ANSI X9.82, Part 3 Deterministic Random Bit Generators (DRBGs)

Elaine Barker
NIST

July, 2003

Section by Section Review (Sections 1-5)

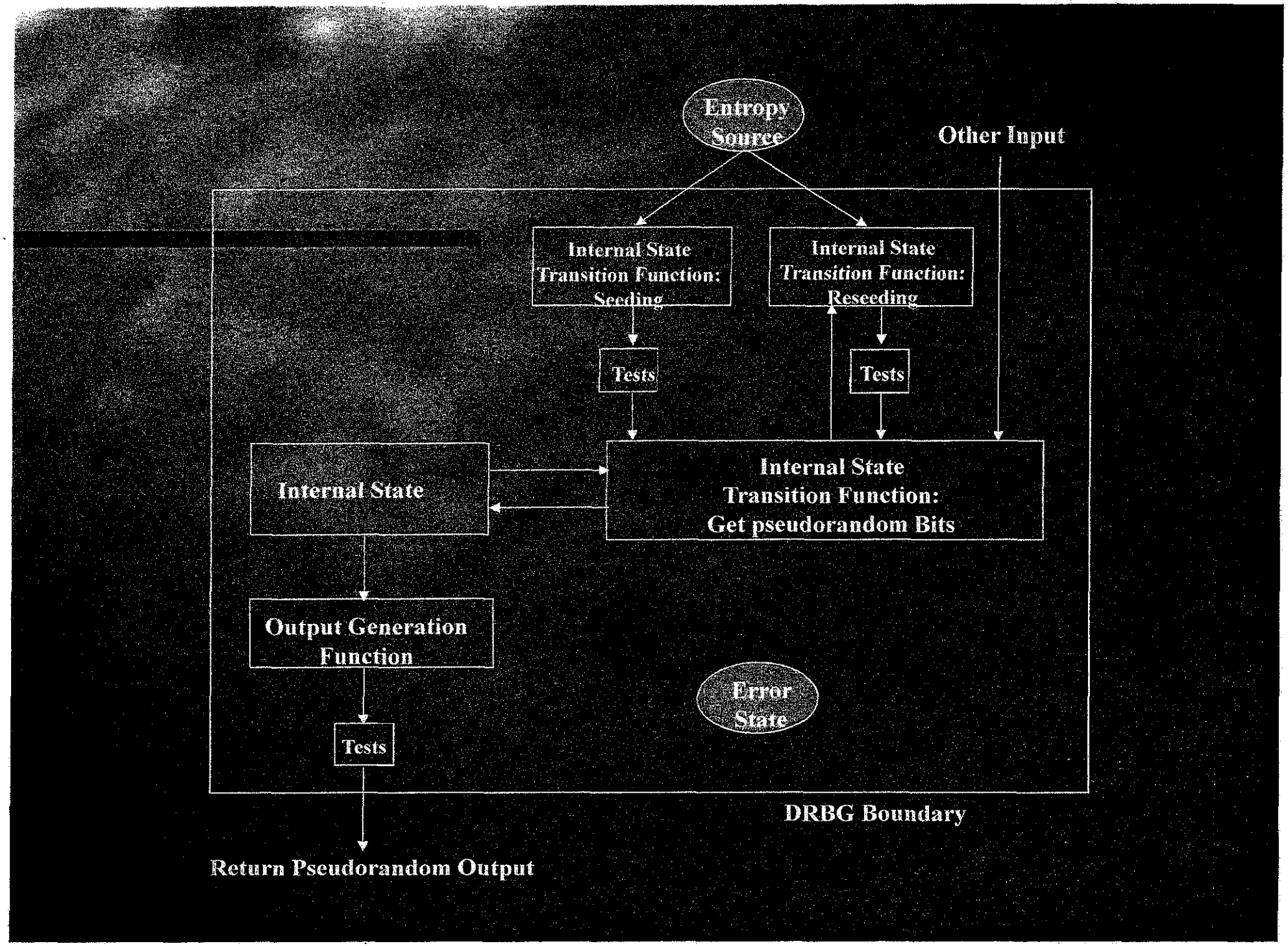
- Scope
- Conformance
- Normative References
  - How should we list the registry documents?
- Terms and Definitions
- Symbols and Abbreviated Terms
  - Only math notation is included; variables are identified in each DRBG section

Sections 6-7

- General Discussion and Organization
- Top-Level Security Objectives and Requirements
  - Inconsistencies between Part1 1 and 3
  - RBG Output
    - Essentially the same requirements and objectives as Part 1
  - RBG Properties and Operation
    - Part 1 testable req. 4 (re protection boundary) not in Part 3.
    - Part 1 testable req. 3 (re backward secrecy) and optional feature 2 (re forward secrecy) are combined as Part 3 objective 3.

