

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

July 20, 2004

Elaine Barker (NIST)

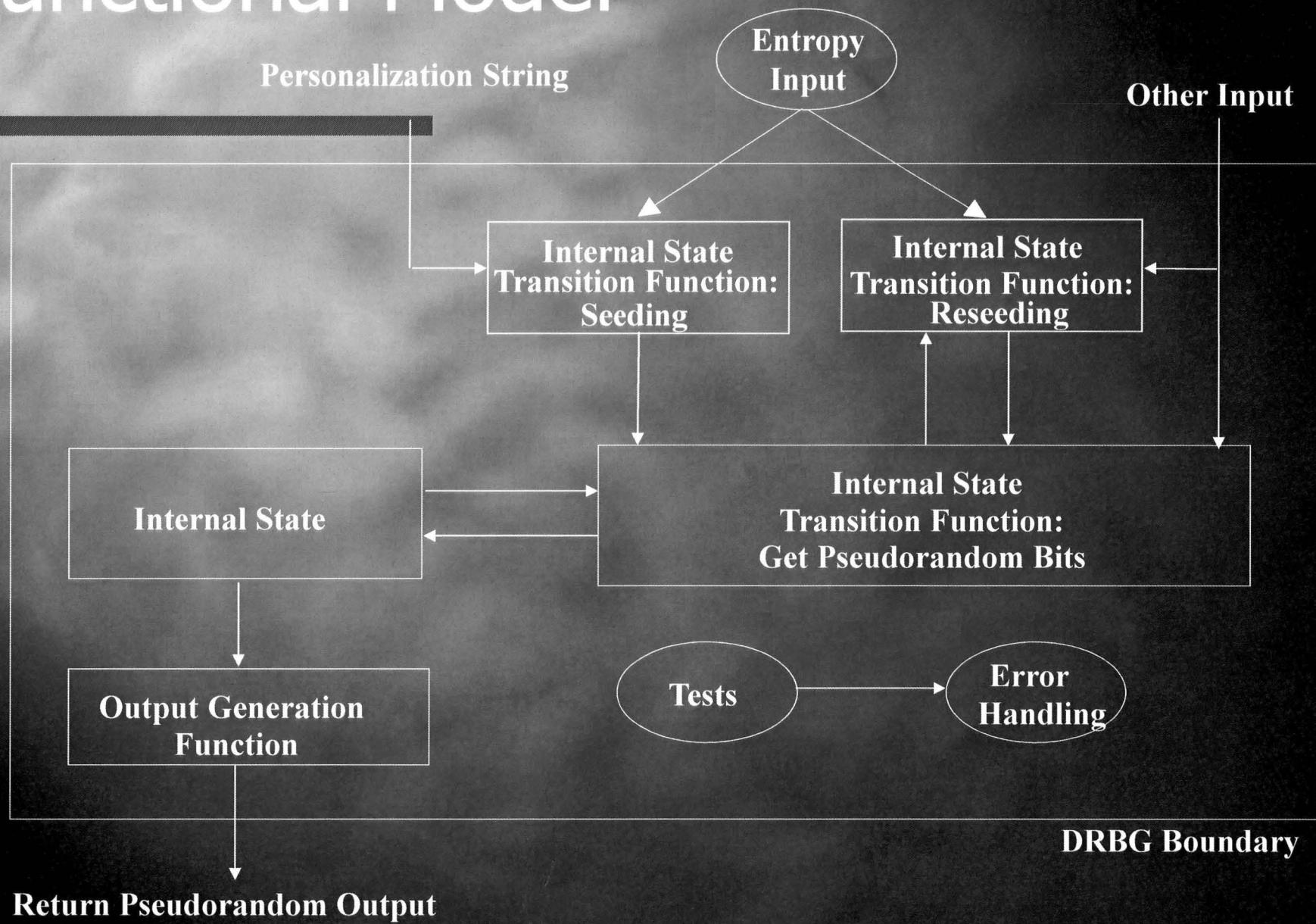
ebarker@nist.gov

301-975-2911

The Plan:

- Stroll through the document
- Hash-based and block-cipher based DRBGs (John Kelsey)
- Number theoretic DRBGs (Don Johnson)
- Address additional questions/issues

Functional Model



Functional Components (1)

