Comments Received on FIPS 186-3

Date: Tue, 14 Mar 2006 08:45:32 -0800 (PST)
From: D Jablon <jablon1363@yahoo.com>

So far, I've found just a few minor non-functional problems, with one exception, on p. 42.

On p. 42, 1.1.3, step 13.8, I think "go to step 15" should probably be "go to step 14".

Can you please let me know if this is not the case?

Curiously, this step relates to a functional question we had in P1363 about the efficiency of verifying pseudo-random probable primes.

The other minor problems that I found so far are listed below.

Best regards,

David

==========

p. 3, in 14. Cross Index, should probably add "IEEE Std 1363-2000, Standard Specifications for Public-Key Cryptography" since it is referenced on pages 92, 94, 117, and 118.

p. 38, 1.1.1, Input:, There's no item 2.

p. 92, Appendix E, 2nd sentence, should change "IEEE P1363" to "IEEE Std 1363-2000". (The "P" was only needed for the "proposed" standard.)

p. 94, E.1.1.4, last sentence, should change "IEEE P1363" to "IEEE Std 1363-2000".

p. 94, E.1.2, footnote 2, should change "IEEE P1363" to "IEEE Std 1363-2000".

p. 117, E.9, last sentence, should change "IEEE P1363" to "IEEE Std 1363-2000".

p. 118, E.10, last sentence, should change "the IEEE P1363 standard" to "IEEE Std 1363-2000".

p. 119, Appendix F, second sentence, should change "Let SHA(...)" to "Let Hash(...)".

Date: Thu, 13 Apr 2006 05:47:09 -0700 (PDT)
From: D Jablon <jablon1363@yahoo.com>

Here are some more comments & corrections.

Item (1) came from a discussion in the IEEE P1363 working group, but we didn't really attempt to gather an official working group consensus on this issue.

-- David

Comments:
=========

(1) It seems like differences between A.1.1.1 and A.1.1.3 unnecessarily make existing implementations of previous standards incompatible with the new FIPS 186-3 recommendation for generating verifiably pseudo-random probabilistic primes.

Why is method A.1.1.3 not compatible with A.1.1.1? Was the former method insecure, at least for the case where L=1024, N=160, and Hash=SHA-1? The changes in the way the hash is used breaks interoperability with FIPS 186-2 generation in this case, and also breaks interoperability with ANSI X9.42 and RFC 2631, which were extended forms of the 186-2 method.

I suggest that the FIPS 186-3 methods could be modified to be interoperable with the FIPS 186-2 method, at least for the 1024/160 case that is not (yet) deprecated.

(2) If these methods aren't consolidated then consider renaming the titles of A.1.1.1 and A.1.1.3. The current titles ...

A.1.1.1 Validation of the Probable Primes p and q that were Generated Using SHA-1
A.1.1.3 Validation of the Probable Primes p >and q that were Generated Using an Approved Hash Function

... are doubly ambiguous, since both methods verify probable primes that (may) have been generated using SHA-1, and both verify probable primes that were generated using approved hash functions.

These ambiguities could be eliminated by highlighting functional differences, as in:

A.1.1.1 Validation of the Probable Primes p and q that were Generated by FIPS 186-2

A.1.1.3 Validation of the Probable Primes p and q that were Generated by FIPS 186-3

(3) On p. 52, in A.2, regarding the sentence:

... The first method, discussed in Appendix A.2.1, may be used when complete validation of the generator g is not required; it is recommended that this method be used only when the party generating g is trusted to not deliberately generate a g that has a known arithmetic relationship to another generator g'.

Strictly speaking, the recommendation in this sentence cannot be followed. In GF(p), there are *always* readily-discernable arithmetic relationships between any g and g', using either the addition or multiplication operator. For example, computing x := g' - g mod p shows one easily determined relationship between g and g'.

More specifically, it must be hard to determine an *exponential* relationship, as in g == g'^x mod p for a known x.

The phrase "arithmetic relationship" could be changed to "exponential relationship", to clarify this.

Similarly, ...

(4) On p. 53, in last paragraph of A.2.2, consider changing "a relationship" to "a known exponential relationship".

Correction:
==========

p. 38, A.1.1.1, Process step 1, change "len(p) != 160" to "len(q) != 160".

Subject: ECDSA hash functions
Date: Fri, 21 Apr 2006 15:56:46 -0700
From: "Robert Jueneman" rjueneman@spyrus.com

Several issues have come up regarding the interpretation of FIPS 186-3 and the hash functions to be used with ECDSA. We would appreciate NISTs point of view of in this matter.

The relevant paragraph in section 6.1.1 states:

An Approved hash function, as specified in FIPS 180-2, is required during the generation of digital signatures. The security strength of the hash function used **shall** meet or exceed the security strength associated with the bit length of *n*. The security strengths for the ranges of *n* and the hash functions are provided in SP 800-57. It is recommended that the security strength associated with the bit length of *n* and the hash function be the same unless an agreement has been made between participating entities to use a stronger hash function; a hash function that provides a lower security strength than is associated with the bit length of *n* **shall not** be used. If the output of the hash function is greater than the bit length of *n*, then the leftmost *n* bits of the hash function output block **shall** be used in any calculation using the hash function output during the generation or verification of a digital signature.

We wont argue with the use of truncation if unequal hash lengths and ECDSA keys must be used, for some reason.

The real issue is whether that type of mismatch should be allowed at all, and if so, why, and under what circumstances.

The first question concerns the use of SHA-1 in combination with ECDSA using P-256 or higher (Suite B). As we understand it at present, since FIPS 186-3 is not yet approved, FIPS 140-2 requires the use of SHA-1 with ECDSA, and no other, at least if the device or module is to receive FIPS certification for the ECDSA function. That obviously flies in the face of sound security practice and the guidance of FIPS 186-3, so the question is whether SHA-1 and ECDSA with P-256 or higher should be allowed at all. SPYRUS presently supports that combination, but only because we understand that we dont have any other choice if we want FIPS 140-2 certification.

The next question concerns lower strength SHA-2 functions with ECDSA of higher strength, e.g., SHA-224 with P-256 keys, or SHA-256 with P-384, or SHA-384 with P-521, etc. At present, SPYRUS supports those combinations. Since we allow SHA-1, we found it hard to argue against a higher strength hash function. However, in retrospect we believe that those choices should be disallowed, and that lower security strength hashes should always result in an error condition being returned.

Finally, what about the reverse case, where a longer hash function is used with a shorter ECDSA key. Should that case be allowed, or rather must that case be allowed?

