Random Bit Generation

Elaine Barker

NIST

September 11, 2012

Background

- Started in 1998 in ASC X9F1 (Financial Services sub-committee)
- Being published in ANSI as ANS X9.82 (4 parts)
- Being published by NIST as SP 800-90 (3 parts)

Subject	NIST SP 800-90	ASC X9.82
Overview and Basic Principles		Part 1
Entropy Sources	90B	Part 2
Deterministic Random Bit Generators (DRBGs)	90A	Part 3
RBG Constructions (DRBGs and NRBGs)	90C	Part 4

X9.82, Part 1 (Overview and Basic Principles

- Completed in 2006
- Need for RBGs – to generate keys, etc.
- Functional model: Entropy source + algorithm for generating bits
- Types of RBGs: DRBGs and NRBGs

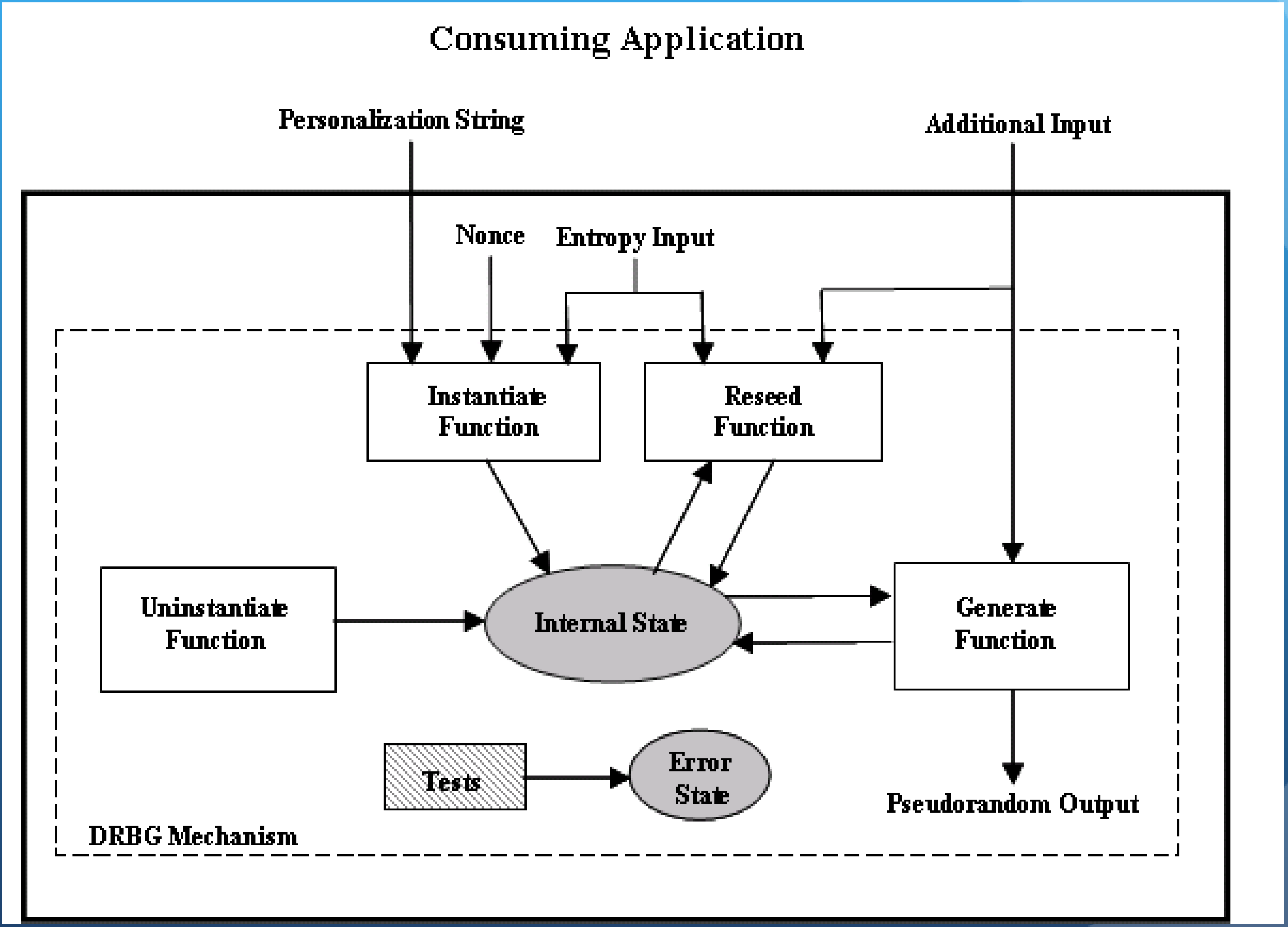
X9.82, Part 1 (contd.)