■ Entropy Input

- Approved NRBG
- Approved DRBG (or DRBG chain)
- Other entropy source
 - Conditioned (translate to bits & remove bias)
 - Entropy assessed
 - Opt. derivation function

Functional Components (2)

- Other Inputs
 - Personalization string
 - Additional input
 - Counters, etc.
- Internal State
 - All the working parameters and other stored values
 - Some portion changes during each request

Functional Components (3)

■ Internal State Transition Function

- Instantiate
 - Generate bits
 - Reseed
 - Unstantiate
- 
- Fundamental

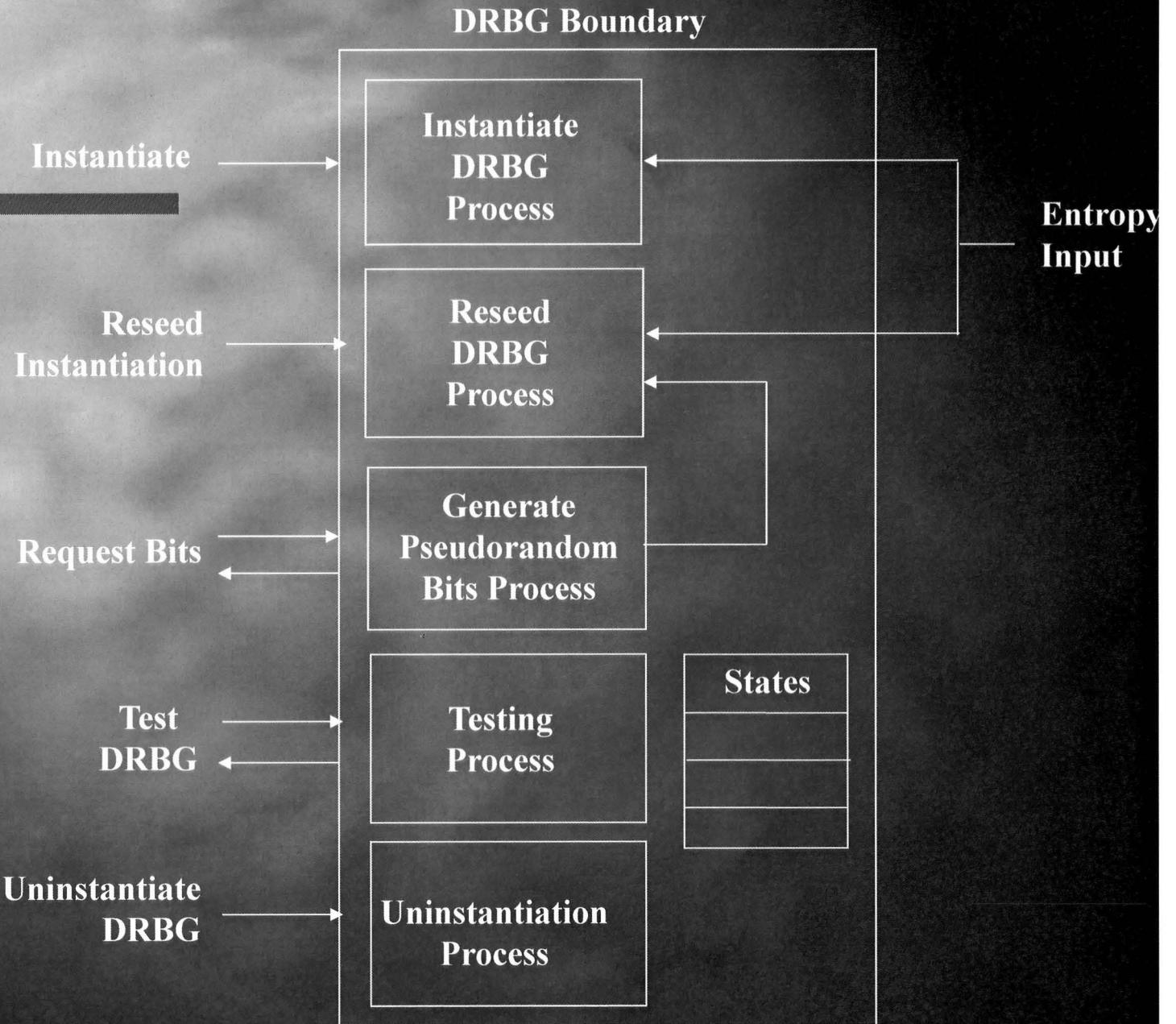
Functional Components (4)

- Output Generation Function
 - Selects bits from the internal state
 - Varies by DRBG
- Support Functions: Testing and Error Handling
 - Performs health checks
 - Aborts for catastrophic errors

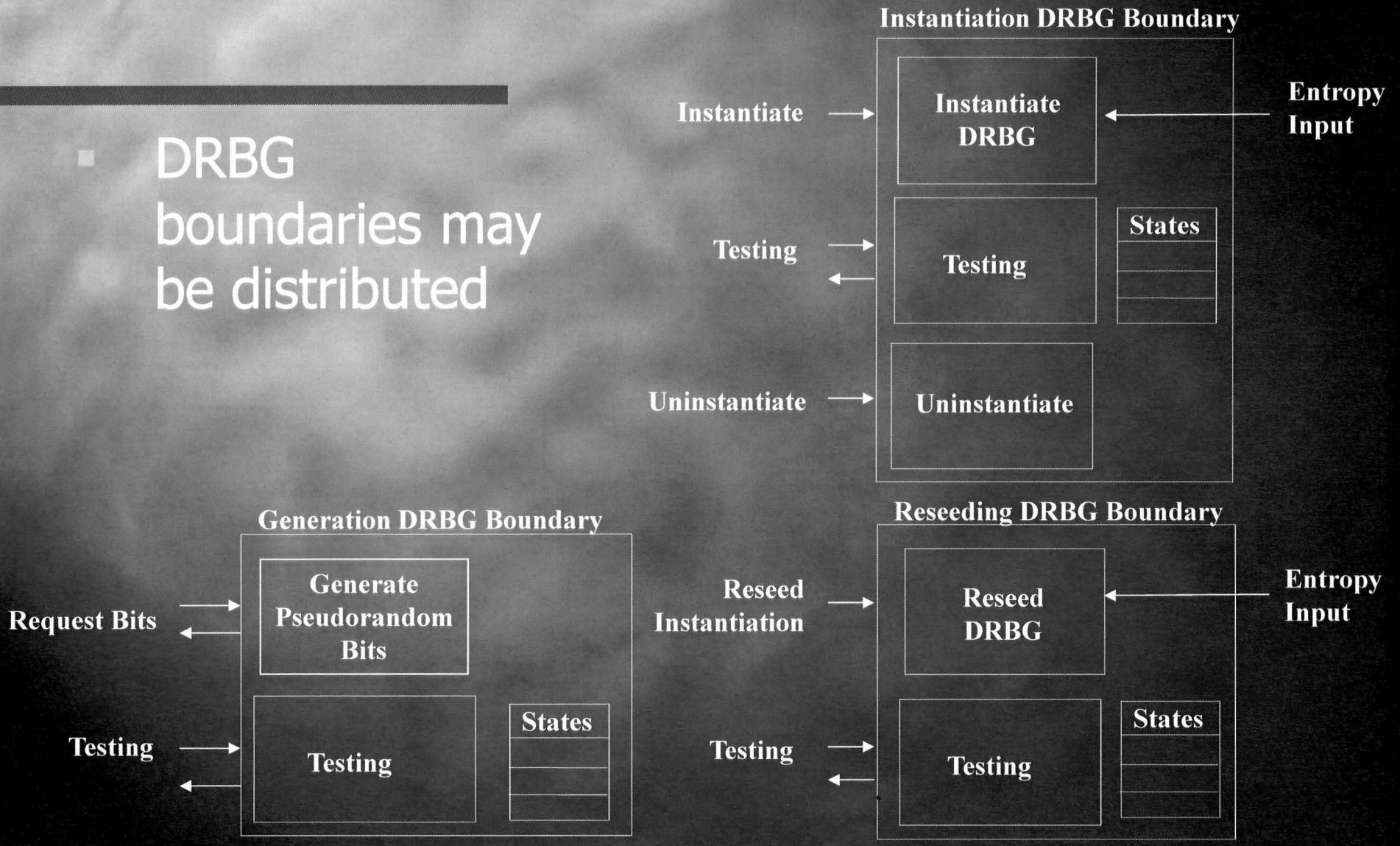
DRBG Boundary

- Physical or logical boundary
- Protects the DRBG internal state
 - Exists only within the boundary
 - Only affected per the DRBG spec.
 - State values remain within the boundary
- Other crypto functions
 - May reside within the boundary
 - May access the DRBG's crypto primitive(s), but not the state

