Comments on Dual-EC-DRBG/NIST SP 800-90, Draft December 2005

Kristian Gjøsteen*

March 16, 2006

Abstract

We analyse the Dual-EC deterministic pseudo-random bit generator (DRBG) proposed in draft of NIST SP 800-90 published December 2005. The generator consists of two parts, one that generates a sequence of points and one that extracts a bit string from the point sequence. We show that the first part is essentially cryptographically sound, while the second is not.

We give heuristic arguments that an efficient distinguisher exists for the bit sequences that are extracted from random point sequences, and construct a simple bit-predictor with advantage 0.0011. We also give experimental evidence validating the heuristics.

1 Introduction

Logically, the Dual-EC-DBRG can be divided into two parts: one that generates a point sequence and one that extracts a bit string from the point sequence. We analyse each part separately.

1.1 Notation

We denote by $x \bmod n$ the remainder of x when divided by n.

Let X be a distribution on some set. We denote the act of sampling x from the set according to the distribution X by $x \xleftarrow{r} X$.

Let X and Y be two distributions. An algorithm A has *distinguishing advantage* δ if

$$\delta = |\Pr[A(x) = 1 \mid x \xleftarrow{r} X] - \Pr[A(x) = 1 \mid x \xleftarrow{r} Y]|$$

The *statistical distance* between the two distributions is

$$\Delta(X, Y) = \frac{1}{2} \sum_s |\Pr[x = s \mid x \xleftarrow{r} X] - \Pr[x = s \mid x \xleftarrow{r} Y]|.$$

*E-mail: kristian.gjosteen@math.ntnu.no

Two distributions are ϵ-*close* if $\Delta(X, Y) \leq \epsilon$. Note that for any adversary, the distinguishing advantage is less than or equal to the statistical distance.

2 Generating point sequences

Note that we ignore reseeding and insertion of additional entropy in generating the point sequences.

Let p be a prime, $\mathbb{F}_p$ be the field with p elements and E be an elliptic curve defined over $\mathbb{F}_p$. Let $n = \#E(\mathbb{F}_p)$ be prime. (This is for convenient notation, we could just as well work in a prime-ordered subgroup.) For any points P and Q such that $P = sQ$, $0 \leq s < n$, define $\log_P Q = s$

Definition 1. Let X_{DDH} be the uniform distribution on $E(\mathbb{F}_p)^4$, while Y_{DDH} is the uniform distribution on the subset of $E(\mathbb{F}_p)^4$ where the tuples (Q, P, R, S) satisfy $\log_Q R = \log_P S$. The *elliptic curve decision Diffie-Hellman (ECDDH)* problem is distinguish the distributions X and Y.

Let $\phi : E(\mathbb{F}_p) \to \{0, 1, \ldots, n-1\}$ be a function such that when R is sampled from the uniform distribution on $E(\mathbb{F}_p)$, $\phi(R)$ is ϵ-close to uniformly distributed on $\{0, 1, \ldots, n-1\}$. (Note that the ϕ defined in the document does not have this property for any usefully small ϵ. It may still be ok, though.)

Let P, Q be non-zero points in $E(\mathbb{F}_p)$. The Dual-EC-DRBG uses the equations

$$s_{i+1} = \phi(s_i P), \tag{1}$$

$$R_i = s_i Q. \tag{2}$$

to expand a seed $s_0 \in \{0, 1, \ldots, n-1\}$ into a sequence of internal states $(s_0, s_1, \ldots, s_k)$ and an output sequence $(R_0, R_1, \ldots, R_k)$.

2.1 Point sequence is pseudo-random

Let X_i be the distribution on $\{0, 1, \ldots, n-1\}^{k+1}$ where the first $i+1$ places are iid. according to the uniform distribution, and the remaining $k-i$ places are determined by (1). X_0 is the distribution followed by the internal states of Dual-EC-DRBG.

Let Y_i be the distribution on $E(\mathbb{F}_p)^{k+1}$ induced by the distribution X_i and the map (2) applied to each place. Y_0 is the actual output distribution of Dual-EC-DRBG, while Y_k consists of $k+1$ random points from $E(\mathbb{F}_p)$.

We shall now prove that under the assumption that ECDDH is a hard problem, distinguishing Dual-EC-DRBG's output (Y_0) from random points (Y_k) is also hard.

