1.4)
Pass restart tests?
No
Yes
Update entropy estimate (Section 3.1.4)
Is conditioning used?
No
Yes
Update entropy estimate (Section 3.1.5)
Validation at entropy estimate.
Validation fails. No entropy estimate awarded.

Conditioning

Entropy estimate is updated if a conditioning component is used.


Noise Source
n_{in} bits with h_{in} bits of entropy
Conditioning Component nw
n_{out} bits with h_{out} bits of entropy
Output



Output_Entropy$(n_{in}, n_{out}, nw, h_{in})$

Let $P_{high} = 2^{-h_{in}}$ and $P_{low} = \frac{(1-P_{high})}{2^{n_{in}-1}}$.

$n = \min(n_{out}, nw)$.

$\psi = 2^{n_{in}-n} P_{low} + P_{high}$

$\mathrm{U} = 2^{n_{in}-n} + \sqrt{2\,n(2^{n_{in}-n})\ln(2)}$

$\omega = U \times P_{low}$

Return $-\log_2(\max(\psi, \omega))$

For vetted conditioning:

$$h_{out} = \text{Output\_Entropy}(n_{in}, n_{out}, nw, h_{in})$$

For non-vetted conditioning:

$$\min(\text{Output\_Entropy}(n_{in}, n_{out}, nw, h_{in}), 0.999\mathrm{n}_{out}, h' \times n_{out})$$

Caveats

- No set of general-purpose statistical tests can measure the entropy per sample in an arbitrary sequence of values.
- Better way is to understand the unpredictability of the noise source outputs, model it, and using the model to estimate the entropy. Run general purpose tests on outputs as a sanity check.
- We require design documentation and an entropy estimate from the designer to support it, but limitations what resources can demand for validation testing, and what expertise we can require from labs.

Planned Updates

NIST SP 800 90C & 90A

Big revision is planned to include simpler constructions:

RBG1 does not have access to a randomness source after instantiation

RBG2 includes one or more entropy sources that are used to instantiate and reseed the DRBG within the construction.

RBG3 construction is designed to provide a security strength equal to the requested length of its output by producing outputs that have full entropy.

Working on preparing a draft. Later 90A will be revised for consistency.

Revising NIST SP 800 90B

Since January 2018, received many comments and suggestions

- to improve technical quality of the standard
- to improve efficiency of the tests (e.g., permutation testing)
- to address issues with specific designs approaches
- editorial comments/typos/clarifications etc.

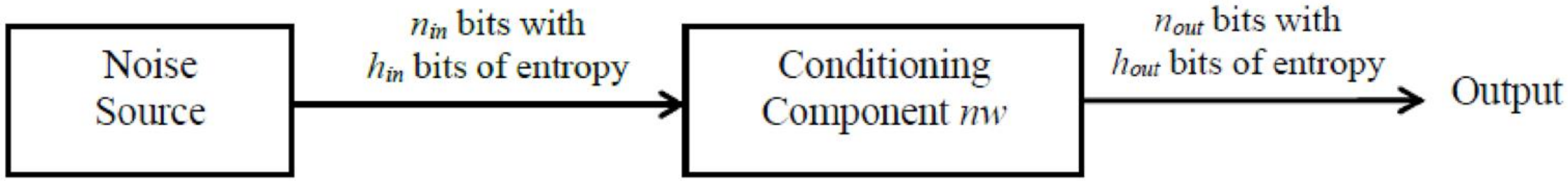
In August 2020, CMVP published the IG 7.19 Interpretation of SP 800-90B Requirements, based on the comments from CMUF Entropy WG.

Full-Entropy Outputs

A full-entropy bitstring has an amount of entropy equal to its length,

- has ideal randomness properties and may be used for any cryptographic purpose regardless of the security strength required.
- may be truncated to any length such that the amount of entropy in the truncated bitstring is equal to its length.

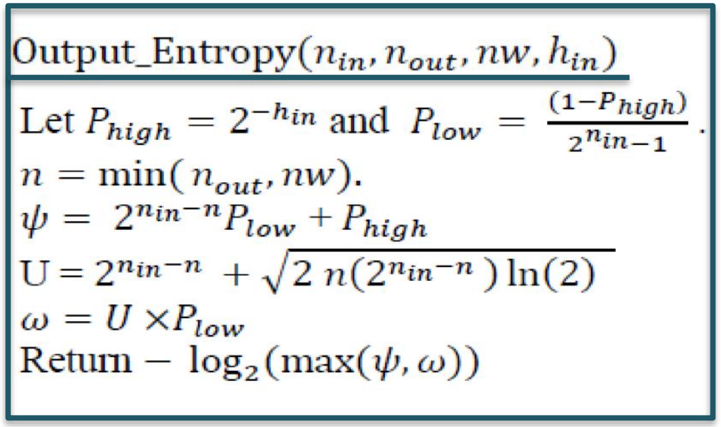
Entropy estimation techniques slightly underestimates the entropy of ideal random sequences, using conditioning components is necessary to achieve full entropy outputs.



Full-Entropy Outputs

NIST SP 800 90 series assumes that a bitstring has *full entropy,* if the amount of entropy per bit is at least $1 - \varepsilon$, where ε is at most 2^{-32}.

Only vetted conditioning components can be used to produce full entropy outputs.



$\text{Output\_Entropy}(n_{in}, n_{out}, nw, h_{in})$

Let $P_{high} = 2^{-h_{in}}$ and $P_{low} = \frac{(1-P_{high})}{2^{n_{in}-1}}$.

$n = \min(n_{out}, nw)$.

