G.4 DRBGs Based on Hard Problems

The **Dual_EC_DRBG** generates pseudorandom outputs by extracting bits from elliptic curve points. The secret, internal state of the DRBG is a value S that is the x-coordinate of a point on an elliptic curve. Outputs are produced by first computing R to be the x-coordinate of the point $S*P$ and then extracting low order bits from the x-coordinate of the elliptic curve point $R*Q$.

Security. The security of **Dual_EC_DRBG** is based on the so-called "Elliptic Curve Discrete Logarithm Problem" that has no known attacks better than the so-called "meet-in-the-middle" attacks. For an elliptic curve defined over a field of size 2^m, the work factor of these attacks is approximately $2^{m/2}$ so that solving this problem is computationally infeasible for the curves in this document. The **Dual_EC_DRBG** is the only DRBG in this document whose security is related to a hard problem in Number Theory.

Constraints. For any one of the three elliptic curves, a particular instance of **Dual_EC_DRBG** may generate at most 2^{32} output blocks before reseeding. Since the sequence of output blocks are expected to cycle in approximately sqrt(n) bits (where n is the (prime) order of the particular elliptic curve being used), this is quite a conservative reseed interval for any one of the there possible curves.

Performance. Due to the elliptic curve arithmetic involved in this DRBG, this algorithm generates pseudorandom bits more slowly than the other DRBGS in this document. It should be noted, however, that the design of this algorithm allows for certain performance-enhancing possibilities. First, note that the use of fixed base points allows one to substantially increase the performance of this DRBG via the use of tables. By storing multiples of the points P and Q, the elliptic curve multiplication can be accomplished via point additions rather than multiplications, a much less expensive operation. In more constrained environments when table storage is not an option, the use of so-called Montgomery Coordinates of the form $(X : Z)$ can be used as a method to increase performance since the y-coordinates of the computed points are not required. A given implementation of this DRBG need not include all three of the NIST-approved curves. Once the designer decides upon the strength required by a given application, he can then choose to implement the single curve that most appropriately meets this requirement. For a common level of optimization expended, the higher strength curves will be slower and tend toward less efficient use of output blocks. To mitigate the latter, the designer should be aware that every distinct request for random bits, whether for two million bits or a single bit, requires the computational expense of at least two elliptic curve point multiplications. Applications requiring large blocks of random bits (such as IKE or SSL), can thus be implemented most efficiently by first making a single call to the DRBG for all the required bits and then appropriately partitioning these bits as required by the protocol. For applications that already have hardware or software support for elliptic curve arithmetic, this DRBG is a natural choice as it allows the designer to utilize existing capabilities to generate truly high-security random numbers.

Resources. Any entropy input source may be used with **Dual_EC_DRBG**, provided that it is capable of generating at least *seedlen* bits. This DRBG also requires an appropriate hash function (see Table 4) that is used exclusively for producing an appropriately-sized initial state from the entropy input at instantiation or reseeding. An implementation of this DRBG must also have enough storage for the internal state (see 10.3.1.1). Some optimizations require additional storage for moderate to large tables of pre-computed values.

Algorithm Choices. The choice of appropriate elliptic curves and points used by **Dual_EC_DRBG** is discussed in Appendix A.1.