Section 8

Functional Model and General Objectives and Requirements


Entropy Source
Other Input
Internal State Transition Function: Seeding
Internal State Transition Function: Reseeding
Tests
Tests
Internal State
Internal State Transition Function: Get pseudorandom Bits
Output Generation Function
Error State
Tests
DRBG Boundary
Return Pseudorandom Output

Section 8 (Contd.)

- Remove last sentence of testable req. 1 in Section 8.1 re verifying conditions?
- DRBG Components…
  - Entropy Source (relies on Part 2, NRBGs)
    - Seeds
    - Other user Input?
  - Other Input Information
    - Input parameters
    - Time-variant information
    - User input?
  - Internal State
    - Depends on the DRBG

Section 8 (Contd.)

DRBG Components... (contd.)

- Internal State Transition Functions
  - Initialization, Reseeding, & Generate output
- Output Generation Function
  - Req. 3 re test output and secret output separation: Stated as separating test and operational output; do we need to separate output that will be public from output that remains secret (I.e., separate DRBG instances)?
- Support Functions
  - No requirements in Part 1

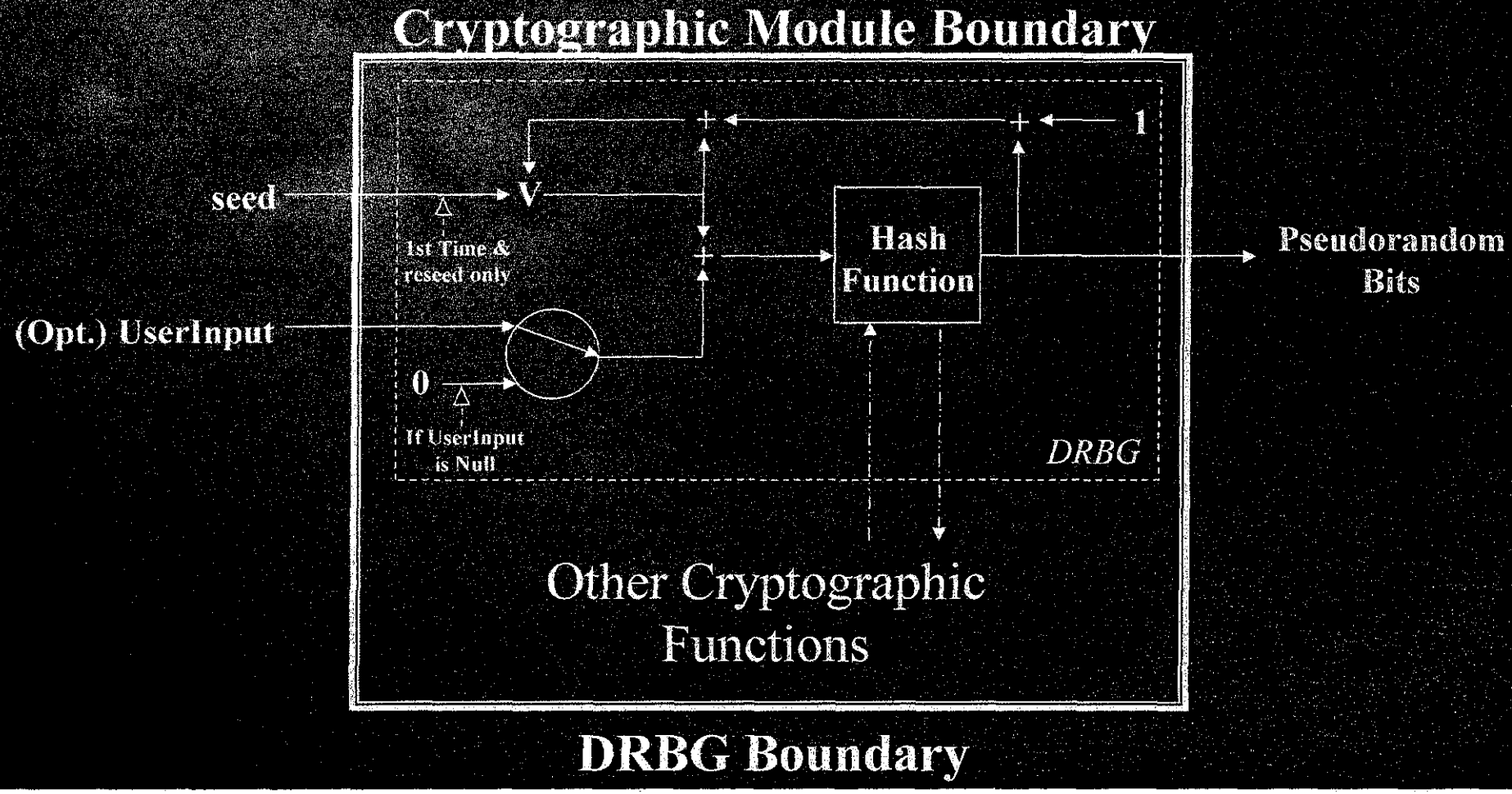
Section 9 – Additional Requirements

DRBG Boundary

- A DRBG **shall** be in a DRBG boundary
- The state an other inputs **shall not** be output from the boundary
- DRBG boundary **shall** be the same as the cryptomodule boundary or contained within the boundary

Additional Requirements (Contd.)

DRBG Boundary (Same as cryptomodule):



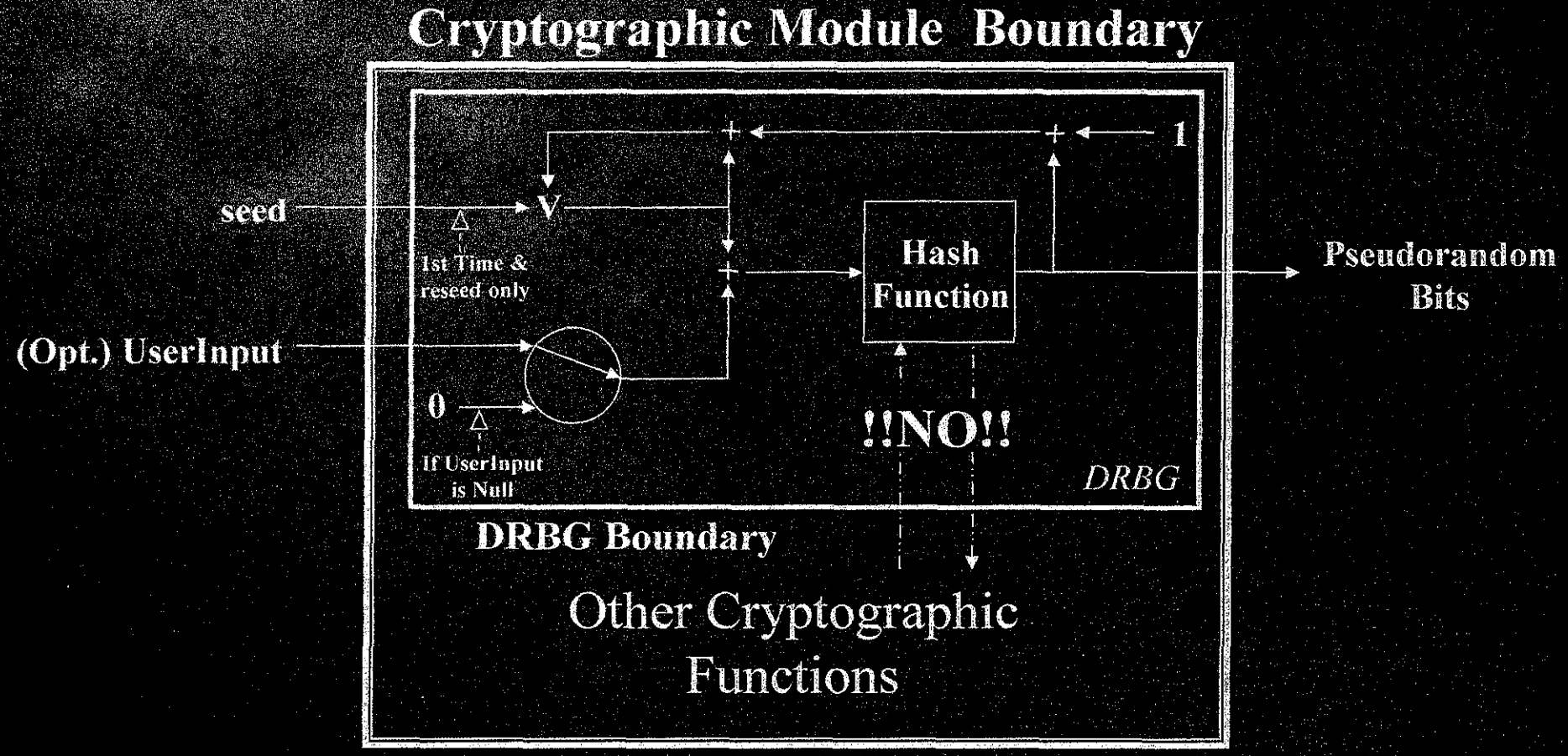
Additional Requirements (Contd.)