$\psi = 2^{n_{in}-n} P_{low} + P_{high}$

$U = 2^{n_{in}-n} + \sqrt{2\,n(2^{n_{in}-n})\ln(2)}$

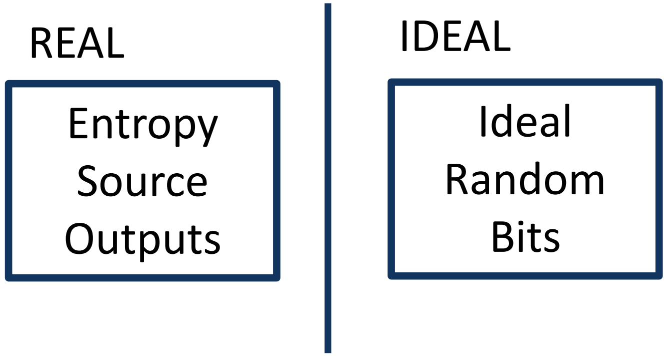
$\omega = U \times P_{low}$

$\text{Return} - \log_2(\max(\psi, \omega))$

For vetted conditioning:

$$h_{out} = \text{Output\_Entropy}(n_{in}, n_{out}, nw, h_{in})$$

Definition is based on a distinguishing game.



Conditioning Components - Truncation

90B says: *Truncating vetted conditioning components is allowed, and entropy estimate is reduced to a proportion of the output*.

Ex: SHA-512 output with 128 bits of entropy truncated to 256 bits.

Consider as vetted: Output has 64-bits of entropy

Consider as non-vetted: Output has approx. 120 bits of entropy.

To avoid this inconsistency, truncated vetted conditioning components can be used as vetted components.

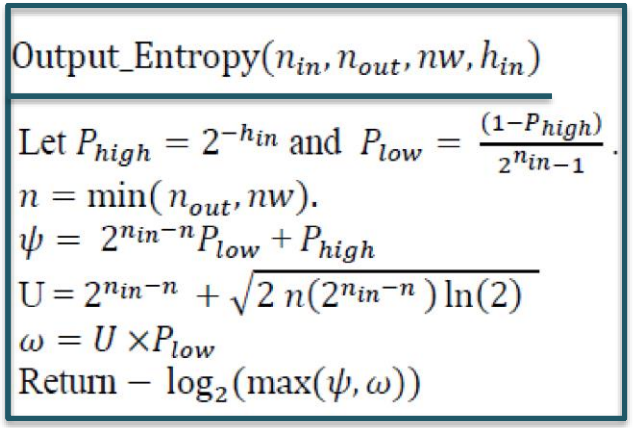
Bijective Conditioning Components

The output entropy formula models the conditioning as a random function, not a bijective function.

In random functions, due to internal collision, the output entropy decreases.

When a bijective function is used, and no need to apply the output entropy formula to reduce the entropy estimate.

Input entropy can be assumed to be same as output entropy.

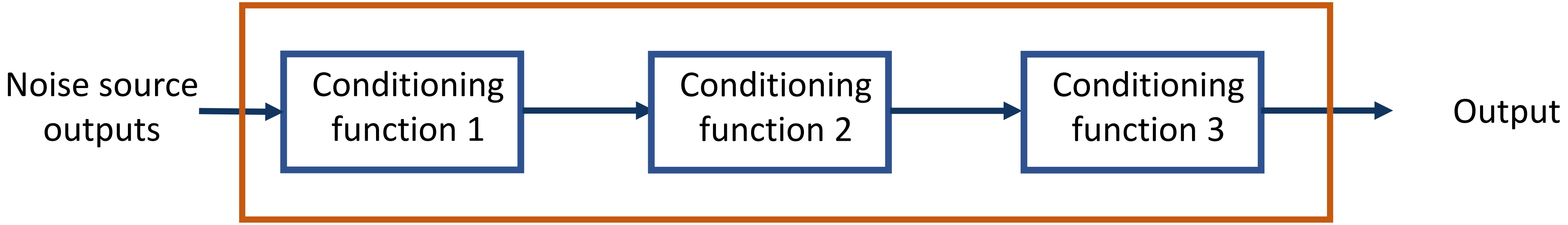


Output_Entropy($n_{in}, n_{out}, nw, h_{in}$)

Let $P_{high} = 2^{-h_{in}}$ and $P_{low} = \frac{(1-P_{high})}{2^{n_{in}-1}}$.
$n = \min(n_{out}, nw)$.
$\psi = 2^{n_{in}-n} P_{low} + P_{high}$
$\mathrm{U} = 2^{n_{in}-n} + \sqrt{2\, n(2^{n_{in}-n})\ln(2)}$
$\omega = U \times P_{low}$
Return $-\log_2(\max(\psi, \omega))$

Chaining Conditioning Components

When there are more than one conditioning components, 90B considers the function as a single non-vetted conditioning component as a combination of each. Not possible to achieve full entropy.



Next update will *allow* chaining conditioning components, by calculating entropy estimate after each conditioning function. If the final conditioning function is vetted, it is possible to achieve full entropy.

Aligning NIST and BSI standards

BSI (Germany) also has similar standards:

- AIS 20: Functionality classes and evaluation methodology for deterministic random number generators
- AIS 31: Functionality classes and evaluation of physical random number generators

There are differences in the BSI’s and NIST’s validation process in terms of definitions, requirements, modeling and evaluation process.

Long term aim: Align NIST 90 Series and AIS 20, and AIS 31 by

- Understanding the source better using stochastic models (especially for physical sources).
- Differentiating between physical and non-physical sources
- Going from black box to a white box approach.
- Online health tests tailored to the characteristics of the design.

Conclusion

- Standard development on random number generation is an ongoing process.
- Standards/guidelines are useful, but they have limitations.
- A good understanding of the design is necessary to estimate entropy.

NIST GitHub page: https://github.com/usnistgov/SP800-90B_EntropyAssessment

Contact: rbg_comments@nist.gov