November 28, 2007

Dear Bruce,

In your November 14, 2007 Wired commentary (http://www.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115), you suggested that the Dual_EC_DRBG random number generator published in NIST Special Publication 800-90 has a property "that can only be described as a back door." We have no evidence that anyone has, or will ever have, the "secret numbers" for the back door that were hypothesized by mathematicians Dan Shumow and Neils Ferguson, that would provide advance information on the random numbers generated by the algorithm. For this reason, we are not withdrawing the algorithm at this time. NIST Special Publication 800-90, which includes a method for randomly generating points if there is a concern about a back door, underwent a rigorous review process that included a public comment period before it was published,. All NIST algorithms, including the Dual_EC_DRBG, undergo continual review throughout their lifetime. If successful attacks are found on an algorithm, the algorithm is withdrawn.

Sincerely,

Elaine Barker
Lead Author, NIST SP 800-90
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD

A.2.1 Generating Alternative *P,Q*

The curve **shall** be one of the NIST curves from FIPS 186-3 that is specified in Appendix A.1 of this Recommendation, and **shall** be appropriate for the desired *security_strength*, as specified in Table 4, Section 10.3.1.

The points *P* and *Q* **shall** be valid base points for the selected elliptic curve that are generated to be verifiably random using the procedure specified in ANS X9.62. The following input is required for each point:

> An elliptic curve $E = (F_p, a, b)$, cofactor *h*, prime *n*, a bit string *domain_parameter_seed*[1], and hash function **Hash**(). The curve parameters are given in Appendix A.1 of this Recommendation. The *domain_parameter_seed* **shall** be different for each point, and the minimum length *m* of each *domain_parameter_seed* **shall** conform to Section 10.3.1, Table 4, under "Seed length". The bit length of *domain_parameter_seed* may be larger than *m*. The hash function **shall** be SHA-512 in all cases.

The *domain_parameter_seed* **shall** be different for each point *P* and Q. A domain parameter seed **shall not** be the seed used to instantiate a DRBG. The domain parameter seed is an arbitrary value that may, for example, be determined from the output of a DRBG.

If the output from the ANS X9.62 generation procedure is "failure", a different *domain_parameter_seed* **shall** be used for the point being generated.

Otherwise, the output point from the generate procedure in ANS X9.62 **shall** be used.

[EBB: Does this take care of the required relationship $Q = aP$ that is stated in Section 10.3?]

A.2.2 Additional Self-testing Required for Alternative *P,Q*

To insure that the points *P* and *Q* have been generated appropriately, additional self-test procedures **shall** be performed whenever the instantiate function is invoked. Section 11.3.1 specifies that known-answer tests on the instantiate function be performed prior to creating an operational instantiation. As part of these tests, an implementation of the generation procedure in ANS X9.62 **shall** be called for each point (i.e., *P* and *Q*) with the appropriate *domain_parameter_seed* value that was used to generate that point. The point returned **shall** be compared with the corresponding stored value of the point. If the generated value does not match the stored value, the implementation **shall** halt with an error condition.

[1] Called a *SEED* in ANS X9.62.

E.1~~.1~~ Potential Bias Due to Modular Arithmetic for Curves Over F_p

Given an integer x in the range 0 to 2^N-1, the r^{th} bit of x depends solely upon whether $\left\lfloor \frac{x}{2^r} \right\rfloor$ is odd or even. If all of the values in this range are sampled uniformly, the r^{th} bit will be 0 exactly ½ of the time. But if x is restricted to F_P, i.e., to the range 0 to p-1, this statement is no longer true.

By excluding the $k = 2^N - p$ values $p, p+1, \ldots, 2^N - 1$ from the set of all integers in Z_N, the ratio of ones and zeroes in the r^{th} bit is altered from $2^{N-1} / 2^{N-1}$ to a value that can be no smaller than $(2^{N-1} - k)/ 2^{N-1}$. For all the primes p used in this Recommendationt, $k/2^{N-1}$ is smaller than 2^{-31}. Thus, the ratio of ones and zeroes in any bit is within at least 2^{-31} of 1.0.

To detect this small difference from random, a sample of 2^{64} outputs is required before the observed distribution of 1's and 0's is more than one standard deviation away from flat random. This effect is dominated by the bias addressed below in section E.2.

Email note to interested parties:
A draft NIST Special Publication (Draft SP 800-90, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*) is available for public comment at http://csrc.nist.gov/publications/drafts.html. Comments should be submitted to ebarker@nist.gov by Wednesday, February 1, 2006. Please place "Comments on SP 800-90" in the subject line.

Instructions to Patrick:
Please place the following on the csrc page:

December 16, 2005: A draft NIST Special Publication (Draft SP 800-90, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*) is available[EBB1] for public comment. Comments should be submitted to ebarker@nist.gov by Wednesday, February 1, 2006. Please place "Comments on SP 800-90" in the subject line.

The draft document is attached.

Instructions to Larry Bassham:
Please place the following on the http://csrc.nist.gov/CryptoToolkit/tkrng.html page:

A draft NIST Special Publication (Draft SP 800-90, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*) is available[EBB2] for public comment. Comments should be submitted to ebarker@nist.gov by Wednesday, February 1, 2006. Please place "Comments on SP 800-90" in the subject line.

Patrick will be placing the document on the drafts page.

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.