4 The Digital Signature Algorithm (DSA)

4.1 DSA Parameters

A DSA digital signature is computed using a set of domain parameters, a private key x, a per-message secret number k, data to be signed, and a hash function. A digital signature is verified using the same domain parameters, a public key y that is mathematically associated with the private key x used to generate the digital signature, data to be verified, and the same hash function that was used during signature generation. These parameters are defined as follows:

p a prime modulus, where $2^{L-1} < p < 2^L$, and L is the bit length of p. Values for L are provided in Section 4.2.

q a prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$, and N is the bit length of q. Values for N are provided in Section 4.2.

g a generator of the subgroup of order q mod p, such that $1 < g < p$.

x the private key that must remain secret; x is a randomly or pseudorandomly generated integer, such that $0 < x < q$, i.e., x is in the range $[1, q-1]$.

y the public key, where $y = g^x \bmod p$.

k a secret number that is unique to each message; k is a randomly or pseudorandomly generated integer, such that $0 < k < q$, i.e., k is in the range $[1, q-1]$.

4.2 Selection of Parameter Sizes and Hash Functions for DSA

This Standard specifies the following choices for the pair L and N (the bit lengths of p and q, respectively):

$L = 1024$, $N = 160$

$L = 2048$, $N = 224$

$L = 2048$, $N = 256$

$L = 3072$, $N = 256$

Federal Government entities **shall** generate digital signatures using use one or more of these choices.

An approved hash function, as specified in FIPS 180-3, **shall** be used during the generation of digital signatures. The security strength associated with the DSA digital signature process is no greater than the minimum of the security strength of the (L, N) pair and the security strength of the hash function that is employed. Both the security strength of the hash function used and the security strength of the (L, N) pair **shall** meet or exceed the security strength required for the digital signature process. The security strength for each (L, N) pair and hash function is provided

in SP 800-57.

SP 800-57 provides information about the selection of the appropriate (*L*, *N*) pair in accordance with a desired security strength for a given time period for the generation of digital signatures. An (*L*, *N*) pair **shall** be chosen that protects the signed information during the entire expected lifetime of that information. For example, if a digital signature is generated in 2009 for information that needs to be protected for five years, and a particular (*L*, *N*) pair is invalid after 2010, then a larger (*L*, *N*) pair **shall** be used that remains valid for the entire period of time that the information needs to be protected.

It is recommended that the security strength of the (*L*, *N*) pair and the security strength of the hash function used for the generation of digital signatures be the same unless an agreement has been made between participating entities to use a stronger hash function. When the length of the output of the hash function is greater than *N* (i.e., the bit length of *q*), 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. A hash function that provides a lower security strength than the (*L*, *N*) pair ordinarily **should not** be used, since this would reduce the security strength of the digital signature process to a level no greater than that provided by the hash function.

A Federal Government entity other than a Certification Authority (CA) **should** use only the first three (*L*, *N*) pairs (i.e., the (1024, 160), (2048, 224) and (2048, 256) pairs). A CA **shall** use an (*L*, *N*) pair that is equal to or greater than the (*L*, *N*) pairs used by its subscribers. For example, if subscribers are using the (2048, 224) pair, then the CA **shall** use either the (2048, 224), (2048, 256) or (3072, 256) pair. Possible exceptions to this rule include cross certification between CAs, certifying keys for purposes other than digital signatures and transitioning from one key size or algorithm to another. See SP 800-57 for further guidance.

4.3 DSA Domain Parameters

DSA requires that the private/public key pairs used for digital signature generation and verification be generated with respect to a particular set of domain parameters. These domain parameters may be common to a group of users and may be public. A user of a set of domain parameters (i.e., both the signatory and the verifier) **shall** have assurance of their validity prior to using them (see Section 3). Although domain parameters may be public information, they **shall** be managed so that the correct correspondence between a given key pair and its set of domain parameters is maintained for all parties that use the key pair. A set of domain parameters may remain fixed for an extended time period.

The domain parameters for DSA are the integers *p*, *q*, and *g*, and optionally, the *domain_parameter_seed* and *counter* that were used to generate *p* and *q* (i.e., the full set of domain parameters is (*p*, *q*, *g* {, *domain_parameter_seed*, *counter*})).

