Cryptanalysis of the Dual Elliptic Curve Pseudorandom Generator

Berry Schoenmakers and Andrey Sidorenko
Dept. of Mathematics and Computer Science, TU Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
berry@win.tue.nl, a.sidorenko@tue.nl

29 May 2006

1 Introduction

The Dual Elliptic Curve Pseudorandom Generator (DEC PRG) is proposed by Barker and Kelsey [2]. It is claimed (see Section 10.3.1 of [2]) that the pseudorandom generator is secure unless the adversary can solve the elliptic curve discrete logarithm problem (ECDLP) for the corresponding elliptic curve. The claim is supported only by an informal discussion. No security reduction is given, that is, it is not shown that an adversary that breaks the pseudorandom generator implies a solver for the ECDLP.

Our experimental results and also empirical argument show that the DEC PRG is *insecure*. The attack does *not* imply solving the ECDLP for the corresponding elliptic curve. The attack is very efficient. It can be run on an ordinary PC.

Actually, the generator is insecure because pseudorandom bits are extracted from points of the elliptic curve improperly. The authors of [2] assume that 240 least significant bits of x-coordinate of a random point of the elliptic curve over the prime field $\mathbb{F}_p$, where $\lceil \log_2 p \rceil = 256$, are indistinguishable from 240 uniformly distributed random bits. We show that this is not the case. Based on this observation, we construct an algorithm (an adversary) that efficiently distinguishes the pseudorandom sequences produced by the DEC PRG from the sequences of uniformly distributed random bits.

We note that the complexity of our attack is proportional to $2^{256-240} = 2^{16}$, so extracting less than 240 bits (say, 2^{176} bits) makes the attack impractical. However, extracting less random bits does *not* guarantee that there exists no other attack that successfully breaks the pseudorandom generator. The reason is that the DEC PRG is *not* provably secure, its security does *not* provably rely on the intractability of the ECDLP. To make a real provably secure pseudorandom generator one has to construct a security reduction, that is, to show that breaking the generator does imply solving a well-known and supposedly difficult problem (e.g., ECDLP, factoring, etc.)

In fact, provable security might be the only argument in favor of the relatively slow DEC PRG versus more efficient generators based on hash functions and block ciphers (e.g., the generators described in Sections 10.1 and 10.2 of [2]). Unfortunately, the DEC PRG is not secure so there are no reasons to use this generator rather than the others.

2 The Dual Elliptic Curve Pseudorandom Generator

Let $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. Let $\mathrm{E}(\mathbb{F}_p)$ denote the elliptic curve over $\mathbb{F}_p$ consisting of all pairs $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ such that

$$y^2 = x^3 + ax + b,$$

where

$$
\begin{aligned}
a &= 115792089210356248762697446949407573530086143415290314195533631308867097853948, \\
b &= 41058363725152142129326129780047268409114441015993725554835256314039467401291
\end{aligned}
$$

and a point at infinity $\mathcal{O}$. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points of the elliptic curve $\mathrm{E}(\mathbb{F}_p)$ such that

$$
\begin{aligned}
x_P &= 48439561293906451759052585252797914202762949526041747995844080717082404635286, \\
y_P &= 36134250956749795798585127919587881956611106672985015071877198253568414405109, \\
x_Q &= 91120319633256209954638481795610364441930342474826146651283703640232629993874, \\
y_Q &= 80764272623998874743522585409326200078679332703816718187804498579075161456710.
\end{aligned}
$$

The constants are taken from Appendix A.1 of [2]. Note that the constant α such that $P = \alpha Q$ is difficult to determine due to the intractability of the elliptic curve discrete logarithm problem (ECDLP).

The seed of the Dual Elliptic Curve pseudorandom generator (DEC PRG) is a random integer $s_0 \in_R \{0, 1, \ldots, \#\mathrm{E}(\mathbb{F}_p) - 1\}$, where $\#\mathrm{E}(\mathbb{F}_p)$ denotes the number of points on the curve. Let $\mathrm{x} : \mathrm{E}(\mathbb{F}_p) \mapsto \mathbb{F}_p$ denote a function that gives the x-coordinate of a point of the curve. Let $\mathrm{lsb}_i(s)$ denote i least significant bits of an integer s. For example $\mathrm{lsb}_3(23) = 7$, since $23 = (10111)_2$. The DEC PRG transforms the seed into the pseudorandom sequence of length $240k$, $k > 0$, as follows.

Algorithm 1 Dual Elliptic Curve pseudorandom generator

Input: $s_0 \in \{0, 1, \ldots, \#\mathrm{E}(\mathbb{F}_p) - 1\}$, $k > 0$
Output: $240k$ bits

```
for i = 1 to k do
    Set s_i ← x(s_{i-1} P)
    Set r_i ← lsb_240(x(s_i Q))
end for
Return r_1, ..., r_k
```

The authors of [2] claim that

> Backtracking resistance is built into the design, as knowledge of s_1 does not allow an adversary to determine s_0 (and so forth) unless the adversary is able to solve the ECDLP for that specific curve. In addition, knowledge of r_1 does not allow an adversary to determine s_1 (and so forth) unless the adversary is able to solve the ECDLP for that specific curve.

