X9.82 Discussion Issues for August 11, 2004

General Issues:

1. Re mutually pointing validation requirements: use "should".
2. Move all definitions into Part 1?
3. Consider offering a draft for interim use.
4. How does prediction resistance relate to the case where entropy is dribbled in?
5. Component interaction, particularly between entropy sources.
6. Restructuring: Part 2 to discuss basic NRBG/conditioned entropy source/entropy source? Construction of an NRBG other than in Part 2? Drop the distinction between DRBGs and NRBGs?
7. Continual public review needed by academics and industry. Have another workshop? Take the document into another forum?
8. Include a technical annex of the decisions made (security considerations?).
9. Remove the 80 (and possibly the 112) bit security level?

Part 1 Issues:

1. Include additional examples (e.g., Monte Carlo)?
2. Include additional text from Wolfgang Killman re different types of entropy?
3. Can min_entropy be determined if there is no independence? More guidance in the text for when correlation exists.

<u>Part 1 agreed upon changes</u>:

1. Enhanced NRBGs are "computationally bounded", rather than having an infinite security level. Basic NRBGs are an informationally secure source of random bits. DRBGs are computationally secure.
2. Include limits in the definition of "statistically unique".
3. Mention "repeatable random".
4. Re-emphasize dependencies.
5. Define entropy with respect to someone's knowledge.
6. Backtracking and prediction resistance: discuss with respect to requests as well as states.
7. Explicitly say that, for DRBGs and enhanced NRBGs, entropy is precious (recycle entropy).
8. Characterize mixing functions for preserving or accumulating entropy.
9. The OGF is used to hide the internal state. OGF slide: didn't like "sole job".
10. Add a DRBG with a continuous reseed capability (see slide 69).
11. Define "random number" in the context of cryptographic use; something about an independent identical distribution for a distributed random process.
12. Check the use of "seed"?
13. Table in 12.5: Assume that there is an entropy source somewhere (DRBGs).
14. Model: Show the OGF updating the state.

Part 2 Issues:

1. Include a discussion of NRBGs that are in the literature?
2. Discuss flawed entropy sources?

3. Are there any mutual min-entropy issues that should be included?
4. Sample at a higher rate to test the entropy sources? Fast sampling may be preferable for health testing. Is anything different if the entropy rate is high?
5. Specify requirements for conditioning functions?
6. Define "conditioned entropy source" (or is this "entropy input" of a "basic NRBG")?
7. Use lots of independent entropy sources? Evaluate at least one? Test entropy sources, if possible.
8. Allow the testing of the output of a conditioning function? May provide a "sense" of the entropy source randomness or bias.
9. Registration procedure for entropy sources and techniques?
10. Look into entropy sources whose patents have expired (e.g., 16 astable multi-vibrators)?
11. Describe what happens when the reseed rate is changed?
12. Prediction resistance using a pool: Use the entire pool when prediction resistance is requested (providing that sufficient entropy is present)? Queue up bits inside/outside the DRBG? Don't use bits available prior to a prediction resistance request?
13. Set appropriate P-values for health testing? Decision based on the number of entropy sources?
14. Allow/require multiple layers of testing?
15. Specify the probability of a failure?
16. Differentiate between start-up (instantiation) and power-on (e.g., card insertion) tests? Test before outputting any random values since start-up or the previous power-on, as opposed to testing immediately upon start-up or power-on? Only test when it will be used? May be platform specific. Consider non-operational vs. operational tests.
17. Operational tests: at power-on or after reset? Whether or not state is lost? When recovering from hibernation?
18. Manufacturer tests (tests RNG and program): at installation or first birthday.
19. Require qualification tests for entropy sources?
20. Low false positive rate. If $P = 10^{-4}$, retry 3 times? Use adaptive tests (layers of tests: health tests, followed by sick tests if there are failures)? Finish as soon as you pass. establish a final error point (absolute limit).
21. Known answer test failures need not be retested.
22. For enhanced NRBGs: if the entropy source fails, but the DRBG is still OK, the decision to continue depends on application requirements?
23. Condition to fewer bits?

Part 2 agreed upon changes:

1. For assessment: document and analysis is required, dynamic assessment is OK.
2. Include discussions of common failure modes.
3. Specify specific entropy sources. Propose/standardize one or more entropy sources and test sets.
4. Specify a low-failure rate for continuous tests.

5. For an enhanced NRBG: DRBG seeding is done at installation (before normal operation).

Part 3 Issues: [Someone needs to add issues and changes for DRBGs, since I was not taking notes]

1. Parameterize the security assumption, rather than using 2^{64}.
2. Add/include other DRBG submissions?
3. Use a different word for "instance" (e.g., seed period).
4. Use a different words/phrases for internal state and "state" as a critical value of the internal state.
5. Do we want to continue to require backtracking resistance?
6. Is there *outlen* or *outlen*/2 bits of security for a hash-based DRBG?
7. Use different counters for the block cipher derivation function?
8. Should any of the DRBGs be removed (e.g., KHF_DRBG)?
9. Clean up the Hash_DRBG, including updating the full *seedlen* bits? We don't have much time.
10. Dual_EC_DRBG: Truncate more bits and reseed less frequently? Are there correlations among bits? Allow a conditioned entropy source instead of a hash derivation function?
11. Straighten out the meanings of "entropy source" vs. "entropy input" vs. "conditioned entropy source". Then use the proper term consistently.
12. For RSA: Use only 1 bit instead of 10-11 bits?
13. Include Blum-Blum-Shub?
14. DRBG boundary vs. cryptomodule boundary:

<u>Part 3 agreed upon changes</u>:

1. A handle may also be a pointer.
2. From John's slide 8: 2^{64} relative to a finite period of time. Consider parallel processing. Use a horizon of 50 years. Define for a specific algorithm.
3. From John's slide 12: allow 2^{64} bytes between reseeds. Eliminate counters, where possible.
4. Provide sample moduli for MS_DRBG
5. Each component of a distributed DRBG needs to be validated.
6. Acknowledge that the DRBG and cryptomodule boundaries are different.
7. Include the Entropy source within the outermost DRBG boundary, as a minimum. Sources outside the DRBG boundary are suspect.
8. Testing is within the DRBG boundary.
9. Add text on reseed management (John).
10. Request entropy up to the "state" size.
11. *min_entropy* = *requested_strength* + 64.
12. The current pseudocode should be "recast" as examples. Provide more generic specifications.