Since SHA-384 is essentially a truncation of SHA-512, there is NO good reason to use SHA-512 with a P-384 key. On the other hand, there are lots of reasons why such a combination should not be allowed, including code bloat, increased testing, and lots of interoperability issues in negotiating what is to be done.

A more interesting case is SHA-256 vs. a 256-bit truncation of SHA-512. Now, as I recall, SHA-512 uses 80 rounds, vs. 32 rounds for SHA-256, but it also uses a longer block size. So which 256-bit hash would be stronger? I certainly cant answer that question. But we would argue that if someone is that concerned about the security of the hash function, then they probably ought to use a stronger ECDSA key as well, which would make the issue moot.

For these reasons, our recommendation for this section of FIPS 186-3 would be the following:

1. Explicitly permit the SHA-2 algorithms to be used with ECDSA signatures, for the purpose of FIPS 140-2 certification.
2. Allow SHA-1 to be used with ECDSA signatures of higher strength, but set a sunset date of between 2008 and 2010 when such usage will not be FIPS 140-2 or FIPS 186-3 compliant. This should be issued as a Change Notice to FIPS 186-2, rather than waiting for FIPS 186-3 to be approved.
3. Disallow the use of lower strength SHA-2 hashes with higher strength ECDSA keys, e.g., SHA-224 with P-256 keys, or SHA-256 with P-384 keys.
4. Specify that higher strength SHA-2 hashes **should not (**as opposed to **shall not)** be used with lower strength ECDSA keys, for reasons of performance and interoperability. I.e., the paragraph should read as follows:

An Approved hash function, as specified in FIPS 180-2, is required during the generation of digital signatures. The security strength of the hash function used **shall** meet or exceed the security strength associated with the bit length of *n*. The security strengths for the ranges of *n* and the hash functions are provided in SP 800-57. For performance and interoperability reasons, the security strength associated with the bit length of *n* and the hash function **should** be the same unless an explicit agreement has been made between participating entities to use a stronger hash function; a hash function that provides a lower security strength than is associated with the bit length of *n* **shall not** be used. If the output of the hash function is greater than the bit length of *n*, then the leftmost *n* bits of the hash function output block **shall** be used in any calculation using the hash function output during the generation or verification of a digital signature.

Your opinion on these issues would be most appreciated.

Regards,

Bob

Date: Mon, 01 May 2006 13:45:52 -0400
From: Bruno Couillard bruno@bc5tech.com

BC5 Technologies' Comments on the following document:
"FIPS PUB 186-3"
DRAFT Version Issued March 2006

Introduction:

The comments provided on the subject document are divided into three (3) categories: Critical, Substantive, and Administrative. Critical comments are comments that are deemed to require resolution before completion of this document. Substantive comments are comments that improve technical accuracy or clarify an item. Administrative comments correct items such as punctuation, grammar and spelling.

Critical Comments

Number	Reference	Comment
1.	Section 2.4, 1st paragraph, Page 24	This standard establishes four possible choices for the pair *L* and *N* (the bit lengths of *p* and *q*, respectively). This standard also makes multiple references to NIST SP 800-57 for information pertaining to security strengths related to such choices. The point of this comment is that NIST SP 800-57 does not define a "security strength" equivalency when the *L*, *N* choice is: (*L*=2048, *N*=256), but this new FIPS PUB 186-3 standard proposes to allow for this possibility. Therefore, it is recommended that either the (*L*=2048, *N*=256) choice is removed from the list of candidates in this FIPS 186-3 standard, or an amendment to NIST's SP 800-57 be made to account for this new possibility or finally, a special note be placed in this FIPS PUB 186-3 standard to define the equivalent "security level" provided by this possible choice.
2.	Section 6.6.1, Table 1	In this table, the first possible bit lengths for n is listed as "161-223". The NIST Special Publication 800-57 always uses the range "16**0**-223" instead. It is recommended that the entry in this table be aligned with the choices offered in SP 800-57 that is "160-223".
3.	Section A.1.1.1, Process area	In the process area of this algorithm, under step 1, change the text to: "if (len(*p*) ≠ 1024) or (len(***g***) ≠ 160), then return INVALID." In other words, the second "*p*" should be replaced by a "*q*".
4.	Section A.1.1.2, 1st paragraph	In this paragraph, the second statement refers the reader to SP 800-57 to determine the adequate hash strength for the specific selected (*L*, *N*) pair. Two comments arise: 1- As per comment 1 above, a note needs to be added

		to ensure that the reader knows what security strength the (*L*=2048, *N*=256) choices corresponds to; and 2- The table presented in SP 800-57 breaks the choices of hash function into five possible categories of operations: a. Digital signature and hash-only application; b. HMAC; c. Key Derivation Function; d. Random Number Generation; and e. Other (To be determined). The question here is: "To which category should the reader be referred to when choosing a proper hash function for the algorithm presented in this section of the FIPS PUB 186-3 standard?" Should it be category "a" or "c" or "d" or even "e"? Based on the answer to this question, a further note may be required given the fact that categories "b", "c" and "e" presented in table 3 of SP 800-57 (August 2005 version) are listed as "To Be Determined".
5.	Section A.1.1.2	Could the algorithm presented in this section be made to accept the "*domain_parameter_seed*" as opposed to generating it? This would further increase the assurance that the domain parameters could not have been selected with any weak parameters. This would require a few modifications to the process, but would improve the level of trust offered by this method.
6.	Section A.1.1.3	The algorithm presented in this section for the validation of the probable *p* and *q* parameters seems to be overly costly from a processing point of view. Step 13 seems to force the entire repeat of the process used during the search for the parameters. It would seem that a more expeditious approach would use the know value of "*counter*" to quickly generate the candidate to be checked and that the primality check be performed on that specific candidate as opposed to every candidates values as currently prescribed.
7.	Section A.1.1.4.1	Shouldn't one of the inputs to the algorithm prescribed in this section be the "*iterations*" required? This would allow for adjusting the probability value required every times this algorithm is called. Furthermore, a note or some text should be added to explain the relationship between "*iterations*" and the probability of a tested candidate of being prime under this algorithm.
8.	Section A.1.1.4.2	Same comment as comment "7" above.