Note that backtracking (in other words, predicting) is equivalent to distinguishing the output of the pseudorandom generator from the sequence of uniformly distributed random bits [6]. In the next section we show that the output of the DEC PRG can be efficiently distinguished from the sequence of uniformly distributed random bits. The distinguishing attack does not imply solving the ECDLP for the given curve. It means that the pseudorandom generator is insecure and cannot be used for cryptographic purposes.

3 The Distinguishing Attack on the DEC PRG

The output of the pseudorandom generator consists of k 240-bit blocks. For a block $r \in \{0, 1\}^{240}$ let $\phi(r)$ denote the number of points T on the elliptic curve $\mathrm{E}(\mathbb{F}_p)$ such that $\mathrm{lsb}_{240}(\mathrm{x}(T)) = r$.

At each step i, the DEC PRG outputs $r_i = \mathrm{lsb}_{240}(\mathrm{x}(s_i Q))$. The generator is secure if and only if r_i is indistinguishable from 240 uniformly distributed random bits for all $i = 1, \ldots, k$. We will see, however, that r_i *can be distinguished* from 240 uniformly distributed random bits.

The argument of this section does not pretend to be a strict justification of the attack. On the contrary, it just gives the reader an intuition of how the attack works.

It is shown by Brown [3] that the sequence of points $s_i Q$ is indistinguishable from the sequence of points chosen uniformly at random under the assumption that the DDH problem and the non-standard x-logarithm problem are intractable in in $\mathrm{E}(\mathbb{F}_p)$. Therefore, it is reasonable to assume that $s_i Q$ behaves like a random point on the curve, $i = 1, \ldots, k$. Then, for $r \in \{0,1\}^{240}$ the probability that a certain output block is equal to r is $\phi(r)/\#\mathrm{E}(\mathbb{F}_p)$. Thus, if $\phi(r_1) > \phi(r_2)$ for $r_k \in \{0,1\}^{240}, k = 1, 2,$ the probability that a certain output block is equal to r_1 is higher than the probability that this block is equal to r_2. Moreover, we will see that the difference between the probabilities is *observable*.

The number of points on the elliptic curve is close to 2^{256} (the difference is of order 2^{224}). Therefore, for $r \in_R \{0,1\}^{240}$, where the notation "$\in_R$" means that the element is chosen uniformly at random from the corresponding set, the average expected value of $\phi(r)$ approximately equals $2^{256-240} = 2^{16}$. On the contrary, for a block r generated by the DEC PRG the expected value of $\phi(r)$ is higher than 2^{16}. Figures 1 and 2 provide an experimental evidence for this fact. Figure 1 gives the number of output blocks r with a certain value of $\phi(r)$ (the total number of generated blocks is 1320000). Intuitively, the outcome distribution should fit the normal distribution. The least-squares method shows that the closest normal distribution has parameters $\mu = 65537.0$ (rather than 65536) and $\sigma = 255.6$. Figure 2 shows that the two distributions are very close indeed.

The latter observation suggests a simple attack on the pseudorandom generator. Take an output block r and calculate $\phi(r)$. The calculation takes time proportional to 2^{16}. If $\phi(r) > 2^{16}$ conclude that the sequence is produced by the DEC PRG. Otherwise, conclude that the sequence is random with uniform distribution. Due to the above argument, for a block r output by the DEC PRG

$$\Pr[\phi(r) > 2^{16}] = 1 - \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{2^{16}} \exp[(z-\mu)^2/(2\sigma^2)]dz \approx 0.50156.$$

Therefore, our attack guesses correctly with probability about 0.50078.

The success probability of the attack can be improved if one takes into account more than one output block, say k blocks, $k > 1$, and calculates the average value of $\phi(r)$. In our experiments, we used $k = 4000$. Note that the sum of k random variables that have normal distribution with mean μ and variance σ^2 has normal distribution with mean $k\mu$ and variance $k\sigma^2$. Then,

$$\Pr[\phi(r_1) + \cdots + \phi(r_k) > 2^{16}k] = 1 - \frac{1}{\sigma\sqrt{2k\pi}} \int_{-\infty}^{2^{16}k} \exp[(z-k\mu)^2/(2k\sigma^2)]dz \approx 0.59757,$$

so the success probability of the improved attack is 0.548785. The running time is proportional to $4000 \cdot 2^{16} \approx 2^{28}$.

The simulation was implemented in C++ programming language using the NTL library [5] both for Windows and Linux platforms. In total 330 files of pseudorandom data were generated by the DEC PRG. Each file contained 4000 240-bit blocks. The seed for the DEC PRG was obtained using the `RtlGenRandom()` generator of the Platform SDK that is claimed to be cryptographically secure[1]. The analysis of 1 file took about 2 hours and 30 minutes on a 3GHz Linux machine with 1Gb of memory.

[1]This pseudorandom generator is built according to FIPS 186-2 Appendix 3.1 with SHA-1 as the iterated function [1]. It gets the seed from the current system information (current process ID, current thread, current time, etc.)