- Security properties
  - Security strengths: 112, 128, 192 and 256
  - Entropy: measure of unpredictability; defined with respect to an observer
  - Measure selected: min-entropy – worst-case measure
  - Backtracking resistance: given result at time *t*, cannot determine previous results
  - Prediction resistance: given result at time *t*, cannot predict next value(s)

SP 800-90 (A) and X9.82, Part 3: DRBG Mechanisms (i.e., algorithms)

- SP 800-90 completed in 2007; revised as SP 800-90A in 2012
- X9.82, Part 3 completed in 2007 (not yet revised)
- SP 800-90A contains 4 DRBG mechanisms (i.e., algorithms)
  - Hash_DRBG: hash function; not in X9.82, Part 3
  - HMAC_DRBG: HMAC (with a hash function)
  - CTR_DRBG: block-cipher-based
  - Dual_EC_DRBG: elliptic curves + hash function

SP 800-90A – Functional Model



SP 800-90A (contd.)

- Input
  - o Entropy input:
    - From an entropy source or RBG
    - Entropy/security strength as specified by an application
    - Input string ≥ security strength in length
  - o Additional input
  - o Nonce
- Internal state:
  - o Information for the DRBG – security strength, critical state values, etc.
  - o Internal state for each separate DRBG instance

SP 800-90A (cont.)

- Functions:
  - Instantiate – get initial entropy
  - Generate- get output
  - Reseed – get new entropy; adds to previous entropy
  - Uninstantiate – kill DRBG instance
  - Health testing
- DRBG boundaries: conceptual, can be distributed
- Functions are within cryptomodules; health testing function within each
- Secure channel required between distributed functions/cryptomodules

SP 800-90A (contd.)

- Security strengths supported: 112, 128, 224, 192, 256; depends on crypto component and entropy provided
- Instantiation: Seeds constructed from entropy input + nonce + (opt.) personalization string
- Reseed: New seed constructed from internal state + entropy + (opt.) additional input
- Backtracking resistance by design
- Prediction resistance by reseeding

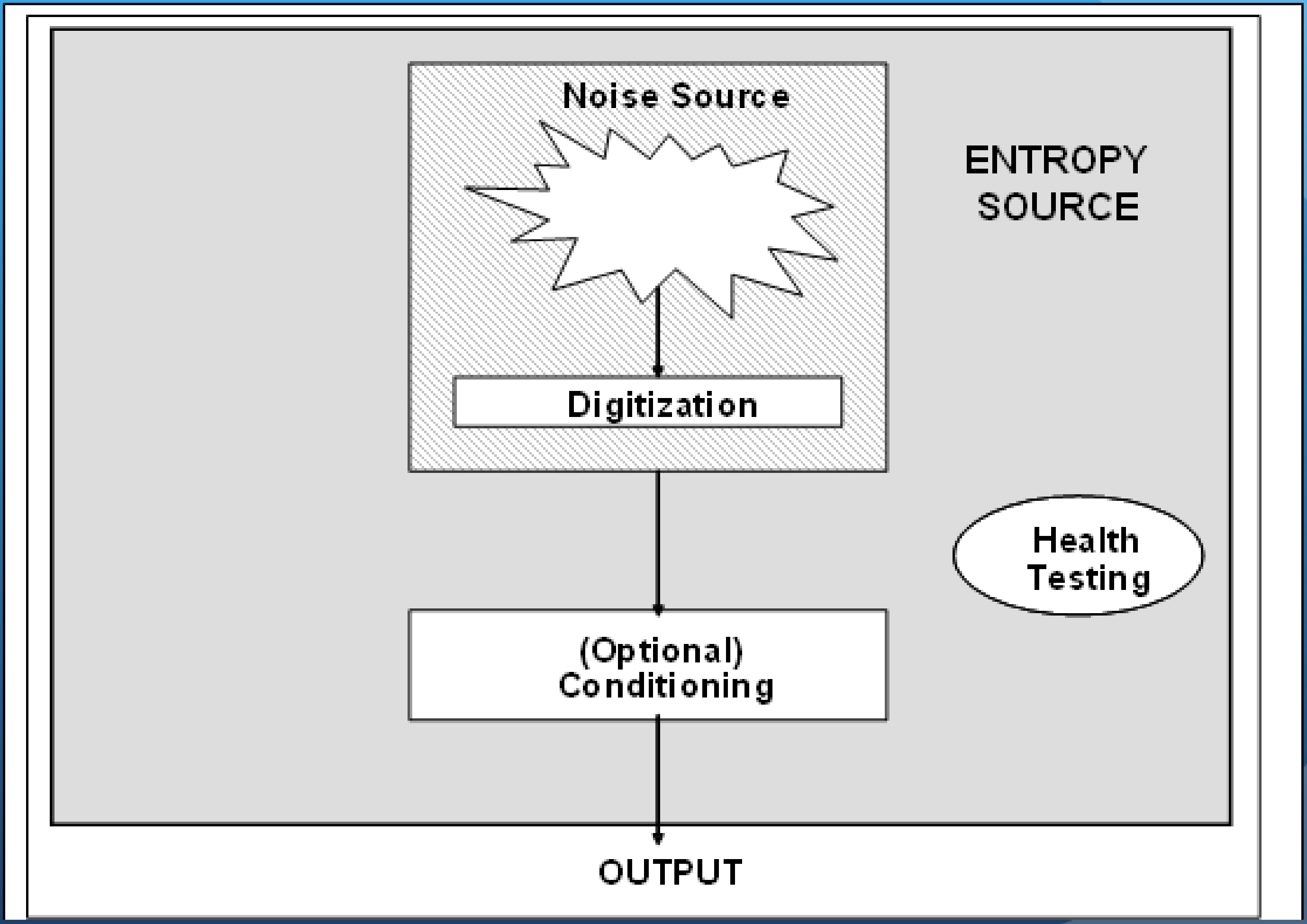
SP 800-90A (contd.)

- Validation testing
- Health testing
- Conversions (e.g., bits to integers)
- Security Considerations (when using Dual_EC_DRBG)
- Pseudocode examples
- DRBG mechanism selection

SP 800-90B and X9.82, Part 2: Entropy Sources

- Provided for public comment last week (HOORAY!!)
- Comments due on December 3, 2012
- Both contain design and health testing requirements (i.e., self tests)
- SP 800-90B also contains validation requirements
- Intent: Provide good design guidance, but don’t rule out good designs
- Acknowledge limited understanding; subject to change

SP 800-90B: Entropy Source Model



SP 800-90B: Development Requirements

- General requirements:
  - Documentation-documentation-documentation (intended to encourage developers to really THINK about and RESEARCH what they're doing):
    - Design (as a whole)
    - (Conceptual) security boundary
    - Range of expected operating conditions
  - Capable of FIPS 140 validation
  - Estimated entropy rate
  - Notification of health test failures
  - (Opt.) Multiple noise sources

Development Requirements (contd.)

- Full entropy requirements:
  - Essentially, each output bit has one bit of entropy (e.g., a 128-bit output has about 128 bits of entropy
  - Note: Full entropy output NOT REQUIRED
- Noise source requirements
  - Documentation:
    - Operation
    - Known conditions for malfunction
    - Protections from adversarial knowledge or influence
  - Exhibit probabilistic behavior
  - Amenable to testing
  - Severe degradation is detectable

Development Requirements (contd.)

- (Opt.) Conditioning component requirements
  - Documentation
    - If used, how is conditioning performed?
    - State and justify estimates of bias and entropy rates
    - How will variations in the noise source behavior affect the bias and entropy rate ?
  - Capable of health and validation testing
  - Approved conditioning components
    - MACs: CMAC, HMAC, CBC-MAC
    - Derivation functions in SP 800-90A
  - Other conditioning components allowed, but require more testing

Development Requirements (contd.)

