comments_to_elaine_Oct12

I've read through the latest X9.82 Part3 document. There are just a small number of things which I'd like to point out in the Dual_EC and MS sections. (There is also one MAJOR issue, that of removing the security_level=256 curves. But I think you've probably heard about that from Debby and Mike. I'm not the final authority on this. No one, including myself, is having an issue with removing the 192 or 256 bit security levels from MS. Those require enormous modulii, and are probably not likely to be popular. I certainly don't have a problem leaving them in, either.)

Before then, I would like to comment about "full_entropy" in general. I don't see how or why a Deterministic RBG should support "every bit is independent of every other bit". To me, that sounds like a Nondeterministic generator. Refreshing the entropy on every call is about the best you could expect of a Deterministic RBG, and that's taken care of by "prediction_resistance". Please reconsider this issue. I'll include some specific instances of where full_entropy raises questions, regardless.

1) Section 9.3, #7: If full_entropy_flag is set, then min_entropy = full_entropy.

I didn't understand this statement.

2) Section 10.3.2.1.
a) In Table 5, I don't understand why entropy would ever be larger than outlen. I would prefer that the full entropy lines be omitted, but if this is not possible, then the table entries ought to be equal to outlen.

b) max_number_of_bits_per_request should be set to outlen*reseed_interval.

(Otherwise, in Section 10.3.2.2.4, #1 in Process would always cause an exit without generating any random.)

c) 2^35 seems excessive for input strings. Could we set this to something more reasonable, say 1024 bytes ?

3) Section 10.3.2.2.3: "Step 1 shall be preceeded by the following step..." The parenthetical remark is referring to one of the Hash DRBGs.

This occurs again in the next section, under "Implementation notes".

4) Section 10.3.3.1.
a) In Table 6 I'd prefer to see the full_entropy lines omitted. If that's not possible, I'd again say that "full_entropy" should equal outlen. The line headed "Minimum entropy input length" seems redundant.

b) max_number_of_bits_per_request should be set to outlen*reseed_interval.

c) 2^35 seems excessive for input strings. Could we set this to something more reasonable, say 1024 bytes ?

5) In Annex B: I don't think that the sections B.5 to B.10.3 are necessary. The details about phi(x) given in 10.3.2.2.4, together with the "NOTE" given in the section "Curves over Binary Fields in Appendix A. should be sufficient for implementors. There is also a reference to FIPS 186-3 for more details, which includes those in B.5 to B.10.3.

6) Unless they are needed for other DRBGs, B.11 to B.15 should also be eliminated.

**********

I'll try to address your questions from "X9.82, Part 3 Issues" pertaining to Dual_EC and MS

12. OK to delete the text you've highlighted. I've addressed most of the Table 5 questions above. min_entropy_length = min_entropy sounds right, but also redundant.

As far as I could tell, you consistently changed all the names, so they're OK.

13. Yes, the secret state is s.

14. Algorithms are OK. The curve preference assumes all are available. If only binary curve types have been implemented, and no type is specified, then the user should get a binary as default (and be happy with it).

Having worked closely with someone who implemented the number theoretic DRBGs, I can state how useful it was for him to work with byte boundaries. He could not find a standard implementation of Hash functions which worked with anything but byte strings. It would have required him to rewrite those functions. Working with bits at some level is unavoidable, since the binary curves, at least, all have odd length "m". But the standard multiprecision libraries are used to dealing with that.

So my vote would be to use byte strings whenever possible: calls to hash functions, personalization strings, additional input. Truncating the final output string to any number of BITS--odd or even--is no big deal. (One should be able to ask for 1 bit at a time, if one is so inclined.)

15. By all means, delete it. The range of e values are those suggested by the inventors of the MS algorithm. e=3 (e.g.) is not a security breech, so long as the other parameters are chosen as specified in 10.3.3.2.3. Table 6 was addressed above.

16. Yes, the secret state is S.

17. Yes, the substitutions are consistent. Again, the range of e values are those given by the inventors.

18. Algorithms are OK

20. PROBLEM ! *I* would like the 256-bit curves ! (But I'll defer to a higher authority for a corporate response on this.)

21. Please see my comments above.

**********

Why one would use Dual_EC or MS.

Each of these DRBGs bases its security on a "hard" number-theoretic problem. For the types of curves used in the Dual_EC_DRBG(), the Elliptic Curve Discrete Logarithm Problem has no known attacks that are better than the "meet-in-the-middle" attacks, with a work factor of sqrt(2^m). In the case of Micali_Schnorr, based loosely on the RSA problem, the work factor of the best algorithm is more complex to state, but well-established.

These algorithms are decidedly less efficient to implement than some of the others in X9.82 Part3. However, in those cases where security is the utmost concern, as in SSL or IKE exchanges, the additional complexity is not usually an issue. Except for dedicated servers, time spent on the exchanges is just a small portion of the computational load; overall, there is no impact on throughput by using a number-theoretic algorithm. As for SSL or IPSEC servers, more and more of these are getting hardware support for cryptographic primitives like modular exponentiation and elliptic curve arithmetic for the protocols themselves. Thus it makes sense to

utilize those same primitives (in hardware or software) for the sake of high-security random.