An independent work is done by Gjøsteen [4] who shows that there exists an algorithm that predicts the next bit of the DEC PRG with advantage 0.0011. The work by Gjøsteen is based on similar ideas to those proposed in this paper.

4 Conclusion

The following lines open Section 10.3 of [2].

> A DRBG[2] can be designed to take advantage of number theoretic problems (e.g., the discrete logarithm problem). If done correctly, such a generators properties of randomness and/or unpredictability will be assured by the difficulty of finding a solution to that problem. This section specifies a DRBG based on the elliptic curve discrete logarithm problem.

Our result shows that the Dual Elliptic Curve pseudorandom generator is not done correctly. The authors of [2] only claim the generator to be secure, no security proof is given. We present an efficient algorithm that distinguishes the output of the generator from the sequence of uniformly distributed random bits, which demonstrates that the generator is in fact insecure.

The main conclusion of this paper is that when designing a provably secure cryptographic scheme (e.g, a pseudorandom generator) one has to pay attention to the security proof (the reduction). An informal argument like the one in Section 10.3.1 of [2] is certainly not good enough. The scheme with a certain choice of parameters can be claimed to be provably secure only if it is shown that for these parameters breaking the scheme is as hard as solving a difficult problem faster than the fastest algorithm known so far.

References

[1] FIPS 186-2, *Digital signature standard*, Federal Information Processing Standards Publication 186-2, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, 2000.

[2] E. Barker and J. Kelsey, *Recommendation for random number generation using deterministic random bit generators*, December 2005, NIST Special Publication (SP) 800-90.

[3] D. Brown, *Conjectured security of the ANSI-NIST Elliptic Curve RNG*, Cryptology ePrint Archive, Report 2006/117, 2006, `http://eprint.iacr.org/`.

[4] Kristian Gjøsteen, *Comments on Dual-EC-DRBG/NIST SP 800-90, Draft December 2005*, March 2006, `http://www.math.ntnu.no/~kristiag/drafts/dual-ec-drbg-comments.pdf`.

[5] V. Shoup, *NTL: A library for doing number theory*, `http://www.shoup.net/ntl/`.

[6] A. C. Yao, *Theory and application of trapdoor functions*, IEEE Symposium on Foundations of Computer Science, 1982, pp. 80–91.

[2] DRBG stands for "deterministic random bit generator".

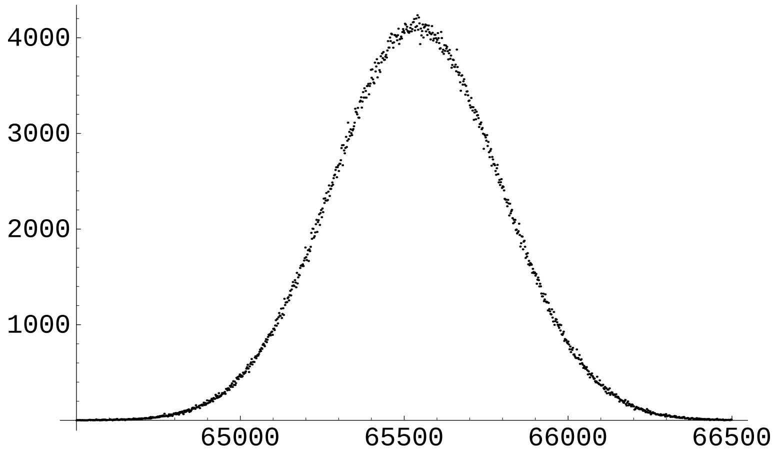

Figure 1: Number of blocks that correspond to a certain number of points on the elliptic curve (in total 1320000 blocks were generated)

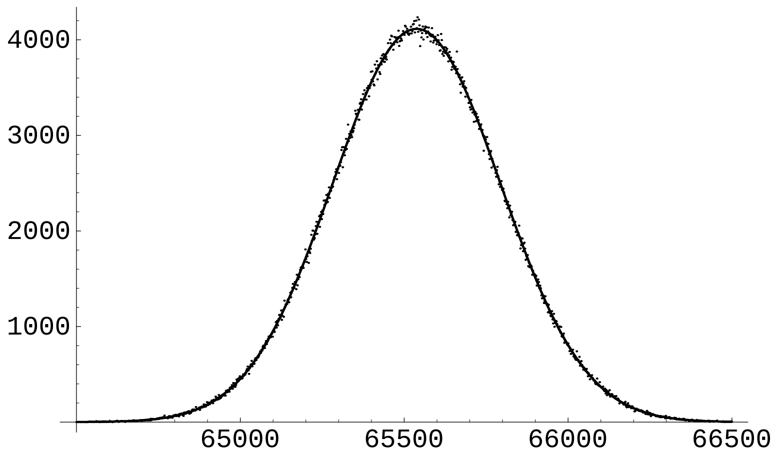

Figure 2: The outcome distribution fits the normal distribution with parameters $\mu = 65537.0$ and $\sigma = 255.6$