# X9.82, Part 3 (DRBGs)

May 2004

Elaine Barker

# Changes

- **DRBG Boundary (8.2):** Separated the boundary requirement from the FIPS 140-2 requirement.
- **Model of DRBG Processes (8.3):** Provided discussions and diagrams for distributed DRBG processes.
- **DRBG INstantiation and the Internal State (8.4):** Usage class removed from the state; state pointer/handle now used to access a state.

# Changes (contd.)

- **Generation and Handling of Seeds (8.5.2):**
  - Derivation function not required if no personalization function, or if full entropy & no personalization string.
  - Additional discussion and a diagram on entropy input from a chain of DRBGs.
- **Other Input (8.7):**
  - Additional text on the use of a personalization string.

# Changes (contd.)

- **Prediction and Backtracking Resistance (8.8):** Additional clarifying text.
- **Instantiating a DRBG (9.5):**
  - Instantiation returns a pointer/handle to the new state
  - Mode parameter added for testing
  - Example of a Get_entropy function is included; includes a *mode* parameter for testing
  - Function added to find a state space

# Changes (contd.)

Reseeding:

- Added text about when reseeding might be done.
- Added *additional_input* parameter to provide additional input for reseeding
- Added a *mode* parameter to allow testing
- Uses a *state_pointer* to access the correct state

# Changes (contd.)

- **Generating Pseudorandom Bits Using a DRBG (9.7):**
  - Uses a *state_pointer* to access the correct state
  - *Additional_input* provided in the function call
- **Removing a DRBG Instantiation (9.8):**
  - Provides the ability to remove/delete an instantiation (e.g., after testing the instantiation process)

# Changes (contd.)

- Self-Testing of the DRBG (9.9):
  - Attempt to specify general known-answer testing for the DRBGs; each DRBG may have additional test code
  - Tests correct operation of each DRBG process implemented
  - Tests the error handling code

# Changes (contd.)

- **Hash-Based DRBGs (10):**
  - Two new hash-based DRBGs: KHF and HMAC
  - Table provided to indicate security strengths, and entropy and seed requirements
- **Hash_DRBG (10.1.2):**
  - Removed the application-specific constant *t*
  - Instantiation (10.1.2.3.3):
    - *seedlen* = *strength* + 64
    - *entropy_input* saved, rather than the *seed*
    - C defined differently: *C* = Hash (0x00 || *V*).

# Changes (contd.)

- **Hash_DRBG (contd.)**
  - **Reseeding (10.1.2.3.4):**
    - Seeding material now includes optional *additional_input*
    - Uses *seedlength* as stored in the state
    - Revised definition of *C* (as in instantiation)
  - **Generating Pseudorandom Bits (10.1.2.3.5):**
    - Uses *additional_input* as provided from the calling parameter
    - Prediction resistance request gets entropy rather than reseeding (counter not reset)
    - Revised update process: $V = (V + \text{Hash}(0x03 \,||\, V) + C + \textit{reseed\_counter}) \bmod 2^{seedlen}$.

# Changes (contd.)

- KHF_DRBG (10.1.3):
  - New hash-based DRBG
  - Uses KHF and Update internal functions
  - Updates two keys during each request for bits, as a minimum
  - Uses a hash function

# Changes (contd.)

HMAC_DRBG (10.1.4):

- New hash-based DRBG
- Uses an internal Update function
- Updates a key during each request for bits, as a minimum
- Uses HMAC

# Changes (contd.)

## Dual_EC_DRBG (10.3.2):

- Includes a table to aid selection of appropriate parameters (e.g., curve, hash function, etc.)
- Specifications include the same information and checks as the hash-based DRBGs.
- Corrections made to the code in accordance with comments
- Need to insert implementation advice as is provided for the hash-based DRBGs (if deemed useful)

# Changes (contd.)

**MS_DRBG (10.3.3):**

- Specifications include the same information and checks as the hash-based DRBGs.
- Corrections made to the code in accordance with comments
- Need to insert implementation advice as is provided for the hash-based DRBGs (if deemed useful)
- Need a definition of **Get_random_modulus** (...)
- Table for M-S parameters expanded to include number of hard bits and the suitable hash functions

# Changes (contd.)

- **Assurance (11):**
  - Inserted a section containing minimum documentation requirements
- **Annexes**
  - Reordered with normative annexes first
  - Added an annex on Implementation Considerations (B)
    - Implementation within a FIPS 140-2 boundary (B.1)
    - Entry of entropy input into a DRBG boundary (B.2)
    - Transfer of state information between boundaries (B.3)

# Stuff to be Added:

- DRBG selection discussion
- Additional security consideration text
- New block cipher-based DRBG?
- Revised block cipher-based derivation function

# Issues to Address

- **Functional Requirements:**
  - Should the detailed discussion of the functional requirements from Part 1 be in Section 7 or in an annex?
- **Testing:**
  - Is this an appropriate approach for known-answer testing?
  - How many pseudorandom bit requests to make and of what length(s)?
  - Should the error handling code be tested?

# Issues to Address (contd.)

Implementation considerations:

- Is it helpful to discuss what steps can be omitted under certain conditions?

# Additional topics:

- **Discussion of Comments**
  - Certicom (Don Johnson)
  - Pitney Bowes (Matt Compagne)
- **RNG Workshop**
  - July 19-22 at NIST
  - Contact ebarker@nist.gov to register
  - Workshop information available at http://csrc.nist.gov/CryptoToolkit/tkrng.html
  - "Excerpts" to be posted about June 21st