- Health test requirements
  - General
    - Document tests and rationale for use
    - Test at startup and continuously
  - Noise source
    - Document known failure modes
    - Tests on digitized samples for variability
    - Bits successfully tested at startup may be used to produce output
    - Entropy rate is determined by samples passing continuous tests after startup
    - Entropy source shall be notified of detected failures
    - Place special emphasis on the detection of misbehavior near the boundary between the nominal operating environment and abnormal conditions.

Development Requirements (contd.)

- Health test requirements (contd.)
  - o Continuous testing
    - Two tests defined; alternative tests allowed if deemed equivalent
    - Additional tests OK
  - o Startup and on-demand testing
    - Run one cycle of continuous tests at startup (at a minimum)
    - Capability of on-demand test required, but not running the tests during operation

- Conditioning component health test requirements (if used)
  - o Document the tests
  - o Known-answer tests run during startup

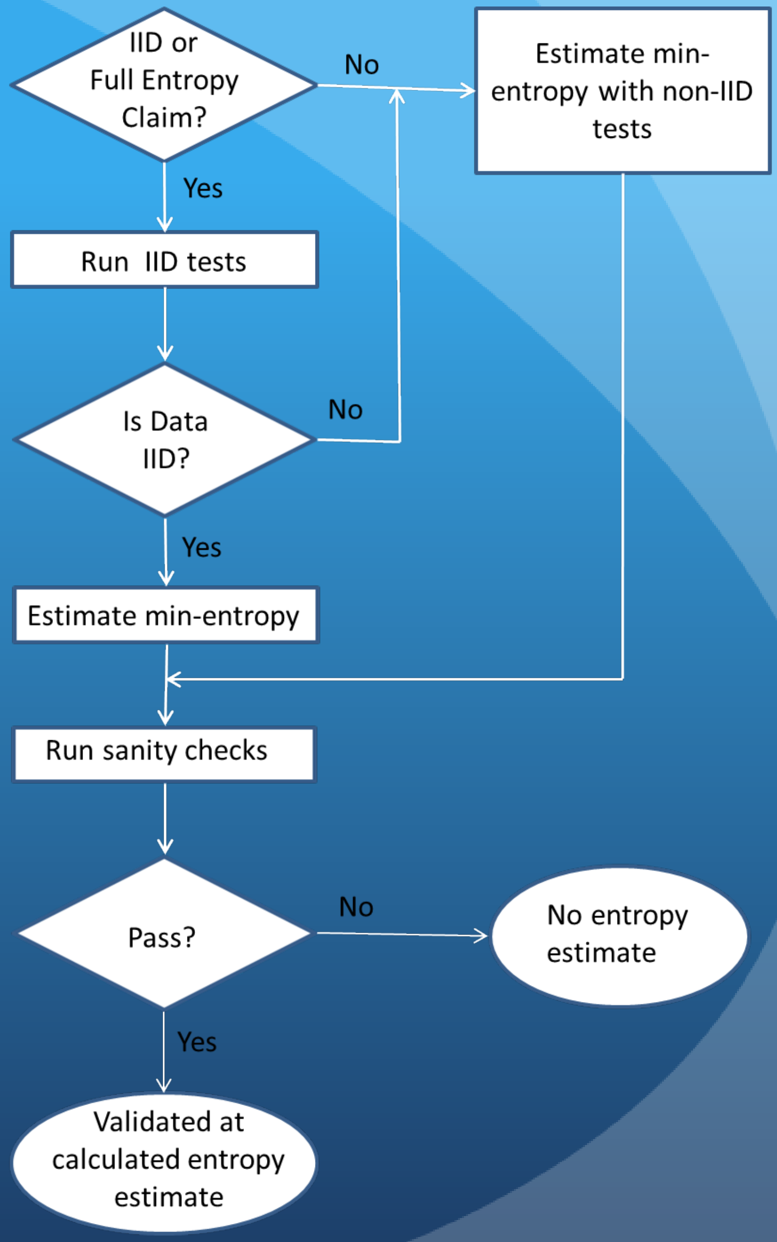
SP 800-90B: Validation Requirements

- Data collection
  - By the lab, or by the developer and witnessed by the lab
  - Raw digitized data from the noise source and conditioning component under normal operations
  - $\geq$ 1,000,000 consecutive samples
  - Not required from the conditioning component if an approved method is used
  - Ordered ranking if multiple bits in a sample

Validation Requirements (contd.)