4.3.1 Domain Parameter Generation

Domain parameters may be generated by a trusted third party (a TTP, such as a CA) or by an entity other than a TTP. Assurance of domain parameter validity **shall** be obtained prior to key pair generation, digital signature generation or digital signature verification (see Section 3).

The integers p and q **shall** be generated as specified in Appendix A.1. The input to the generation process is the selected values of L and N (see Section 4.2); the output of the generation process is the values for p and q, and optionally, the values of the *domain_parameter_seed* and *counter*.

The generator g **shall** be generated as specified in Appendix A.2.

The security strength of a hash function used during the generation of the domain parameters **shall** meet or exceed the security strength associated with the (L, N) pair. Note that this is more restrictive than the hash function that can be used for the digital signature process (see Section 4.2).

4.3.2 Domain Parameter Management

Each digital signature key pair **shall** be correctly associated with one specific set of domain parameters (e.g., by a public key certificate that identifies the domain parameters associated with the public key). The domain parameters **shall** be protected from unauthorized modification until the set is deactivated (if and when the set is no longer needed). The same domain parameters may be used for more than one purpose (e.g., the same domain parameters may be used for both digital signatures and key establishment). However, using different values for the generator g reduces the risk that key pairs generated for one purpose could be accidentally used (successfully) for another purpose.

4.4 Key Pairs

Each signatory has a key pair: a private key x and a public key y that are mathematically related to each other. The private key **shall** be used for only a fixed period of time (i.e., the private key cryptoperiod) in which digital signatures may be generated; the public key may continue to be used as long as digital signatures that were generated using the associated private key need to be verified (i.e., the public key may continue to be used beyond the cryptoperiod of the associated private key). See SP 800-57 for further guidance.

4.4.1 DSA Key Pair Generation

A digital signature key pair x and y is generated for a set of domain parameters (p, q, g {, *domain_parameter_seed*, *counter*}). Methods for the generation of x and y are provided in Appendix B.1.

4.4.2 Key Pair Management

Guidance on the protection of key pairs is provided in SP 800-57. The secure use of digital signatures depends on the management of an entity's digital signature key pair as follows:

1. The validity of the domain parameters **shall** be assured prior to the generation of the key pair, or the verification and validation of a digital signature (see Section 3).
2. Each key pair **shall** be associated with the domain parameters under which the key pair was generated.
3. A key pair **shall** only be used to generate and verify signatures using the domain parameters associated with that key pair.
4. The private key **shall** be used only for signature generation as specified in this Standard and **shall** be kept secret; the public key **shall** be used only for signature verification and may be made public.
5. An intended signatory **shall** have assurance of possession of the private key prior to or concurrently with using it to generate a digital signature (see Section 3.1).
6. A private key **shall** be protected from unauthorized access, disclosure and modification.
7. A public key **shall** be protected from unauthorized modification (including substitution). For example, public key certificates that are signed by a CA may provide such protection.
8. A verifier **shall** be assured of a binding between the public key, its associated domain parameters and the key pair owner (see Section 3).
9. A verifier **shall** obtain public keys in a trusted manner (e.g., from a certificate signed by a CA that the entity trusts, or directly from the intended or claimed signatory, provided that the entity is trusted by the verifier and can be authenticated as the source of the signed information that is to be verified).
10. Verifiers **shall** be assured that the claimed signatory is the key pair owner, and that the owner possessed the private key that was used to generate the digital signature at the time that the signature was generated (i.e., the private key that is associated with the public key that will be used to verify the digital signature) (see Section 3.3).
11. A signatory and a verifier **shall** have assurance of the validity of the public key (see Sections 3.1 and 3.3).

4.5 DSA Per-Message Secret Number

A new secret random number k **shall** be generated prior to the generation of each digital signature for use during the signature generation process. This secret number **shall** be protected from unauthorized disclosure and modification.

k^{-1} is the multiplicative inverse of k with respect to multiplication modulo q; i.e., $0 < k^{-1} < q$

and $1 = (k^{-1}\ k) \bmod q$. This inverse is required for the signature generation process (see Section 4.6). A technique is provided in Appendix C.1 for deriving k^{-1} from k.

k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

Methods for the generation of the per-message secret number are provided in Appendix B.2.