DRBG Boundary (Same as cryptomodule) (contd.):

- Basic crypto functions of the DRBG **may** be used by other crypto functions within the boundary
- DRBG internal state **shall not** be affected by the other crypto functions

Additional Requirements (Contd.)

- DRBG boundary within a cryptomodule boundary:



Additional Requirements (Contd.)

DRBG boundary within a cryptomodule boundary (contd.):

- DRBG components **shall not** be accessible by functions outside the DRBG boundary (but within the cryptomodule boundary)

Additional Requirements (Contd.)

- DRBG state
  - **Shall** be determined prior to producing DRBG output
  - Contents
    - DRBG purpose (usage class)
    - Values derived from the seed(s)
    - Keys (depending on the DRBG)
    - DRBG-specific information
    - Security strength provided
    - Transformation of seed or initial state
  - **Shall not** be accessible outside the DRBG
  - Transitions on demand; may transition in response to events or continuously

Additional Requirements (Contd.)

- Seeds
  - **Shall** be acquired prior to the generation of DRBG output
  - Seed use
    - **Shall** be kept secret
    - **Should not** be used to generate both secret and public values
    - **Shall not** be reused (including use for another instantiation)
    - Need to add: DRBG **shall not** provide output until a seed(s) is available and initial state is determined
  - Seed entropy
    - Needs to be reworded: entropy ≥ security strength

Additional Requirements (Contd.)

- Seeds (Contd.)
  - Seed size
    - Needs to be reworded: seed size $\geq$ 2 x security strength
  - Entropy source
    - An Approved NRBG, or an Approved DRBG seeded by an Approved NRBG
  - Seed Privacy
    - Handling consistent with security required for target data

Additional Requirements (Contd.)

- Reseeding
  - **Shall** have a specified seedlife
  - Check that two successive seeds aren't the same via a one-way transformation on the seed or initial state
- Seed separation
  - Different seeds **should** be used for different types of data (i.e., different instances)

Additional Requirements (Contd.)

- Keys
  - May be generated from seeds or provided from an external source
  - Key Use
    - DRBG **shall not** provide output until a key(s) is available
  - Key entropy
    - Need to rewrite: Entropy **shall** be ≥ security strength

Additional Requirements (Contd.)

- Keys (contd.)
  - Key size
    - Selected to support the desired security strength
  - Keys determined from a seed
    - Keys **shall** be independent of the rest of the initial state determined by the seed (Has this been accomplished in 3BlockCipher_DRBG?)
    - Depending on the DRBG
      - Initial state (including all keys) is determined from a single seed, or
      - Multiple seeds are used to determine different parts of the initial state

Additional Requirements (Contd.)

Keys (contd.)

- Keys provided from an external source
  - Keys **shall** have full entropy (How do we enforce this?)
  - Keys **shall** be protected in accordance with SP 800-57 (including outside the cryptomodule and DRBG boundary)
    - (Can we refer to a NIST special pub?)
- Rekeying
  - Keys **shall** have a finite keylife
- Key separation
  - DRBG keys **shall not** be used for any other purpose
  - Different DRBG instances **should** use different keys (consider problem when using externally provided keys)

Additional Requirements (Contd.)

User Input

- Optional
- Length and value are arbitrary

Forward and Backward Secrecy

- Achievable as observed from outside the DRBG boundary (black box view)
- From inside the DRBG boundary
  - Each DRBG provides backward secrecy (as viewed from inside the DRBG boundary) – Is this really true?
  - Some level of forward secrecy provided when new entropy provided

DRBGs Based on Hash Functions

- SHA1_Hash_DRBG
- Hash_DRBG
- Keyed_Hash_DRBG

SHA1_Hash_DRBG

- Revised from X9.30
- To be used only for 80 bits of security
- Need to rewrite: seed **shall** have entropy ≥ 80 bits
- State: purpose, *V*, initial value *t* for the hash function, seed entropy, seed transformation
- Specifications for initializing and reseeding the state, and generating output
- DRBG strength and attributes: ???
- Reseeding: How often?