- Validation testing
  - Tester will verify the continuous tests
  - Developer indicates whether noise source data is IID or not
  - Min-entropy estimate produced by the tests will define the validated min-entropy per sample
  - Full entropy only if IID data
- Documentation required describing the component(s) to be tested

SP 800-90B: Testing Strategy – Noise Source (to get assessed entropy from the noise source)



Testing Strategy (contd.)

- Conditioning component
  - **Approved** method
    - Method must be implemented correctly
    - Entropy source entropy estimate = $\min(outlen, entropy\_in)$
    - Full entropy only if $2 \times entropy\_in \geq outlen$.
  - Non-approved method
    - Run specified tests on the conditioned output to get min-entropy per conditioned output
    - Entropy source entropy estimate = min($noise\_source\_assessed$ $entropy$, min-$entropy\_per\_conditioned\_output$).

SP 800-90B: Tests

- Determining if the noise source data is IID
  - 6 shuffling tests
  - Chi Square test
  - Tests for independence (binary and non-binary data)
  - Tests for goodness of fit (binary and non-binary data)
- Tests for non-IID data: 5 tests
- Sanity checks: 2 tests

SP 800-90C and X9.82, Part 4: RBG Constructions

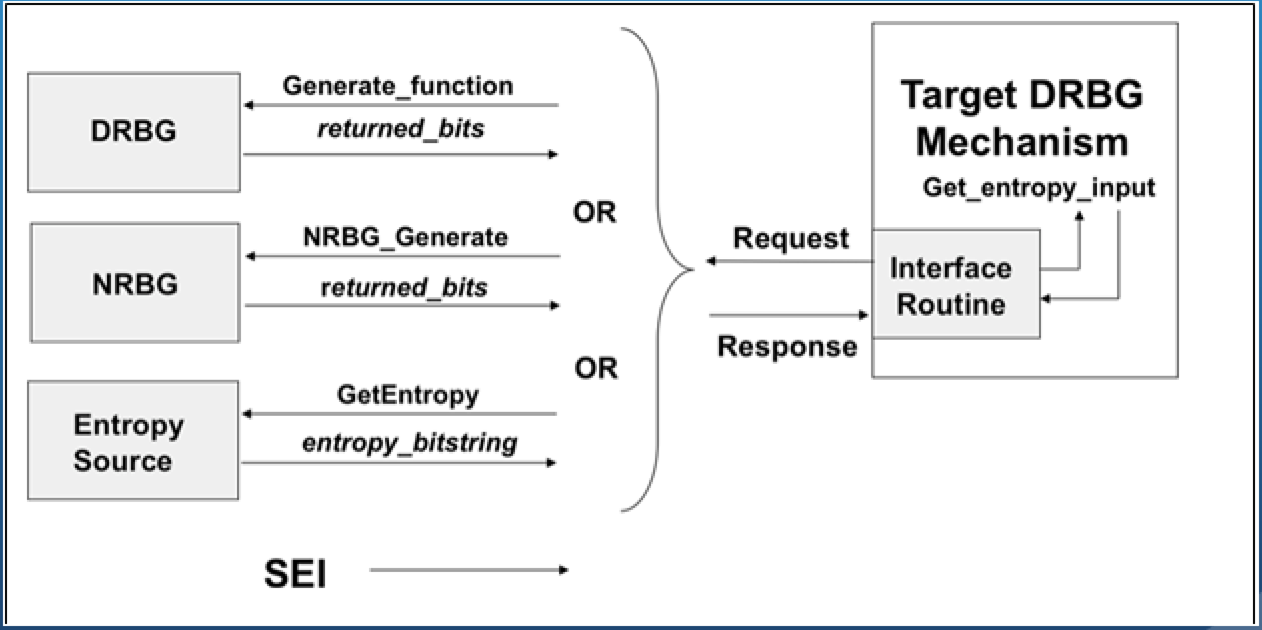
- Provided for public comment last week; comments due on December 3, 2012
- Purpose: To construct RBGs from **approved** entropy sources and DRBG mechanisms
  - o DRBGs (a.k.a. pseudorandom number generators)
  - o NRBGs (a.k.a. true random number generators)
- Extract of X9.82, Part 4; most constructions included

RBG Constructions (contd.)

