Appendix

Generation of Primes
This appendix includes algorithms for generating the primes p and q used in the DSA. These algorithms require a random number generator, and an efficient modular exponentiation algorithm. Generation of p and q shall be performed as specified in this appendix, or using other FIPS approved security methods.

RABIN – MILLER PROBABILISTIC PRIMALITY TEST

In order to generate the primes p and q, a primality test is required. There are several fast probabilistic algorithms available. The following algorithm is a simplified version of a procedure due to M.O. Rabin, based in part on ideas of Gary L. Miller. [See Knuth, The Art of Computer Programming, Vol. 2, Addison-Wesley, 1981, Algorithm P, page 379.] If this algorithm is iterated n times, it will produce a false prime with probability no greater than $1/4^n$. Therefore, $n \geq 50$ will give an acceptable probability of error. To test whether an integer is prime:

Step 1. Set $i = 1$ and $n \geq 50$.

Step 2. Set $w$ = the integer to be tested, $w = 1 + 2^a m$, where m is odd and $2^a$ is the largest power of 2 dividing $w - 1$.

Step 3. Generate a random integer $b$ in the range $1 < b < w$. Step 4. Set $j = 0$ and $z = b^m \mod w$.

Step 5. If $j = 0$ and $z = 1$, or if $z = w - 1$, go to step 9. Step 6. If $j > 0$ and $z = 1$, go to step 8.

Step 7. $j = j + 1$. If $j < a$, set $z = z^2 \mod w$ and go to step 5.

Step 8. $w$ is not prime. Stop.

Step 9. If $i < n$, set $i = i + 1$ and go to step 3. Otherwise, $w$ is probably prime.
GENERATION OF PRIMES

The DSA requires two primes, p and q, satisfying the following three conditions:

a. \(2^{159} < q < 2^{160}\)

b. \(2^{L-1} < p < 2^L\) for a specified \(L\), where \(L = 512 + 64j\) for some \(0 \leq j \leq 8\)

c. \(q\) divides \(p - 1\)

This prime generation scheme starts by using the SHA-1 and a user supplied SEED to construct a prime, \(q\), in the range \(2^{159} < q < 2^{160}\). Once this is accomplished, the same SEED value is used to construct an \(X\) in the range \(2^{L-1} < X < 2^L\). The prime, \(p\), is then formed by rounding \(X\) to a number congruent to 1 mod \(2q\) as described below.

An integer \(x\) in the range \(0 \leq x < 2^g\) may be converted to a \(g\)-long sequence of bits by using its binary expansion as shown below:

\[
x = x_1*2^{g-1} + x_2*2^{g-2} + ... + x_{g-1}*2 + x_g \rightarrow \{ x_1,...,x_g \}.
\]

Conversely, a \(g\)-long sequence of bits \(\{ x_1,...,x_g \}\) is converted to an integer by the rule

\[
\{ x_1,...,x_g \} \rightarrow x_1*2^{g-1} + x_2*2^{g-2} + ... + x_{g-1}*2 + x_g.
\]

Note that the first bit of a sequence corresponds to the most significant bit of the corresponding integer and the last bit to the least significant bit.

Let \(L - 1 = n*160 + b\), where both \(b\) and \(n\) are integers and \(0 \leq b < 160\).

Step 1. Choose an arbitrary sequence of at least 160 bits and call it SEED.

Let \(g\) be the length of SEED in bits.

Step 2. Compute \(U = SHA-1[SEED] XOR SHA-1[(SEED+1) mod 2^g]\).
Step 3. Form q from U by setting the most significant bit (the $2^{159}$ bit) and the least significant bit to 1. In terms of boolean operations, $q = U \ OR \ 2^{159} \ OR \ 1$. Note that $2^{159} < q < 2^{160}$.

Step 4. Use a robust primality testing algorithm to test whether q is prime$^1$.

Step 5. If q is not prime, go to step 1.

Step 6. Let counter = 0 and offset = 2.

Step 7. For k = 0,...,n let

$$V_k = \text{SHA-1}[(\text{SEED} + \text{offset} + k) \mod 2^g].$$

$^1$A robust primality test is one where the probability of a non-prime number passing the test is at most $2^{-80}$.

Step 8. Let W be the integer

$$W = V_0 + V_1 \times 2^{160} + ... + V_{n-1} \times 2^{(n-1) \times 160} + (V_n \mod 2^b) \times 2^{n \times 160}$$

and let $X = W + 2^{L-1}$. Note that $0 \leq W < 2^{L-1}$ and hence $2^{L-1} \leq X < 2^L$.

Step 9. Let $c = X \mod 2q$ and set $p = X - (c - 1)$. Note that $p$ is congruent to 1 mod 2q.

Step 10. If $p < 2^{L-1}$, then go to step 13.


Step 12. If p passes the test performed in step 11, go to step 15.

Step 13. Let counter = counter + 1 and offset = offset + n + 1.

Step 14. If counter $\geq 2^{12} = 4096$ go to step 1, otherwise (i.e. if counter < 4096) go to step 7.

Step 15. Save the value of SEED and the value of counter for use in certifying the proper generation of p and q.