- All DRBG processes may be in the same boundary



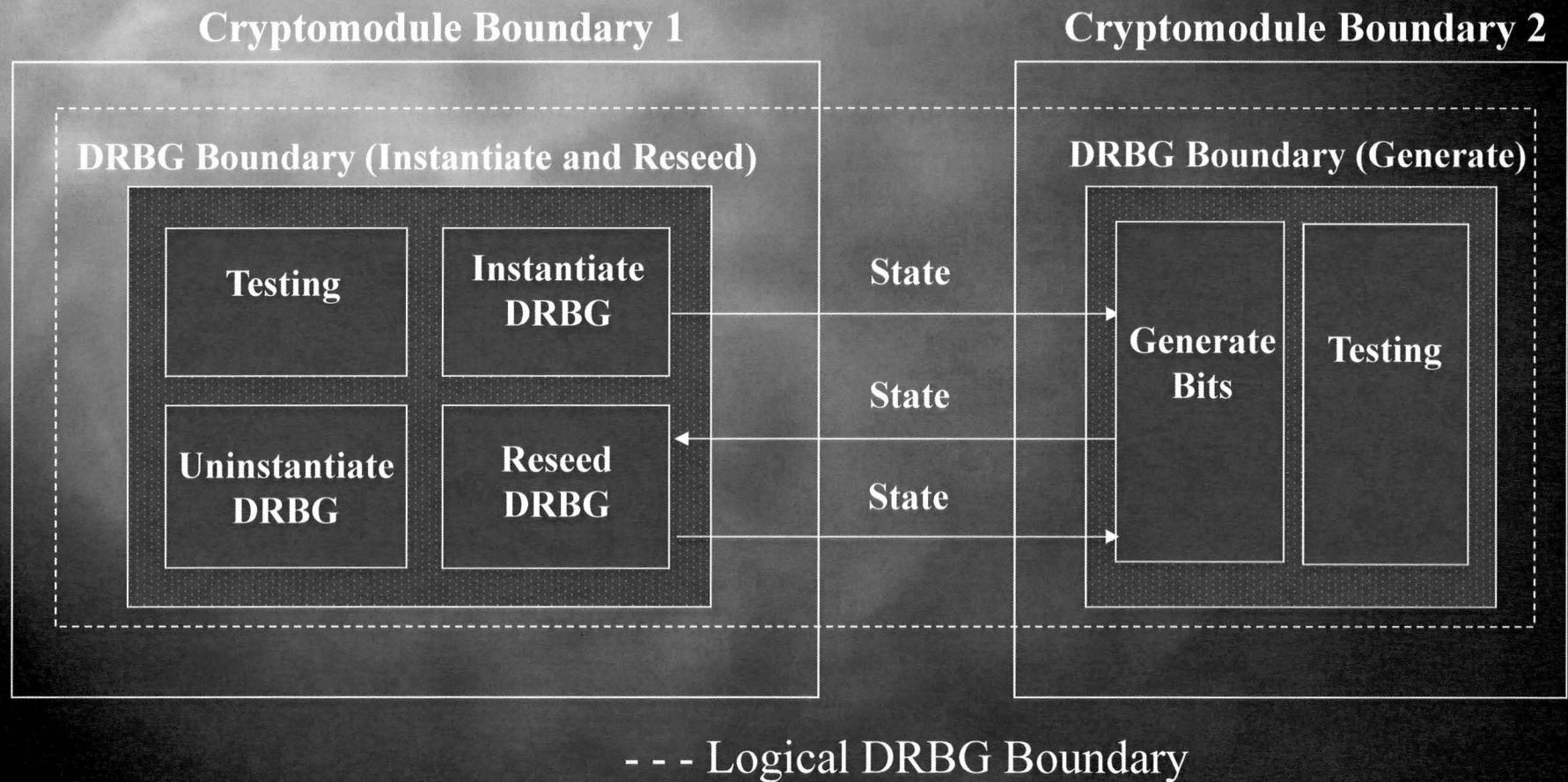
- DRBG boundaries may be distributed



DRBG Boundaries and Cryptographic Modules

- DRBG processes used by applications shall be in FIPS 140-2 cryptomodule boundaries
- A DRBG may be distributed across multiple cryptographic modules
- All DRBG processes for a given DRBG in a cryptomodule shall be in the same DRBG boundary
- Multiple DRBGs may be in the same or in different DRBG boundaries within the same cryptomodule