- Concepts
  - (Conceptual) single and distributed boundaries
  - Live entropy sources: available when needed
  - Prediction resistance: obtain fresh entropy
  - Enhanced NRBG
  - Sources of entropy input (SEI)
    - Entropy source
    - RBG (DRBG or NRBG)
    - Chain of RBGs

DRBGs

- With or without a live entropy source
- Live entropy source allows prediction resistance

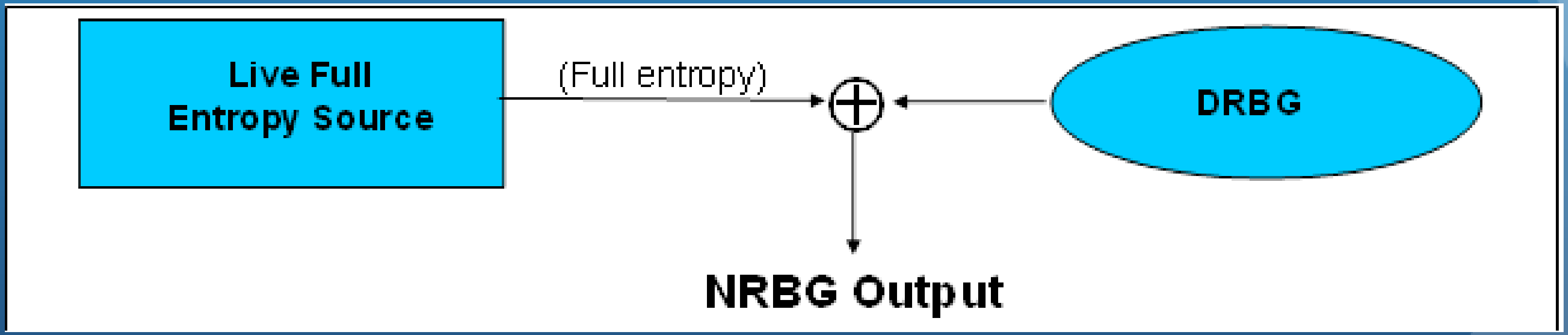


NRBGs

- Two constructions: XOR and Oversampling
- Live entropy source always required
- Approved DRBG mechanism required for enhanced NRBG
  - Instantiated at the highest security strength possible
  - Fallback if an undetected entropy source failure
  - Can be accessed directly (same or different instantiation)
- Full entropy output
- Backtracking and prediction resistance always provided

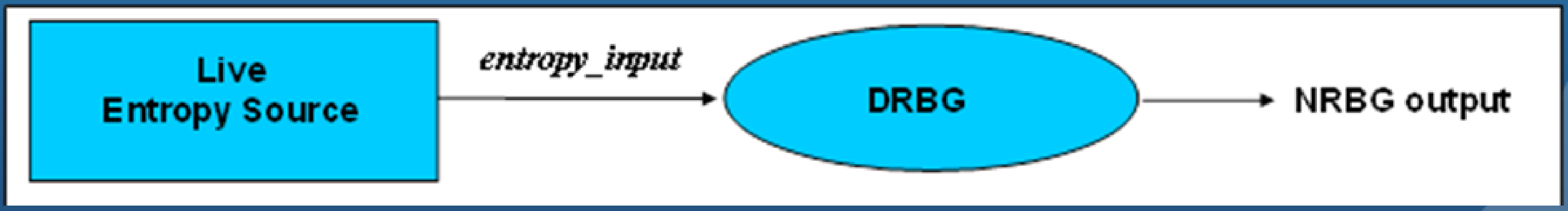
NRBGs: XOR Construction

- Requires full entropy source
- Entropy used to seed the DRBG not used for other purposes



NRBGs: Oversampling Construction

- Entropy source need not provide full entropy output
- Entropy_input = 2*n*; entropy output = *n*



RBG Constructions (contd.)

- Additional constructions
  - o Using an RBG as an SEI
  - o Using an entropy source as an SEI
- Testing
  - Health testing
  - Implementation validation
- RBG configurations
  - NRBGs: XOR and oversampling constructions
  - DRBGs: With and without a live entropy source
  - More complete examples in X9.82, Part 4

Issues

- How to test entropy sources in the CMVP labs?
- Are entropy source validation tests useful?
- Are additional approved conditioning components required?
- How would we specify a “basic” NRBG (i.e., without a DRBG mechanism) and maintain assurance of good output?