4.6 DSA Signature Generation

The intended signatory **shall** have assurances as specified in Section 3.1.

Let N be the bit length of q. Let **min**(N, *outlen*) denote the minimum of the positive integers N and *outlen*, where *outlen* is the bit length of the hash function output block.

The signature of a message M consists of the pair of numbers r and s that is computed according to the following equations:

$r = (g^k \bmod p) \bmod q$.

z = the leftmost **min**(N, *outlen*) bits of **Hash**(M).

$s = (k^{-1}(z + xr)) \bmod q$.

When computing s, the string z obtained from **Hash**(M) **shall** be converted to an integer. The conversion rule is provided in Appendix C.2.

Note that r may be computed whenever k, p, q and g are available, e.g., whenever the domain parameters p, q and g are known, and k has been pre-computed (see Section 4.5), r may also be pre-computed, since knowledge of the message to be signed is not required for the computation of r. Pre-computed k, k^{-1} and r values **shall** be protected in the same manner as the the private key x until s has been computed (see SP 800-57).

The values of r and s **shall** be checked to determine if $r = 0$ or $s = 0$. If either $r = 0$ or $s = 0$, a new value of k **shall** be generated, and the signature **shall** be recalculated. It is extremely unlikely that $r = 0$ or $s = 0$ if signatures are generated properly.

The signature (r, s) may be transmitted along with the message to the verifier.

4.7 DSA Signature Verification and Validation

Signature verification may be performed by any party (i.e., the signatory, the intended recipient or any other party) using the signatory's public key. A signatory may wish to verify that the computed signature is correct, perhaps before sending the signed message to the intended recipient. The intended recipient (or any other party) verifies the signature to determine its

authenticity.

Prior to verifying the signature of a signed message, the domain parameters, and the claimed signatory's public key and identity **shall** be made available to the verifier in an authenticated manner. The public key may, for example, be obtained in the form of a certificate signed by a trusted entity (e.g., a CA) or in a face-to-face meeting with the public key owner.

Let M', r', and s' be the received versions of M, r, and s, respectively; let y be the public key of the claimed signatory; and let N be the bit length of q. Also, let **min**(N, *outlen*) denote the minimum of the positive integers N and *outlen*, where *outlen* is the bit length of the hash function output block.

The signature verification process is as follows:

1. The verifier **shall** check that $0 < r' < q$ and $0 < s' < q$; if either condition is violated, the signature **shall** be rejected as invalid.
2. If the two conditions in step 1 are satisfied, the verifier computes the following:

   $w = (s')^{-1} \bmod q$.

   z = the leftmost **min**(N, *outlen*) bits of **Hash**(M').

   $u1 = (zw) \bmod q$.

   $u2 = ((r')w) \bmod q$.

   $v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q$.

   A technique is provided in Appendix C.1 for deriving $(s')^{-1}$ (i.e., the multiplicative inverse of s' mod q).

   The string z obtained from **Hash**(M') **shall** be converted to an integer. The conversion rule is provided in Appendix C.2.
3. If $v = r'$, then the signature is verified. For a proof that $v = r'$ when $M' = M$, $r' = r$, and $s' = s$, see Appendix E.
4. If v does not equal r', then the message or the signature may have been modified, there may have been an error in the signatory's generation process, or an imposter (who did not know the private key associated with the public key of the claimed signatory) may have attempted to forge the signature. The signature **shall** be considered invalid. No inference can be made as to whether the data is valid, only that when using the public key to verify the signature, the signature is incorrect for that data.
5. Prior to accepting the signature as valid, the verifier **shall** have assurances as specified in Section 3.3.

An organization's policy may govern the action to be taken for invalid digital signatures. Such policy is outside the scope of this Standard. Guidance about determining the timeliness of

digitally signed messages is addressed in SP 800-102, Recommendation for Digital Signature Timeliness.