Example of A DRBG in Two Cryptomodules



DRBG Boundaries and Cryptographic Modules

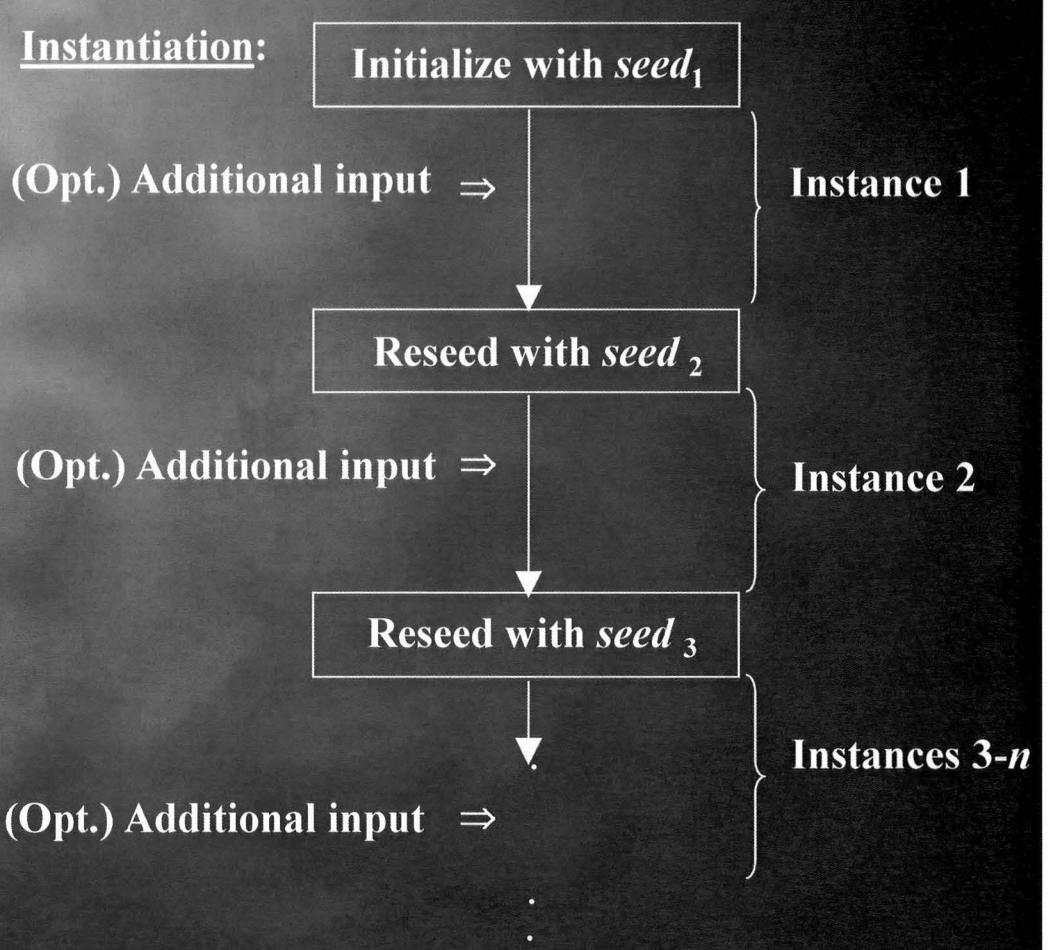
(contd.)

- FIPS 140-2 issues:
 - When a DRBG is distributed, the state needs to be transferred between DRBG boundaries within cryptomodule boundaries.
 - Entropy input source may be outside cryptomodule boundary
 - Manual and electronic entry
 - Different requirements for different FIPS 140-2 levels

Instantiation and the Internal State

- A DRBG is instantiated for one or more purposes
- Reseeding creates a new instance
- Additional input can be provided

Instantiation:

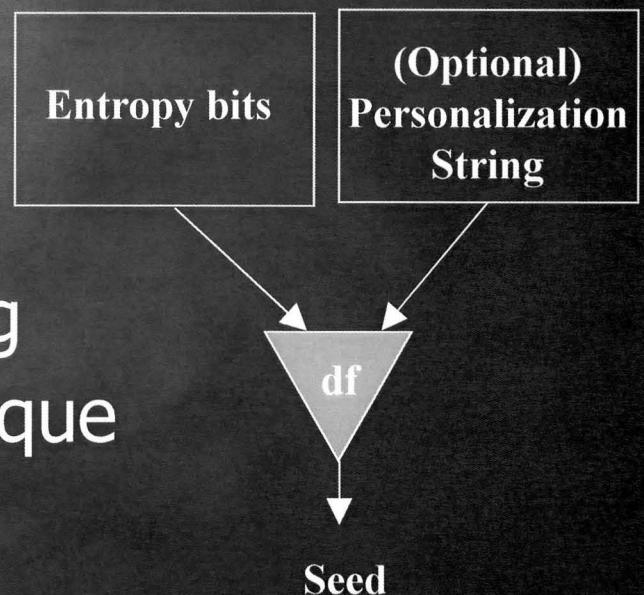


Instantiation and the Internal State (contd.)

- The internal state contains:
 - Values derived from the seed
 - DRBG-specific information
 - Prediction resistance flag
 - Security strength
 - (Opt.) Transformation of entropy input

Seeds (1)

- Acquired prior to the generation of pseudorandom output
- Used to instantiate the DRBG and initialize the state
- Seed construction:
 - Entropy input
 - (Opt.) personalization string
 - Goal: The seed shall be unique



Seeds (2)

- Entropy requirements
 - A seed shall contain sufficient entropy for the desired security level
 - Entropy shall be distributed across the seed
 - $entropy \geq \max(128, security_level)$
- Seed size: Depends on the DRBG and the security level

Seeds (3)

■ Entropy Input Source:

- Approved NRBG
- Approved DRBG (or chain of DRBGs) seeded by an NRBG
 - Entropy of higher DRBG \geq lower DRBG requirement
- Other source whose characteristics are known
- Need not be co-located with the instantiation process

Seeds (4)

- Entropy input and seed privacy
- Reseeding
 - Why? Reduce security risks; Recover from compromise
 - Replace seeds periodically
 - As specified
 - Check that entropy input is different
 - Combine new and old entropy to generate new seeds
 - Alternatively, reinstantiate

Seeds (5)

- Seed use
 - DRBGs used to generate secret and public information
 - Should use different instantiations
 - Entropy input and seeds shall remain secret
 - Different instantiations and instances shall use different seeds
 - No output until sufficient entropy is available

Seeds (6)

■ Seed separation

- DRBG seeds shall not be used for other purposes
- Recommend different seeds for different data types
- DRBG seed separation a cost/benefit decision

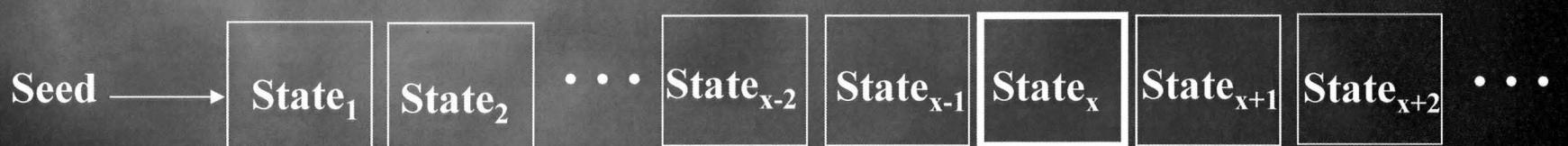
Other Input

- During instantiation, generation or reseeding
- Another source of entropy?
- Personalization string
- Additional_input

Backtracking Resistance

Backtracking resistance

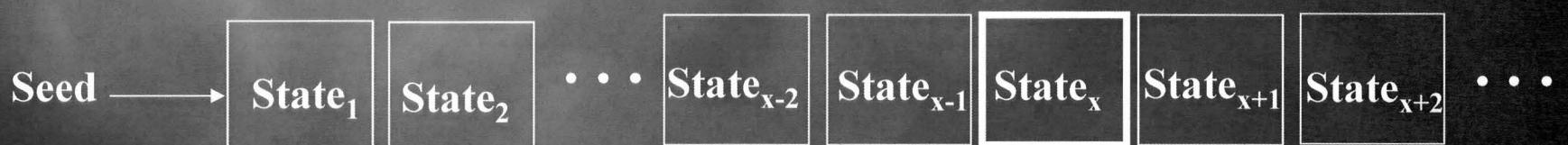
- Compromise of the state has no effect on the security of prior outputs
- Built into the DRBG design



Prediction Resistance

■ Prediction resistance

- Compromise of the state has no effect on the security of future outputs
- Obtain sufficient new entropy



Supported Security Strengths

- Support 80, 112, 128, 192 and 256 bits
- Determined during instantiation by the request and the crypto primitive used
- Entropy requirement must support all requests

Security Strength, Entropy and Seed Size

- Seed (i.e., entropy input) must have sufficient entropy
- $\min_entropy = \max(128, \text{requested_strength})$
- Entropy input size: a range of sizes
- Seed size depends on the DRBG and the security level

DRBG Purposes and States

- Recommend different instantiations for different purposes
- One internal state per instantiation
- DRBGs handle multiple states
 - Allow sufficient space for multiple states
 - Allow a state for health testing

State Table

Handle

1	V ₁ , C ₁ , reseed_counter ₁ , strength ₁ , prediction_resistance_flag ₁ , ...
2	V ₂ , C ₂ , reseed_counter ₂ , strength ₂ , prediction_resistance_flag ₂ , ...
<i>n</i>	Null, Null, 0, 0, 0, ...

Instantiating a DRBG (1)

- Instantiate process:

- Input:

- Requested strength
 - Prediction resistance request
 - (Opt.) Personalization string
 - DRBG-specific parameters
 - Mode

- Output:

- Status
 - State_handle

Instantiating a DRBG (2)

- Get_entropy function:

- Input:

- Minimum entropy
 - Minimum length
 - Maximum Length

- Output:

- Status
 - Entropy_input

Instantiating a DRBG (3)

■ Find_state_space function:

- Input:

- Mode

- Output:

- Status
 - State_handle

■ Derivation functions

- Hash_df

- Block_Cipher_df (Coming)

Reseeding (1)

■ How requested?

- Explicitly by an application
- When prediction resistance is requested
- At the end of the seedlife
- Triggered by external events

Reseeding (2)

■ Reseed process:

■ Input:

- State_handle
- (Opt.) Additional_input
- Mode

■ Output:

- Status

Generate Pseudorandom Bits (1)

- Generate bits for only one value
- Multiple requests may be used to construct a single value

Generate Pseudorandom Bits (2)

■ Generate process:

■ Input:

- State_handle
- Number of bits requested
- Strength to be provided
- (Opt.) Additional input
- (Opt.) Prediction resistance request
- Mode

■ Output:

- Status
- Pseudorandom bits

Removing a DRBG Instantiation

- Used to release an instantiation's state space
- Uninstantiate process:
 - Input:
 - State_handle
 - Output:
 - Status

Self-Testing (1)

- To obtain assurance that the implementation continues to operate correctly (health testing)
- Used during validation
- Test the DRBG processes within the DRBG boundary
- Strawman testing process provided

Self Testing (2)

- Test for correct results
- Test error handling
- Abort for failures

DRBGs

- Hash-based:
 - Hash_DRBG
 - HMAC_DRBG
 - KHF_DRBG
- Block cipher-based:
 - CTR_DRBG
 - OFB_DRBG
- Number theoretic
 - Dual_EC_DRBG
 - MS_DRBG

Assurance

- Designs have been evaluated
- Documentation shall be available
- Implementations may be validated
- Operational (health) tests shall be performed

Summary of Part 2 (DRBGs)

- DRBG and Crypomodule Boundaries
- The internal state and the seed must be protected
- Seeds must have sufficient entropy (in accordance with the security level)
- DRBG processes:
 - Required: Instantiate, Generate, Self-test
 - Optional: Reseed
- Assurance