9.	Section E.1.1.2, Table E-1	The same comment as comment "2" above.

Substantive Comments

Number	Reference	Comment
1.	Section A.1.2.1.2, Process, Step 3	The text associated to step 3 describing the process for this algorithm should read: "Using $\lceil L/2 + 1 \rceil$ **as the length** and *qseed* **as the *input_seed***, use the random …"
2.	Section A.2.3	Suggest that the title for this section be: "Verifiable **Canonical** Generation of Generator *g*" to align this title with the title used for section A.2.4.
3.	Section A.2.3, process area.	In the process area under step "7", it would make sense to add a note to clarify the usage of the text string "ggen". This is the first time such as text string is being used in the document and its usage is not clear at first.
4.	Section C.1.1, bullet #2.	In the description of "*timestamp_signature*", shouldn't there be a note (or foot note) indicating that there is an underlying assumption that the digital private key used for performing such "*timestamp_signature*" shall not be used for any other purposes?
5.	Section C.2	It is suggested that the usage of "*entropy*" for describing the strength of the "nonce" be changed to: "*unpredictability work factor*" or something similar. The idea being that the "*nonce*" may actually be generated by the sender using a hash function over a secret counter and contain "0" entropy, but still be "*unpredictable*" to the recipient.
6.	Section D.5, Process description	In step "1" and step "6.4" of the process description, where does the "*nlen*" superscript value comes from? Should this be somehow related to the input "*security_strenght*" or in fact replaced by that input?
7.	Section E.1.1.3	Would suggest changing the title to: "Choice of Basis **for Binary Fields**".
8.	Section E.2.1	Would suggest using the same style of description as used for E.2.2., E.2.3 and E.2.4, That is: "The modulus …written as $A = A_5 \ldots + A_0$, where each A_i, is a 64-bit integer. **As a concatenation of 64-bit words, this can be denoted by:** $A = (A_5 \| A_4 \| \ldots \| A_0)$. The expression for *B* is

Number	Reference	Comment
		$B = T + S_1 \ldots + S_3$ mod p; Where the 192-bit terms are given by $T = A_2 \| A_1 \| A_0$ $S_1 = A_3 \| A_3$ $S_2 = A_4 \| A_4 \| 0$ $S_3 = A_5 \| A_5 \| A_5$"
9.	Section E.2.5	Same comment as comment 8 above with respect to trying to remain constant in the way things are described.

Administrative Comments

Number	Reference	Comment
1.	Section 6.1.1, Table 1	The header for this table should be moved to the next page with the rest of the table's content.
2.	Section C.2, 2nd paragraph	The first line should read: "…verifier-supplied date (i.e., supplied by entity B) with …"
3.	Section D.5, Process description	In step "1" of the process description, it would be advised to increase the font used for the value "*X*" in the formula to make it the same size as the "*X*" used in step "3".
4.	Section D.6, Process description	Step 2 should be changed to: "…in the sequence {5,-7,9,-11,13,-15,17} for …" to remove the comma after "17" in the list.
5.	Section D.7, Input description	The input value for "a' should be changed to: "…in the sequence {5,-7,9,-11,13,-15,17} as …" to remove the comma after "17" in the list.

Date: Fri, 12 May 2006 05:08:04 -0700 (PDT)
From: Daniel Bleichenbacher <daniel_bleichenbacher@yahoo.com>

this is just a small comment on an apparent type in the DSS draft.

In Section B.3.2 "Generation of the prime factors p and q for RSA" on page 63: Step 7 of the algorithm appears to check that the RSA modulus cannot be factored using Fermat's factoring method. If that's the intention then one should reject p and q if |p-q| < 2^(nlen/2 + security_strength + 20) and not |p-q| < 2^(nlen/2 - (security_strength + 20)).

Daniel Bleichenbacher

From: Robert Zuccherato <robert.zuccherato@entrust.com>
Date: Tue, 16 May 2006 09:31:19 -0400

Note on terminology: We use the term "simplicity" or "simplicity argument" some times below. The simplicity argument is that, all else being equal, a design specification such as FIPS 186-3 should be as simple as possible, as complexity comes with an implementation cost. We see simplicity as one of the factors to be considered along with security, performance, and interoperability. Obviously, there are things that can indicate that a simpler specification is not preferred, but when this is the case, some rationale would be useful. That is, flexibility just for flexibility's sake is not a good justification.

1. Overall comment: No calculation examples are specified but are critical to ensure correct understanding of the various calculations and also for (limited) testing of correct implementation. Without calculation examples, it is possible that there are alternate meanings of the specification (or even errors) that are different from those intended by NIST and which would lead to costly implementation errors. As the 80-bit security is expiring in 2010, there need to be examples using a security level of 112-bit or higher. For this reason (and also the number of comments below) we hope there will be at least **one more draft FIPS 186-3** for public comment before this specification is finalized.

2. Overall comment: This specification is not clear about whether the algorithms listed must be implemented exactly as specified or if equivalent methods that obtain the same results are permitted. We suggest that equivalent methods be allowed.

3. Section 2.2: "TTA" should be after "TSP" in alphabetical order.

4. Section 2.3: "len(a)" is out of alphabetical order.

5. Section 3 says "Technically, a key pair used by a digital signature algorithm could also be used for purposes other than digital signatures (e.g., for key establishment). However, a key pair used for digital signature generation and verification **shall not** be used for any other purpose." This is true. However, it should also be true that a key pair used for digital signatures in this FIPS **shall only** be used for digital signatures methods specified in this FIPS, that is, it would be a violation of the FIPS to use such a key pair for other digital signature methods not in this FIPS. As this may not be obvious to the reader, we think it should be added to the text.

6. Section 3.2 says "The signatory **may** optionally verify the digital signature using the signature verification process. This may be prudent for a high-value message, when multiple users are expected to verify the signature, or if the verifier will be verifying the signature at a much later time." This is all true, bu the basic reason is to serve as a final check to detect possible signature generation calculation errors, which is why the stated reasons make sense.

Also, as ECDSA signature verification may use the private key, for this error checking purpose it should use the public key. So the sentences should be improved to say "The signatory **may** optionally verify the digital signature using the signature verification process using the associated public key. This optional verification serves as a final check to detect otherwise undetected signature generation calculation errors, so this may be prudent when signing a high-value message, when multiple users are expected to verify the signature, or when the verifier will be verifying the signature at a much later time."

7. Section 3.3: The signature verification process should specify exactly 2 return codes, "Signature Verified" and "Signature Not Verified". Then each return code can be discussed as to what it means. It is an architectural flaw to not give the return codes explicit names.

8. Section 3.3 says "If the verification process fails, no inference can be made as to whether the data is correct, only that the signature is incorrect for that data." This is not strictly true; it could be that the public key that was used is the wrong one, either because of damage to the value of the key or access to the wrong key for some reason. It could also be that a different RSA signature format (of the 3 allowed) was specified during signature verification than during signature generation. Suggest rewording to "If the verification process fails, no inference can be made as to whether the data is correct, only that in using the specified public key and the specified signature format, the digital signature does not verify for that data."

9. Section 3.3 says "However, if a verification or assurance process fails, the digital signature **should** be considered invalid." This is true, however, if an assurance process fails, then the public key itself should be considered invalid and not used. A new assurance process, with perhaps a new key pair, should be initiated.

