E.4 DRBGs Based on Hard Problems

The **Dual_EC_DRBG** and **MS_DRBG** base their 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 MS_DRBG, which is 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. 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 servers 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 numbers.

E.4.1 Implementation considerations.

E.4.1.1 Dual_EC_DRBG

Random is produced in blocks of bits representing the x-coordinates on an elliptic curve. Because of the various security levels allowed by this Standard there are multiple curves offered, with differing block sizes. The size is always a multiple of 8, about 16 bits less than a curve's underlying field size. Blocks are concatenated, then truncated if necessary, to fullfil a request for any number of bits. (Up to a maximum per call of 10,000 times the block length. The smallest blocksize is 216, meaning at least 2M bits can be requested on each call.)

An important detail concerning Dual_EC_DRBG is that every call for random, whether it be for 2 million bits or a single bit, requires that at least one full block of random be produced: no unused bits are saved internally from the previous call. Each block produced requires two point multiplications on an elliptic curve—a fair amount of computation. Applications such as IKE and SSL are encouraged to aggregate all their needs for random bits into a single call to Dual_EC_DRBG, then parcel out the bits as required during the protocol exchange. A C structure, for example, is an ideal vehicle for this.

To avoid unnecessarily complex implementations it should be noted that *every* curve in the standard need not be available to an application. For instance, one may choose to do arithmetic only over the prime order fields in a software application, or perhaps a particular binary curve in a hardware application. To improve efficiency there has been much research done on the implementation of elliptic curve arithmetic; descriptions and source code are available in the open literature.

As a final comment on implementation of the Dual_EC_DRBG, note that having fixed base points offers a distinct advantage for optimization. Tables can be precomputed which allow nP to be attained as a series of point additions, resulting in an 8 to 10-fold speedup, or more, if space permits.

E4.1.2. Micali-Schnorr

Micali-Schnorr was invented to be a more efficient version of the predecessor algorithm Blum-Blum-Shub DRBG. The latter uses the recursion $x_i = x_{i-1}^2 \bmod n$ to generate its state sequence, producing a single bit of random as the least significant bit of x_i. Later it was shown that O(ln(ln n))) bits could be taken on each iteration, but this is still a very small percentage of those produces. The MS generator allows a much larger percentage of n bits to be used on each iteration, and has an additional advantage in that no output bits are used to propagate the sequence. (It does, however, rely on a stronger assumption for its security than the intractability of integer factorization.)

As the X9.82 standard evolved, committee members argued for restricting the number of bits generated on each exponentiation to O(ln(ln n)) *hard* bits, as is done in BBS. The result is that the efficiency argument for choosing MS over BBS doesn't apply. Nonetheless, a user does have more options in the choice of parameters.

Micali_Schnorr offers an alternative to Dual_EC_DRBG in the class of algorithms based on a hard problem from number theory, and presents an advantage in its simplicity. All that's required for implementation is a routine which computes $x^e \bmod n$; this can be readily found in commercial and open source toolkits.