Hash_DRBG

- Uses any Approved hash function
- Can meet different security levels
- Uses three hash functions: **Shall** be the same
- Need to rewrite: seed **shall** have entropy ≥ desired security strength
- State: purpose, *V*, *C*, counter, application-specific constant *t*, security strength, transformed seed

Hash_DRBG (Contd.)

- Specifications for initializing and reseeding the state, and generating output
  - Should the functions be written for the case where multiple hash functions are available?
- DRBG strength and attributes: ???
- Reseeding: How often?

Keyed_Hash_DRBG

- Based on Hash_DRBG with 1-3 keys
- Requires 1 -4 seeds: for V and each different internally generated key
- Need to rewrite: seeds **shall** have entropy ≥ desired security strength
- All keys provided externally, or all keys generated internally – reasonable?
  - Can we check the entropy of externally provided keys?
  - Should we allow some keys to be the same (see "*which_keys*" table on page 67)?

Keyed_Hash_DRBG (Contd.)

- State: purpose, *V*, *C*, counter, application-specific constant *t*, security strength, keys, transformed seed and keys
- Specifications for initializing and reseeding the state, and generating output
  - Should the functions be written for the case where multiple hash functions are available?
- DRBG strength and attributes: ???
- Reseeding and rekeying: How often?

DRBGs Based on Block Ciphers

3BlockCipher_DRBG

- Adaptation of the RNG in X9.17 and X9.31
- Uses any Approved block cipher (i.e., TDES or AES)
- Uses 3 instances of the same block cipher (including key sizes) – reasonable?

3BlockCipher_DRBG (Contd.)

- Requires 3 keys (key sets), one for each block cipher
  - 1 key TDES = DES: not allowed
  - 2 key TDES: 112 bits of key per key set
  - 3 key TDES: 168 bits of key per key set
  - AES-X: X bits for each key
- Keys (key sets) may all be the same or some (or all) may be different
- All keys **shall** be generated internally (see comment [ebb25]

3BlockCipher_DRBG (Contd.)

- Need to rewrite: seeds **shall** have entropy ≥ desired security strength
- *V* and all keys derived from a single seed
  - AES key wrap algorithm adapted as key derivation function (for now)
- State: purpose, *V*, counter, keys or key sets, security strength, transformed seed

3BlockCipher_DRBG (Contd.)

- Specifications for initializing and reseeding the state, and generating output
  - X9.17 use of DT has been changed
    - TDES: counter
    - AES: date/time || counter (allow date/time to be a constant?)
- DRBG strength and attributes: ???
- Reseeding and rekeying: How often?

DRBGs Based on Number Theoretic Problems

Dual_EC_DRBG

- Based on elliptic curve logarithm problem
- Requires 2 points P and Q: P is assigned to G (the generating point) – OK?
- Need to rewrite: seeds **shall** have entropy ≥ desired security strength

Dual_EC_DRBG (Contd.)

- State: purpose, *S*, prime modulus *p*, domain parameters (includes *P*), *Q*, security strength, transformed seed
- Specifications for initializing and reseeding the state, and generating output
- DRBG strength and attributes: ???
- Reseeding and rekeying: How often?

MS_DRBG

- Based on RSA problem of integer factorization
- Formatted DRBG specification still to be developed

Section 11 - Assurance

DRBG designs will be evaluated prior to X9.82 adoption (including statistical tests)

Implementation accuracy may be asserted or, preferably, may be validated.

Validation

- DRBG **shall** be within a FIPS 140-2 cryptomodule
- DRBG **shall** be designed to be tested
- Validation process to be specified in a TG?

Assurance (Contd.)

Operational Testing

- DRBG **shall** perform self tests at power-up, on demand and under various conditions
- DRBG output **shall** be inhibited during testing
- Results from known-answer tests **shall not** be used as operational DRBG output
- Enter an error state upon test failure
  - DRBG **shall not** output data
  - User intervention required to exit the error state

Assurances (Contd.)

Self tests

- Deterministic algorithm: perform known-answer test
- Software/firmware integrity test during power up (MACs, digital signatures, EDCs)
- Critical functions tested during power-up and on demand
- Software/firmware load test on all software and firmware that is loaded from an external source

Assurances (Contd.)

Self test (contd.)

- Manual key entry test
  - Perform an EDC (at least 16 bits) or provide duplicate entries (Is an EDC good enough?)
- Continuous RBG test (Is this appropriate for a DRBG?)