10. Section 4.2 has an entry of L=2048 and N=256, but these 2 numbers are not at the same security level, L=2048 is assessed at a security level of 112 bits but N=256 would normally be assessed at a security level of 128 bits, but (to be consistent) will need to be reduced to an assessment of 112 bits. But a consistent L and N for the 112 bit security level is already in the list, so it is unclear why this entry exists. To reduce possible confusion, some rationale should be provided, such as this being for transition purposes, where everything except the public key keysize is at 128 bits and therefore almost ready to convert to fully 128 bits.

11. Section 4.2: With a non-CA user limited to the first 3 pairs, a user cannot use a DSA signature with a security level beyond 112 bits which means the protection cryptoperiod is only until 2030. That is, if a digital signature needs to be deemed valid beyond 2030, then DSA cannot be used by a non-CA user. Limiting DSA key sizes to 2048 bits does not leave much leeway for possible unforeseen developments in cryptanalysis. Therefore, it seems prudent to at least allow 3072 bit DSA keys.

12. Section 4.2: As DSA key pairs **shall not** be used for other purposes, it should be specifically stated that FFC domain parameters **may** be used for other purposes. Also the title of this section should be changed from DSA domain parameters to FFC domain parameters to help avoid possible confusion, as FFC domain parameters may be also used for key establishment, as specified in other NIST documents.

13. Section 4.2: Given the previous comment (that the domain parameters are really FFC), it is best if each use has its own canonically-generated generator. This would mean, at least, a generator for DSA use and a generator for Key Establishment use.

14. Section 4.4.2: It is probably implied and may be deemed obvious but should be explicitly stated that a DSA key pair can only be used for DSA signatures, and **shall not** be used in another type of (FFC) digital signature algorithm (such as Nyberg-Rueppel or Pintsov-Vanstone), as use in another algorithm could weaken the security.

15. Sections 4.5, 4.6 and 6.3: These sections allow the precomputation of the values k and k^{-1}. These are sensitive values however, and if they were made available to an attacker could potentially compromise the user's private key. Thus, a comment should be made in these sections that these values **shall** be protected for confidentiality and authenticity if they are precomputed.

16. Section 4.7 item 4 says "No inference can be made as to whether the data is valid, only that the signature is incorrect for that data." This is true, but we suggest improving it to: "No inference can be made as to whether the data is valid, only that when using that public key the signature is incorrect for that data." It is possible that the reason a signature fails is the public key used in the verification has been corrupted or incorrectly accessed, that is, if the wrong key was used to validate the signature.

17. Section 5.1 says "An RSA key pair used for digital signatures **shall** only be used for digital signatures, not for other purposes (e.g., key establishment)." We suggest that this further state that an RSA key pair **shall** only be used for ANS X9.31-1998 or PKCS #1 version 2.1 signatures (either the PSS method or the PKCS 1.5 method) and **should** be used for only one of these three formats.

Note that OIDs would likely need to be enhanced to support such a granular specification of signature format.

18. Section 5.1: With a non-CA user limited to the first 2 keysizes, a user cannot use an RSA signature with a security level beyond 112 bits which means the protection cryptoperiod is only until 2030. That is, if a signature needs to be

deemed valid beyond 2030, then RSA cannot be used by a non-CA user. Limiting RSA key sizes to 2048 bits does not leave much leeway for possible unforeseen developments in cryptanalysis. Therefore, it seems prudent to at least allow 3072 bit RSA keys.

19. Section 6, Paragraph 1: This paragraph requires that the most current version of ANS X9.62 be used. Thus, it will automatically require compliance with updates to X9.62 that have not yet been produced. Since they have not yet been produced, we do not know what will be in these updates and they may impose requirements that are inconsistent with NIST's desires. We also note that for RSA, only the 1998 version of ANS X9.31 is allowed. We suggest that a similar logic be applied to ECDSA and that only the 2005 version of ANS X9.62 be allowed in this FIPS. If and when any ANSI X9 algorithm specification is updated, then NIST can decide whether (and how) to update their specification also.

20. Section 6.1: As ECDSA key pairs **shall not** be used for other purposes, it should be specifically stated that ECC domain parameters **may** be used for other purposes. Also, to help avoid possible confusion, the title of this section should be changed from ECDSA domain parameters to ECC domain parameters, as ECC domain parameters may also be used for key establishment and random number generation as stated in other NIST documents.

21. Section 6.1: Given that the domain parameters are really ECC, it is best if each use has its own canonically-generated generator. This would mean, at least, a generator for ECDSA use, a generator for Key Establishment use and two generators for the ECC RNG.

22. Section 6.1: It should be stated that either use of NIST recommended curves or the use of a verifiable generator are the methods that are preferred. It would be best if the verifiable generator be required to be used for any new domain parameters.

23. Section 6.1.1 Table 1 indicates that ECDSA domain parameters may support any security level. As noted in the comments on Section 4.2 for DSA and Section 5.1 for RSA, this means that only ECDSA may be used by a non-CA user when a security level of 128 bits or more is desired, which is when protection for the signature needs to extend beyond 2030. However, besides security level/keysize considerations related to that timeframe, there can also be algorithm security considerations. One way to mitigate possible algorithm security considerations is for FIPS 186-3 to at least allow digital signatures from multiple algorithms, so that if and when one algorithm would be broken (by a significant advance in mathematics, for example), the other algorithm may still be considered secure and so the signed message would still verify using an unbroken algorithm. This multiple algorithm possibility may be a rationale for NIST to modify this FIPS to allow RSA/DSA keys for non-CA users to be larger than 2048 bits.

24. Section 6.1.1 says "Normally, a CA **should** use a bit length of *n* that is equal to or greater than any bit length of *n* used by its subscribers." We think that this should state "assessed security level" rather than "bit length" to allow for interoperability of key lengths that are assessed the same but are slightly different in actual bit length. The cross certification scenario makes this a very real possibility, so this needs to be made explicit.

25. Section 6.1.1, Paragraph 4: This paragraph states that it is okay (subject to agreement of the participating entities) to use a group order *n* that is less than the hash function size. However, it is disallowed to use a hash function whose output size is less than the group order. It seems unreasonable to allow *n* to be less than the hash function size and to not allow the hash function size to be less than *n*. We suggest that this requirement either be removed or justified. Is the intent to allow flexibility if a hash is broken?

26. Section 6.2: A requirement should be placed in this section (copied from the second paragraph of Section 6) saying that ECDSA public keys SHALL only be used for digital signatures and not for other purposes, as is the case with DSA and RSA public keys. Also, they should not be used for digital signature methods other than ECDSA (such as ECPV, etc.)

