

FIPS 186-3, DIGITAL SIGNATURE STANDARD  
PROPOSED REVISIONS

U.S. DEPARTMENT OF COMMERCE  
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY  
Gaithersburg, MD 20899

DATE OF CHANGE: 2012 Month Day

Federal Information Processing Standard (FIPS) 186-3, Digital Signature Standard, specifies three techniques for the generation and verification of digital signatures that can be used for the protection of data: the Digital Signature Algorithm (DSA), the Elliptic Curve Digital Signature Algorithm (ECDSA) and the Rivest-Shamir-Adelman (RSA) algorithm. FIPS 186-3 is used in conjunction with the hash functions specified in FIPS 180-4, Secure Hash Standard (SHS).

The following revisions to FIPS 186-3 are proposed:

## 1. The Use of Random Bit/Number Generators

FIPS 186-3 makes several references to **approved** random number generators (RNGs) and **approved** random bit generators (RBGs) throughout the Standard. In some cases, FIPS 186-3 specifically states that NIST SP 800-90 Deterministic Random Bit Generators (DRBGs) are to be used. In other cases, FIPS 186-3 states that an **approved** RNG is to be used, with a parenthetical reference to NIST SP 800-90. The motivation for these statements was to encourage adoption of the more secure NIST SP 800-90 DRBGs. Note that SP 800-90 has been revised to be SP 800-90A, and further references will be to SP 800-90A, rather than to SP 800-90. Also, note that the **approved** RNGs specified in FIPS 140-2, Annex C, except those in NIST SP 800-90A, are currently deprecated and will be disallowed after December 2015, as specified in NIST SP 800-131A.

With regard to the use of random bit/number generators by this Standard:

*Any random number generator (RNG) or random bit generator (RBG) that is **approved** for use in FIPS 140-validated modules may be used, subject to the transition schedule specified in SP 800-131A. Specific references to SP 800-90A should not be considered as a requirement to use a generator specified in SP 800-90A until such time as the use of the other generators is no longer allowed.*

This change is intended to accommodate transition issues regarding transitioning from the validation of FIPS 186-2 to FIPS 186-3.

Note that when randomly or pseudorandomly-generated numbers or integers are specified in the Standard, a conversion from random (or pseudorandom) bits is required.

## 2. Definition Clarification

Several of the terms in the Standard require more precise definitions.

- a) “Random number generator”: Change the definition to “*See random bit*”

*generator.”*

- b) Insert the following definition for “*Random bit generator*”:

*A device or algorithm that can produce a sequence of random or pseudorandom bits that appears to be statistically independent and unbiased.*

- c) Insert the following definition for “*Random number (or integer, prime, seed, or value)*”:

*A random or pseudorandom “item” that is determined using the output of an **approved** random bit generator and possibly an **approved** method for transforming the bits output by the generator to meet the criteria for that “item” (e.g., convert bits into integers or use the bits to “find” a prime number).*

- d) Insert the following definition for “*Randomly generated*”:

*Randomly or pseudorandomly generated; i.e., generated using an **approved** random bit generator.*

### **3. The Reuse of a Prime Number Generation Seed for RSA Key Pair Generation**

Section 5.1 contains the following statement in the last paragraph:

*If the prime number generation seeds are retained, they **shall** only be used as evidence that the generated values (i.e.,  $p$  and  $q$ ) were determined in an arbitrary manner, and the seeds **shall** be protected in a manner that is (at least) equivalent to the protection required for the private key.*

This change notice specifies the following change to this statement:

*If any prime number generation seed is retained (e.g., to regenerate the RSA modulus  $n$ , or as evidence that the generated prime factors (i.e.,  $p$  and  $q$ ) were determined in an arbitrary manner), then the seed **shall** be kept secret and **shall** be protected in a manner that is (at least) equivalent to the protection required for the associated private key.*

### **4. Methods Used for the Generation of $k$**

Appendices B.2 and B.5 have the following requirement:

“Two methods are provided for the generation of  $k$ ; one of these two methods **shall** be used.”

This change notice specifies the following change to both appendices (the underlined text is the change):

“Two methods are provided for the generation of  $k$ ; one of these two methods or another **approved** method **shall** be used.”

### **5. Processing Step Error in the Secret Number Generation for ECDSA**

In Appendices B.5.1 and B.5.2, processing step 1 (i.e.,  $N = \text{len}(q)$ ) is incorrect. This change notice specifies the following change to step 1: “ $N = \text{len}(n)$ ,” ; i.e., “ $q$ ” is changed to “ $n$ ”.

This change may be significant if the cofactor is greater than one; for the NIST-recommended curves, the cofactor is one, so in this case, both values produce the same value for  $N$ .

## 6. Criteria for IFC Key Pairs

Appendix B.3.1, item A, has the following statement:

“Using these methods, primes of 2048 or 3072 bits may be generated; primes of 1024 bits **shall not** be generated using these methods. Primes of 1024 bits **shall be** generated using conditions based on auxiliary primes (see Appendices B.3.4, B.3.5, or B.3.6).”

This change notice makes the following change to the above statement (the changed text is underlined):

“Using these methods,  $p$  and  $q$  with lengths of 1024 or 1536 bits may be generated;  $p$  and  $q$  with lengths of 512 bits **shall not** be generated using these methods. Instead,  $p$  and  $q$  with lengths of 512 bits **shall be** generated using conditions based on auxiliary primes (see Appendices B.3.4, B.3.5, or B.3.6).”

## 7. Salt Length for RSASSA-PSS

PKCS #1, versions 2.1, contains the statement:

“If  $emLen < hLen + sLen + 2$ , output “encoding error” and stop”,

where  $emLen$  is  $\lceil (\text{modulus\_length} - 1)/8 \rceil$ ,  $hLen$  is the length of the output block of a hash function (in octets), and  $sLen$  is the length of a salt (in octets). Typical salt lengths in octets are  $hLen$  and 0.

Section 5.5 of FIPS 186-3, item e, contains the statement;

“For RSASSA-PSS, the length of the salt ( $sLen$ ) **shall be**:  $0 \leq sLen \leq hLen$ , where  $hLen$  is the length of the hash function output block (in bytes or octets).”

These statements are consistent with the 2048 and 3072-bit moduli for all **approved** hash functions. However, when a 1024-bit modulus is used with SHA-512 and a salt length equal to  $hLen$  (512 bits = 64 octets, in this case), then:

$$emLen = \lceil (1024 - 1)/8 \rceil = 128 \text{ octets,}$$

$$hLen = sLen = 64 \text{ octets, and}$$

$hLen + sLen + 2 = 130$ , which is greater than  $emLen$ , so the process produces an error (see the statement in PKCS #1 that is provided above).

Therefore, this change notice makes the following change to the statement in item e) of Section 5.5 in FIPS 186-3:

“For RSASSA-PSS:

If  $nlen = 1024$  bits, and the output length of the **approved** hash function output block is 512 bits, then the length of the salt **shall** be  $0 \leq sLen \leq hLen - 2$ .

Otherwise, the length of the salt ( $sLen$ ) **shall** be:  $0 \leq sLen \leq hLen$

where  $hlen$  is the length of the hash function output block (in bytes or octets).”

**6. Changes to the Referenced Documents for Item 14 of the Announcement:**

- i. Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators.
- l. Special Publication (SP) 131A, Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths.