X9.82, Part 3 Discussions during the teleconference:

Next draft to include:

1. 2 new hash-based DRBGs: KHF_DRBG and HMAC_DRBG
2. Revised Hash_DRBG to:
   - Provide backtracking resistance,
   - Eliminate the required application-specific constant t; such information can be provided in a personalization string
3. Possibly a block cipher-based DRBG
4. An un-instantiate process
5. Possibly an abort process for test failures and entropy input source failures

Operational testing specifications

   - Will test each process (instantiation, generation, reseeding)
   - Test diagrams will be removed, since they don't adequately show the testing process

6. Implementation considerations with respect to FIPS 140-2 cryptomodules
7. Discussion on DRBG selection
8. Expanded security considerations annex
9. Add Find_state_space function
10. Add Uninstantiate process to Hash_DRBG
11. Generic testing in Section 9
12. Specify the string lengths to be tested for each DRBG
13. Modify Hash_DRBG per John's suggestions
14. Usage_class -> state_pointer
15. Remove test diagrams for each DRBG
16. Non-secret input -> additional input
17. Section discussing why each DRBG should be chosen.
18. Entropy input source entropy >= requested strength requirements
19. Define Abort error handling function.
20. Provide a discussion of personalization strings.

Part 3 Issues (Mostly)

1. Need to finalize the functional requirements.
2. Section 7.2.2, General Functional Requirements, requirement 1: A requirement on the validator doesn't seem to be a functional requirement.
3. Documentation requirements in Section 7 don't seem to be functional requirements. Have a separate section on documentation requirements?

   Look at all the documentation requirements. Are they appropriate for DRBGs? Do they belong in Section 7 at all?
4. Section 7.2.5, Internal State: Does the distinction between the various information in the internal state need to be made (i.e., administrative portion vs. working portion)? This seems to serve no useful purpose (at present), so just seems to make the state more complicated.
5. Section 7.2.7, Output Generation Function, requirement 3: Should we require the state to change whenever output is produced, or can we just require that the same bits from the internal state shall not be reused?
6. Section 8.3, para. 2, last sentence: Should we really require that the other input be unguessable? We're no longer providing additional input. Other input is now the personalization string (part of the seed construction) and other DRBG initialization stuff.
7. Section 8.4, Seeds, item 6, para. 3, 2nd sentence: Should prediction resistance be mentioned here?
8. Section 8.7, Backtracking and Prediction Resistance: How do I get the indistinguishability concept in here?
9. We need to thoroughly discuss the conceptual API in Part 1. What needs to be included there, and what is really internal to the RBG?
10. Section 9.6.1: Does the instantiation call for a DRBG really need a full entropy flag?
11. Section 9.6.1: Can the requested strength parameter be left out if an implementation only supports one security level? An application would need to be aware of the DRBGs implementation so as not to assume a greater strength than the DRBG can provide?

    The same question arises for other parameters (e.g., usage class, prediction resistance flag, personalization string, etc.)
12. Section 9.6.2, Request for Entropy: Should the *min_entropy*, *min_length* and *max_length* parameters be allowed to be missing when an implementation is intended to always use a single value? Can the *min_length* and *max_length* parameters be combined when they are intended to be the same? See the para. under the numbered list.
13. Section 9.6.4.3: What do we want to use as the block cipher derivation function? We are currently specifying the AES key wrap for both TDEA and AES.

14. Section 9.7, Reseeding: Does the conceptual API in Part 12 need a reseeding function? Should teh usage class be allowed to be omitted when an implementation only has one usage class?

15. Section 9.8, Generating Bits: The order of the input parameters is different in Part 1. Which should change?

16. Section 9.8: Part 1 needs to add an optional additional input parameter. Does Part 3 need to handle the full entropy flag?

    Can some of the input parameters be omitted in an implementation always handles only a single value for the parameter?

17. Section 9.9, Inserting additional entropy between requests: Do we want to include this process?

    Para. 1, last sentence: Should this be incorporated (i.e., if insufficient entropy is available, update anyway or don't update).

    Can some input parameters be omitted?