27. Section 6.4: It would be useful to state that ECDSA signature verification can be done using either the public key or private key if it is available, assuming this is the intent of NIST.

28. Section A.1.1.1, Process, Step 1: It should be "**len**(*q*)≠160" not "len(p)".

29. Section A.1.1.4.2: It is not clear why the Enhanced Miller-Rabin Probabilistic Primality Test is included here. The additional information that this test provides over the test in Section A.1.1.4.1 may be useful when generating or validating RSA moduli, however it is not useful when generating DSA parameters, unless it already exists for RSA purposes and so could be reused. So the EMR test should be put in the RSA section and a note added elsewhere that it can be used instead of the normal MR test.

30. Section A.1.2: It is worth mentioning that a straightforward way for two parties to contribute to the seed in a way that is not open to manipulation less than the workfactor associated with a hash function is the following procedure:

A) The first party selects an arbitrary value v1, commits to it by hashing the value, and sends the hash h(v1) to the second party.
B) The second party selects an arbitrary value v2 and sends v2 to the first party.
C) The first party responds with their selected value v1.
D) The second party verifies that the hash of the selected value h(v1) is equal to the hash value received h(v1').
E) These two values are then bitwise Boolean exclusive-Ored (v1 XOR v2) by both parties to produce the seed used to generate the domain parameters. Each party knows that the seed was not selected totally by the other party.

31. Section A.1.2: It is not clear why the method of constructing primes *p* and *q* are included here. Inclusion of another generation algorithm will complicate domain parameter validation testing. The existing method is 99.9999+% sure of generating primes while the new method is 100%; the existing method might generate one false prime sometime, but even this is not expected to happen. For simplicity considerations, is adding the new prime construction method worth the cost of complicating validation testing?

32. Section A.1.2.1.1, Output, Point 1: Change "validation" to "generation".

33. Section A.2.1: Allowing unverifiable generation of the generator *g* may open an avenue for an attacker to forge signatures. Thus, we recommend that this method of generating *g* be disallowed for new domain parameters. Existing domain parameters would be grandfathered and allowed to use the old method.

34. Section A.2.3, Process, Step 7: This step is not specified in sufficient detail. For example, the construction of the *domain_parameter_seed* is not described, beyond saying that it is the concatenation of *firstseed*, *pseed*, and *qseed*. The order and encoding of these values is not specified. Also, the encoding of "*ggen*" is not described.

35. Section B.1: Generally speaking, when there are two acceptable methods of performing an operation, we would prefer if NIST would choose one method over the other, for simplicity. This would simplify implementations and the validation process. Thus, we would suggest that NIST choose one of the two methods (Section B.1.1 or B.1.2) as the approved method for FFC key generation. We would prefer that B.1.1 be chosen, but B.1.2 is acceptable.

36. Section B.2: We would prefer that NIST choose one of the two acceptable methods for per-message secret number generation, for simplicity, in order to simplify implementation and validation. We would prefer that the method in B.2.1 be chosen, but B.2.2 is acceptable.

37. Section B.3.1 Item 1 (b): Besides security, a cryptosystem should be designed with performance, interoperability and simplicity in mind. Given these other design criteria, it is unclear why an RSA public key should not always use a public exponent e = 65,537 as this simplifies the generation, validation, transmission and storage of an RSA public key while providing the best performance (among the allowed values of e as a larger e will use more modular arithmetic calculations). In effect, this decision would allow e = 65,537 to be a (NIST) RSA domain parameter, which simplifies things. Some rationale should be given for this flexibility, which comes at a design and implementation cost, or e = 65,537 be consistently used.

38. Section B.3.1 Item 3 says "In the extremely rare event that $d \leq 2^{nlen/2}$, then new values for p, q and d **shall** be determined, and a different value of e may be used." This is slightly confusing, it would be better to say "In the extremely rare event that $d \leq 2^{nlen/2}$, then a new RSA key pair shall be generated using a different random number seed."

39. Section B.3.2, Process, Step 3: It is unclear at this point if a sieve procedure can be used to filter out obviously composite values of X_{p1} and X_{p2}. For example, a literal reading of this step would require a composite test on even integers until an appropriate prime is found. It is suggested that the wording in this step be changed so that integers with small prime factors can be sieved out before searching for the first prime, in order to improve efficiency.

40. Section B.4: For simplicity, only one method of generating ECDSA key pairs is required. We suggest that B.4.1 be the only approved method, but B.4.2 is acceptable.

41. Section B.5: For simplicity, only one method of generating per-message secret numbers is required. We suggest that B.5.1 be the only approved method, but B.5.2 is acceptable.

42. Appendix C: Entrust has long been active in promoting timestamping and thus, we appreciate the emphasis being placed on timeliness and timestamping. However, we question the inclusion of this appendix in FIPS 186-3, which primarily describes a cryptographic algorithm, as it was not in it in previous versions. Perhaps it would be better if this appendix were included instead in NIST SP 800-57, which provides key management guidance or be its own standard.

43. Section C.1: Since there are a number of international standards dealing with timestamping and implementing versions of the protocols described in this appendix, it would be appropriate if they were referenced here. In particular, RFC 3161, ISO/IEC 18014, ANS X9.95, and OASIS DSS should be mentioned.

44. Section C.1: In addition to the techniques currently described in this appendix, it is important that one additional technique be included. The situation where an end entity obtains one trusted timestamp from a TTA on $SIG_A(M)$ is useful in providing evidence that the signature was created before the given time.

This supports validation of signatures after public key expiry or revocation and thus is important in many PKI environments. This technique would be similar to the one described in Section C.1.3.3, except that the additional signature created in Step 3 would be omitted.

45. Section C.1.3.2, Step 1: The *user_supplied_info* should contain H(M), it does not need to equal it.

46. Section C.1.3.3, Step 1: The *user_supplied_info* should contain $SIG_A(M)$, it does not need to equal it.

47. Section D.1: The values x, x_1, and x_2, which are computed in this algorithm, are never used. We recommend that they be removed, for simplicity.

48. Section D.2.2, Process, Step 2: The exponent on the 2 should be *n-i*, not *n-1*. (Incidentally, this shows the need for calculation examples.)

49. Section D.5: In generating the 2 RSA primes probabilistically, either 8 MR tests followed by 1 Lucas test or 50 MR tests are required. Adding text that discusses things to consider in deciding whether to use one or the other would be useful. For example, if code space is the critical concern, then 50 MR tests would probably be preferred. As the Lucas test is somewhat complex, if simplicity of design is the critical concern, then 50 MR tests would be preferred. There are no known non-prime values that pass 8 MR tests and 1 Lucas test, so the 8MR+Lucas method may provide some additional assurance regarding primality. It is also likely that the 8MR+Lucas method has better performance, although this depends on the machine. (This also serves as an example of giving rationale why both methods are included.)

50. Section D.5: The minimum rounds criteria for the probabilistic prime methods are a patchwork from the relevant standards; these should be unified and made consistent. Section B.3.2 requires 27 iterations of the Miller-Rabin test, however D.5 requires either (A) at least 8 MR along with one Lucas test, or (B) 50 MR. Why is 27 MR OK for B.3.2 and is 8 MR along with a Lucas test also OK? In A.1.1.4 for DSA primes, it should be stated that 8 MR+1 Lucas is OK. Basically, for a certain size candidate prime, regardless of the signature algorithm in which it will be used, a user needs to know the options for validating it is prime. These options should be consistent across all signature algorithms.

51. Section D.5: Assuming the Shawe-Taylor algorithm is kept, it is allowed to be used for generation of DSA prime cryptovariables, but it is not specified how to use the S-T method for RSA prime cryptovariables. Either the S-T method for RSA should be specified (but this may be too complex) or it should at least be specified that this omission is deliberate.

52. Section D.6 has Process step 1. Test whether *C* is an exact square. But no pseudocode is specified for testing if an integer is an exact square.

53. Section E: We think now is the time for NIST to generate canonical G's for specific uses (for example, digital signature, key establishment, random number generation, and perhaps more) for each of the NIST recommended curves, for those that wish to use them. A Koblitz curve can use the same seed as the corresponding random binary curve. The existing G for each domain parameter can be grandfathered for general use.

Date: Fri, 9 Jun 2006 15:14:18 -0400
From: "Garcia, Paul X" <GarciaPX@state.gov>

IRM/IA concurs on above mentioned subject without comment.

From: "Joshua E. Hill" <jhill@infogard.com>
Date: Mon, 12 Jun 2006 17:13:45 -0700

The draft FIPS 186-3 document generally seems like an excellent refinement of the current FIPS 186-2 standard, but we have a few comments on the document:

In Section 3, there are several paragraphs that appear to make diagrams normative requirements. It should be made clear that these diagrams summarize the normative requirements that are described later, and are not themselves the requirements (changing the word "shall" in these cases would be helpful). This occurs in:

- Section 3.1, Paragraph 1
- Section 3.2, Paragraph 1
- Section 3.3, Paragraph 1

In Section 5.1, Paragraph 3, there is an allowance for using the Chinese Remainder Theorem (CRT) with PKCS#1, but there is no mention of ANSI X9.31. It is not clear why using CRT with ANSI X9.31 (as allowed in ANSI X9.31-1998 Section 4.2, #4 and in Appendix E.1) is not permitted as well.

In Section 5.4, Paragraph 2 includes the statement, "The criteria for RSA key generation in ANSI X9.31 are consistent with the criteria in Appendix B.3.1." This seems slightly misleading, as the draft FIPS 186-3 differs in a few regards:

- FIPS 186-3 requires *e* to be larger than 65537, whereas ANSI X9.31-1998 only requires a value of 3 or greater.
- FIPS 186-3 requires the size of p1, p2, q1, q2 to vary with the size of the key. ANSI X9.31 does not.

In Section 5.4, it should be made clear if (in order to be compliant with FIPS 186-3) the procedure specified in B.3.2 must be followed even when generating keys for ANSI X9.31 signatures. The current language seems to defer to the ANSI X9.31 standard, which does not quite enforce the same requirements as B.3.1. (Note, the size of p1, p2, q1, q2 does not vary with the key size, for instance.) In addition, the ANSI X9.31 document does not specifically require the use of the procedure outlined in ANSI X9.31-1998 Section 4.1.2.1, whereas Section 5.5 of FIPS 186-3 does require this procedure for generation of keys for use with PKCS#1.

In A.1.1.4, a reference is made to an older edition of The Art of Computer Programming. This reference should be updated to refer to the currently available edition: Page 395 for the 3rd edition (copyright 1998).

In Section A.1.1.4, Paragraph 3, the number of iterations is expressed as being a fixed integer greater than or equal to 50. This is inconsistent with this routine's use described in B.3.2, Step 3 and D.5, Step 6.

In Section B.3.1, *e* is required to be in a particular interval (greater than or equal to 65537). This is a significant change from either PKCS#1 or ANSI X9.31 and has the potential of being fairly disruptive. It may be useful to cite a reason that this requirement is being imposed.

The length requirements for p1, p2, q1, q2 are not consistent throughout the document. Note the descriptions for these parameters (or their immediate antecedents) in:

- Section B.3.1, Step 2b
- Section B.3.2, Step 2
- Section D.5 (the description for r1 and r2)

In the last two of these instances (B.3.2, Step 2 and D.5) the statement is likely incorrect; it seems likely that both should reference $2^{(security\_strength+21)}$ and $2^{(security\_strength+40)}$, not (security_strength+21) and (security_strength+40).

In B.3.2, Step 7.2, it is unclear why a new value for p should be regenerated, when just regenerating a new q should suffice. If the intent was to start over entirely, then a jump to Step 2 seems more appropriate.

It is important to note that testing the key generation algorithms described in B.3.1 and D.5 will likely eventually be conducted through the CAVP. Meaningful testing will require the ability to externally enter all random quantities used in the generation procedure. An interface that supports this ability should be included in the procedural description of the key generation process. The current description does not make it clear that this style of testing should be included in a design, or for that matter, that such an interface would be allowed.

In D.5, the procedures specified used *nlen*, which is not passed in.

It is not clear how the number of Miller-Rabin test iterations for B.3.2, Step 3 and D.5, Step 6.2 was selected. ANSI X9.31-1998 uses the same number of iterations and in explanation (in ANSI X9.31-1998 Section B.2), it references The Handbook of Applied Cryptography, Table 4.4. (From the description, it seems likely that Table 4.3 is actually the correct reference.) Table 4.3 is excerpted from Damgård, Landrock and Pomerance's "Average Case Error Estimates for the Strong Probable Prime Test." The estimates in this paper are based on using a procedure that is different than the procedure outlined in FIPS 186-3 (and ANSI X9.31). This paper's analysis is based on the random search method, where a random odd candidate integer is selected from the correct interval for each round. The procedure adopted in FIPS 186-3 (and ANSI X9.31) is different (incremental search for p1, p2, q1 and q2 in B.3.2 and the CRT based approach used in D.5), so it is not clear that the analysis in the paper directly applies for the procedure outlined in FIPS 186-3.