18. Section 10.1.2.1, Table 1: Are the security strengths in this table correct? Are the seed length ranges correct?

    Need to correct the DRBG to provide backtracking resistance.

    Need to provide more guidance on values for *t* for various applications.

19. Section 10.1.2.3.2: Should the seedlen be max (strength + 64, outlen) or be fixed for each hash function (e.g., SHA-1 = 192, SHA-224 = 256, SHA-256 = 320, SHA-384 = 384, SHA-512 = 512)?

20. Hash_DRBG: Are the following processes handled appropriately?

    Instantiation, including use of the personalization string, use of the derivation function, and transforming the seed material?

    Reseeding, including combining the old entropy with the new entropy bits.

    Generating random bits, including the handling of the prediction resistance flag, updating when the maximum number of updates is reached.

    Adding entropy to the Hash_DRBG: see issue 17 above.

    Should discussions be included as to which steps can be omitted under what conditions (currently included in the specs.)?

    What needs to be discussed re the generator strength and attributes? What is the suggested reseeding limit (this would be the value of *max_updates*)?

Questions re the DRBG Self Testing Process
(Using Hash_DRBG as an example)

1. Should we test all processes implemented (i.e., instantiation, reseeding and generating bits)?
2. If instantiation is tested:
   - Should we use a special test usage class, or test each usage class implemented? Using a special test usage class would be simpler, but using the real ones would test that the proper value of *t* is used and that the initial state is set properly.
   - Should we test each of the security strengths implemented?
   - Should a special test flag be added to the **Get_entropy (...)** call? This would allow the insertion of a fixed value. Getting sufficient entropy to handle all multiple variations of the test would be tough. Does the DRBG need to test the entropy source?
3. If the reseeding process is tested, many of the same issues mentioned above apply.
4. For the Hash_DRBG (...) process:
   - Should different usage classes be tested?
   - Should different numbers of requested bits be used?
   - Should different strengths be tested (as appropriate for the implementation)?
   - What should we do about testing the prediction resistance aspect?
   - Should we test the reseeding process when the counter reaches the maximum number of updates?
5. Should subsets of these tests be performed, depending on whether the testing is done at power up or on demand?
6. Other issues?

10.1 Deterministic RBGs Based on Hash Functions

10.1.1 Discussion

The hash-based DRBGs specified in this Standard have been designed to use any Approved hash function and may be used by applications requiring various levels of security, providing that the appropriate hash function is used and sufficient entropy is obtained for the seed. The following are provided as DRBGs based on hash functions:

1. The **Hash_DRBG (...)** specified in Section 10.1.2.
2. The **KHF_DRBG (...)** specified in Section 10.1.3.
3. The **HMAC_DRBG (...)** specified in Section 10.1.4.

The maximum security level that could be supported by each hash function when used in a DRBG is equal to the number of bits in the hash function output block. However, this Standard supports only five security levels for DRBGs: 80, 112, 128, 192, and 256. Table 1 specifies the security strengths and entropy and seed length requirements that **shall** be used for each Approved hash function.

Table1: Security Strength, Entropy and Seed Length requirement for Each Hash Function

Hash Function	Security Strength	Required Minimum Entropy	Seed Length for Hash_DRBG	Seed Length for KHF_DRBG & HMAC_DRBG
SHA-1	80, 112	128	160-512	160 - 2^{32}
	128	128	192-512	160 - 2^{32}
SHA-224	80, 112, 128	128	224-512	224 - 2^{32}
	192	192	256-512	224 - 2^{32}
SHA-256	80, 112, 128	128	256-512	256 - 2^{32}
	192	192	256-512	256 - 2^{32}
	256	256	320-512	256 - 2^{32}
SHA-384	80, 112, 128	128	384-1024	384 - 2^{32}
	192	192	384-1024	384 - 2^{32}
	256	256	384-1024	384 - 2^{32}
SHA-512	80, 112, 128	128	512-1024	512 - 2^{32}
	192	192	512-1024	512 - 2^{32}
	256	256	512-1024	512 - 2^{32}