Date: Mon, 12 Jun 2006 21:54:33 -0700
From: Wan-Teh Chang <wtchang@redhat.com>

Attached are my review comments on Draft FIPS 186-3, Appendices D-F. I also reviewed Draft FIPS 186-3 proper and portions of Appendices A-B, but I left my review comments at work. I will send you those comments tomorrow.

I tried to make sure the page numbers and section numbers are correct, but I may have made a mistake. If you have any questions about my comments, please
let me know.

Wan-Teh Chang

Here are my review comments on Appendices D-F.

1. Page 84, Appendix D: this appendix uses three symbols to denote the multiplication operator:
- big dot: in D.1 steps 4,5,6
- small dot: in D.7 step 8
- asterisk (*): most places

This is a little confusing.

2. Page 84, Appendix D: this appendix is missing "..." in several places, specifically, in D.2.1, D.4, and D.7.

3. Page 86, Appendix D.2.2: under "Process", step 1 says "where b1 = 0 or 1". "b1" should be "bi".

4. Page 92, Appendix E: it would be nice to say that the recommended elliptic curves are the same as FIPS 186-2.

5. Page 92, E.1.1.1: the last sentence says "the private and public keys for a curve are approximately the same length." This statement is true only if the public keys are ompressed. Otherwise, the public key should be approximately twice the length of the private key because the public key has two coordinates.

6. Page 92, E.1.1.2: the first sentence says "For each cryptovariable length". Cryptovariable" is not defined in this document.

7. Page 93, E.1.1.3: in the first bullet item, the bit string (am-1 a2 a1 a0) is missing "...". The polynomial on the next line is also missing "...". The second bullet item has the same problems.

8. Page 93, E.1.1.3: in the second bullet item, change "an element theta" to "a field element theta".

9. Page 93, E.1.1.3: I just wanted to make sure it is correct for the subscripts of the two bit strings in the first two bullet items to be in reverse order.

10. Page 93, E.1.1.3: in the paragraph under the second bullet item, change "For a given field degree m" to "For a given field of degree m".

11. Page 93, E.1.1.3: in the paragraph above the third bullet item, change "from which to choose" to "to choose from".

12. Page 93, E.1.1.3: in the third bullet item, remove "m" from "t^a has the lowest degree m". (The degree of t^a is a.)

13. Page 94, E.1.1.4: in the second bullet item, insert "are those" between "Special curves" and "whose coefficients".

14. Page 94, E.1.1.5: question: how do I generate my own base points?

15. Page 94, E.1.2: in the first paragraph, in "the cofactor is always f = 1", change "f" to "h".

16. Page 95, E.1.2: the last sentence says "The integers p and n are given in decimal form". As an implementor, I'd like to see p and n given in hexadecimal.

17. Page 98, E.1.3: the last sentence says "Integers (such as T, m, and n) are given in decimal form". As an implementor, I'd like to see n given in hexadecimal.

18. Page 106, E.2: in the third paragraph, remove "and reduce" from "the integer sum or difference and reduce".

19. Page 106, E.2: the last sentence repeats what the first paragraph says, so you can remove the last sentence. If you keep it, change "moduli p" to "modulus p".

20. Page 108, E.2.5: after the formula A = A1 * 2^521 + A0, add "where each Ai is a 521-bit integer."

21. Page 109, E.3: review the whole section to make sure you underline "u" and "v" where they denote a bit sequence. Note that in step 2 "u" denotes an integer.

You should also point out that F denotes both a function of an integer and a function of two bit sequences.

22. Page 109, E.3: in step 3, the semicolon after F(1) should be a comma. Add "..." after F(2).

23. Page 109, E.3: in footnote 3, the "S" in "Standard" is red.

24. Page 110, E.3: near the bottom, change the semicolon in F(u;v) to a comma.

25. Page 111, E.3: in the equation for c6, change the semicolon to a comma.

26. Page 111, E.4: the third and fourth bullet items use a slanted dot to denote the multiplication operator.

27. Page 113, E.4: in step 4, the first "then" uses a different font from the rest.

28. Page 114, E.4: in step 11.3, "Endwhile" uses a different font from the rest.

29. Page 116, E.9: in the first sentence, insert "is" between "Suppose that alpha" and "an element".

30. Page 117, E.10: same as above.

31. Page 119, Appendix F: in the first sentence, change "SHA(...)" to "Hash(...)" because that's what's used in the proof. Explain why we don't take the leftmost N bits of Hash(M) in this proof.

32. Page 119, Appendix F: the proof of the Lemma begins with "g^p mod p". Change "g^p" to "g^q".

33. Page 119, Appendix F: "Theorem" should be in boldface and underlined, like "Lemma".

34. Page 119, Appendix F: in the proof of the Theorem, the first three lines use the single quote character as the prime character in s', M', and r'. The statement of the Theorem uses the correct prime character.

35. Page 119, Appendix F: in the proof of the Theorem, perhaps change "so that by the lemma" to "so by the lemma".

Date: Tue, 13 Jun 2006 10:20:58 -0700
From: Wan-Teh Chang <wtchang@redhat.com>

This note is my review comments on Draft FIPS 186-3 proper and portions of Appendices A and B.

I welcome two changes in Draft FIPS 186-3
- explicitly allow using the same RBG to generate both the DSA private key 'x' and the per-message secret number 'k'. In contrast, 186-2 specifies separate RBG algorithms for generating 'x' and 'k'.
- officially recognize PKCS #1 RSA, which is widely used in practice.

Most of the items below are fixes for typos, minor errors, or cosmetic issues. The most important items are: 17, 18, 32, 33, 34.

1. Page 2, item 8: I suggest adding a hyperlink to the URL http://csrc.nist.gov/cryptval.

2. Page 2, item 10: I seem to recall that Bureau of Export Administration has been renamed Bureau of Industry and Security (BIS).

3. Page 3, item 12: change "NISTs" to "NIST's". Add a period (.) to the last sentence of this paragraph.

4. Page 6: the page number 38 for A.1.1.1 is not aligned to the right.

5. Page 11, Sec. 2.1: in the definition of "Approved", the second "either" probably should be removed. (I believe that sentence was truncated because I have seen the complete sentence in some other document. Unfortunately I don't recall where.)

6. Page 11, Sec. 2.1: in the definition of "Bit string", there are extra spaces in "0 s" and "1 s".

7. Page 13, Sec. 2.1: "May" (in boldface) should be defined along with "Shall" and "Should". The document uses "may" on pages 22, 25, and 26.

8. Page 14, Sec. 2.1: add the definition of "Timeliness".

9. Page 15, Sec. 2.3: in the definition of "m", it would be nice to use GF(2^m) (consistently throughout the document) instead of F sub 2^m to denote a binary field.

10. Page 16, Sec. 2.3: in the definition of "nlen", should say "The length of the RSA modulus n in bits".

11. Page 16, Sec. 2.3: in the definitions of "r" and "s", should say "One component of a DSA or ECDSA signature". See the definition of "(r,s)" on page 17.

12. Page 16, Sec. 2.3: in the definition of "seedlen", should say "The length of the seed for

the domain_parameter_seed in bits".

13. Page 17, Sec. 2.3: in the definition of "{, a, b, }", add "..." after "b,"

14. Page 19, Sec. 3: in the third paragraph, there is an extra comma (,) after "(i.e., the signed data)".

15. Page 19, Sec. 3: in the fifth paragraph, "the key pair owner actually possesses the associated private key" should be either "the public key owner actually possesses the associated private key", or "the key pair owner actually possesses the private key".

16. Page 22, Sec. 3.3: in Figure 4, change "Alleged Signatory" to "Claimed Signatory" (two occurrences).

17. Page 23, Sec. 3.3: in the last paragraph, I don't understand why "should" rather than "shall" is used in the sentence "However, if a verification or assurance process fails, the digital signature should be considered invalid."

18. Page 24, Sec. 4.1: in the definition of "g", the constraint on g should be "1 < g < p".
^

19. Page 24, Sec. 4.2: the last sentence on this page should read "If the output of the hash function is longer than N" or "If the length of the output of the hash function is greater than N".

20. Page 26, Sec. 4.4.2: item 3 is a little ambiguous. Does it mean the key pairs shall only be used with their associated domain parameters and shall not be used with other domain parameters?

21. Page 27, Sec. 4.5: in the second paragraph, "multiplicative" and "with respect to multiplication" are redundant. I suggest removing "with respect to multiplication".

22. Page 27, Sec. 4.5: also in the second paragraph, the exponent -1 in k^-1 (except the first instance) is a little too low.

23. Page 33, Sec. 6.1: in the second paragraph, I don't know whether the comma (,) before the optional information {domain_parameter_seed} should be outside or inside the curly braces. The definition of "{,a,b }" on page 17 implies the comma should be inside. There are several other instances of this problem. Since this is just a cosmetic problem, I won't list the other instances of the problem.

24. Page 33, Sec. 6.1: in the second paragraph, may want to change "generating point" to "base point".

25. Page 34, Sec. 6.1.1: in the first paragraph, use GF(p) instead of F sub p, and GF(2^m) instead of F sub 2^m.

26. Page 34, Sec. 6.1.1: in the second paragraph, "where xxx indicates the bit length of n" is only an approximation. For example, n for K-233 is 232 bits long, and n for K-409 is 407 bits long. The correct statement is probably "where xxx indicates the bit length of the field elements" or "where xxx indicates the bit length of the field size".

27. Page 34, Sec. 6.1.1: in the third paragraph, the last sentence should read "If the output of the hash function is longer than ..." or "If the length of the output of the hash function is greater ...".

28. Page 35, Sec. 6.2: "that is associated" should be "that are associated".

29. Page 35, Sec. 6.3: in the second paragraph, the exponent -1 in k^-1 (except the first instance) is a little too low. Remove "with respect multiplication", which is redundant with "multiplicative".

30. Page 35, Sec. 6.3: in the third paragraph, change "computation" to "the computations".

31. Page 35, Sec. 6.4: in item 2, it's better to refer to Section 6.2.1 instead of Section 6.2.

32. Page 42, Sec. A.1.1.4: in the last paragraph, the current version of Knuth's book is The Art of Computer Programming, Vol. 2, 3rd Ed., Addison-Wesley, 1998, Algorithm P, page 395.

33. Page 43. Sec. A.1.1.4: the first paragraph specifies that iterations >= 50 based on the 1/4 upper bound of the error probability. However, Handbook of Applied Cryptography by A. Menezes, P. van Oorschot, and S. Vanstone, Chapter 4 cites tigher upper bounds on the error probability that allow us to reduce the required number of iterations significantly. Can we use the smaller number of iterations given in the Handbook of Applied Cryptography? See Handbook of Applied Cryptography, Chapter 4, Sections 4.48 and 4.49, pages 148-149. (The book's chapters can be downloaded from http://www.cacr.math.uwaterloo.ca/hac/)

34. Page 46, Sec. A.1.2.1.1: under "Process", step 4 says "Get an arbitrary sequence of seedlen bits as first seed". Please clarify whether the most significant bit of the bit sequence must be 1.

In CMVP's DSA Validation System, our experiments showed that if the most significant byte of SEED is 0, that SEED value will fail the PQG Domain Parameter Generation Test, even though FIPS 186-2 Appendix 2.2 says SEED is an arbitrary sequence of at least 160 bits. I hope FIPS 16-3 can clarify this point.

Date: Wed, 14 Jun 2006 11:11:45 -0400
From: "Savard, Stephen M." Stephen.Savard@cse-cst.gc.ca

Editorial Comments

1	Page 1, second last paragraph	Change "FIPS approved digital signature …" to "FIPS Approved digital signature …".
2	Page 11	In the bit string definition, change "0 s and 1 s" to "0's and 1's".
3	Page 18	The diagram can be made to look better by making the diagram more symmetric. Under Signature Verification, move the Message/Data text to the right by two letters so it looks the same (with respect to the arrow underneath) as the Message/Data text under Signature Generation.
4	Page 25 section 4.3, page 26 section 4.4, page 37 Appendix A, page 57 section B.1	Change (p, q, g {, domain_parameter_seed, counter}) to (p, q, g, {domain_parameter_seed, counter})
5	Pages 33-34	Fix Table 1 to appear on the same page
6	Page 86 section D.4	Change "Y_0, Y_0 + 1, , Y_0 + J" to "Y_0, Y_0 + 1, … , Y_0 + J". This occurs again in the same section in the middle of page 87. Use consistent notation when describing sequences. This notation is used for example in section D.2.1
7	Page 87 section D.5	Add a space after + to change "security_strength +21" to "security_strength + 21".
8	Page 89 section D.6	There is a typo in section 2 which has a comma after 17 at the end of a sequence. It should read {5, -7, 9, -11, 13, -15, 17}.
9	Page 89 section D.6	In step 4, change "K_rK_{r-1} K_0" to "K_rK_{r-1} … K_0".
10	Page 90 section D.7	The value a has a comma after the last number in the sequence and should read {5, -7, 9, -11, 13, -15, 17}. Also try to put "-15" on the same line.
11	Page 91 last sentence	Be consistent in either adding spaces or no spaces in between the equations, especially terms like -1. This should be consistent throughout the whole document.