Claim 1. *Let A be an algorithm that can distinguish Y_i from Y_{i+1} with advantage δ. Then there exists an algorithm A' that can solve the ECDDH problem with advantage at least $\delta - \epsilon$. The run-time of A' is essentially that of A.*

Proof. Let $(Q, P, R, S) \in E(\mathbb{F}_p)^4$. The algorithm A' proceeds as follows: It samples $R_0, \ldots, R_{i-1}$ independently and uniformly at random from $E(\mathbb{F}_p)$. It sets $R_i = R$ and $s_{i+1} = \phi(S)$. Then it computes $s_{j+1} = \phi(s_j P)$ for $i+1 \leq j < k$ and $R_j = s_j Q$ for $i+1 \leq j \leq k$. It runs A with p, E, n, P, Q and $(R_0, R_1, \ldots, R_k)$ as input. If A claims that its input is from Y_i, A' claims that its input is distributed according to Y_{DDH}. If A claims that its input is from Y_{i+1}, A' claims that its input is uniformly distributed in $E(\mathbb{F}_p)^4$, that is according to X_{DDH}.

Note that if there exists s such that $R = sQ$ and $S = sP$, then $R_i = sQ$ and $s_{i+1} = \phi(sP)$, so $(R_0, R_1, \ldots, R_k)$ is distributed according to Y_i.

If S is independent of Q, P and R, then by the properties of ϕ, s_{i+1} will be ϵ-close to uniform and independent of $R_0, R_1, \ldots, R_i$. Therefore R_{i+1} will also be independent of $R_0, \ldots, R_i$, and distributed ϵ-close to uniform. This means that $(R_0, \ldots, R_k)$ is distributed ϵ-close to Y_{i+1}. Call this distribution $\tilde{Y}_{i+1}$.

Let δ' be the distinguishing advantage of A'. We get

$$\begin{aligned}
\delta &= |\Pr[A(y) = 1 \mid y \xleftarrow{r} Y_{i+1}] - \Pr[A(y) = 1 \mid y \xleftarrow{r} Y_i]| \\
&= |\Pr[A(y) = 1 \mid y \xleftarrow{r} Y_{i+1}] - \Pr[A(y) = 1 \mid y \xleftarrow{r} \tilde{Y}_{i+1}] + \\
&\quad \Pr[A(y) = 1 \mid y \xleftarrow{r} \tilde{Y}_{i+1}] - \Pr[A(y) = 1 \mid y \xleftarrow{r} Y_i]| \\
&\leq |\Pr[A(y) = 1 \mid y \xleftarrow{r} Y_{i+1}] - \Pr[A(y) = 1 \mid y \xleftarrow{r} \tilde{Y}_{i+1}]| + \\
&\quad |\Pr[A'(x) = 1 \mid x \xleftarrow{r} X_{DDH}] - \Pr[A'(x) = 1 \mid x \xleftarrow{r} Y_{DDH}]| \\
&\leq \epsilon + \delta',
\end{aligned}$$

in other words, $\delta' \geq \delta - \epsilon$. □

Claim 2. *Let A be an algorithm that can distinguish the point sequence of length k generated by (1) and (2) from a random point sequence with advantage δ. Then there exists an algorithm A' that solves the ECDDH problem with advantage at least $\delta/k - \epsilon$.*

Proof. By a standard hybrid argument, A will be able to distinguish Y_i from Y_{i+1} with advantage at least δ/k, for some i. From the previous claim, this gives us an ECDDH distinguisher with advantage at least $\delta/k - \epsilon$. □

3 Bit strings from point sequences

Note that Dual-EC-DRBG does not output the point sequence generated, but instead generates a bit sequence from the point sequence. We have already shown that the point sequence will look random, so we can restrict ourselves to consider the following question: Given a sequence $(R_0, R_1, \ldots, R_k)$ of uniformly distributed, independently sampled points, how do we extract a bit string that consists of uniformly distributed, independently sampled bits?

Let $\xi : E(\mathbb{F}_p) \to \{0, 1, \ldots, p-1\}$ be the function giving the x-coordinate of a point. Let p be a τ-bit prime ($2^{\tau-1} < p < 2^\tau$), and let $0 \leq t < \tau$. Define the function $f : E(\mathbb{F}_p) \to \{0,1\}^{\tau-t}$ given by $f(P) = \xi(P) \bmod 2^{\tau-t}$.

The proposed way to extract a bit sequence from the point sequence is by extracting the $\tau - t$ least significant bits from the x-coordinate of the points and concatenating:

$$f(R_0)||f(R_1)||\ldots||f(R_k).$$

For this to be acceptable, we must have that if R is sampled uniformly at random from $E(\mathbb{F}_p)$, then $f(R)$ is uniformly distributed in $\{0,1\}^{\tau-t}$.

3.1 The distinguisher

The first observation we make is that if $t = 0$, then it is very easy to distinguish the bit string from a random bit string, simply by dividing the bit string into strings of length τ and checking if they all represent the x-coordinate of some point on the curve. A random l-bit string will pass that test with probability roughly $2^{-l/\tau}$, so this is an effective distinguisher. What happens when $t > 0$?

We shall in this section argue heuristically and use several approximations. First we note that $\#E(\mathbb{F}_p) \approx p$ (the difference is on the order of $\sqrt{p}$). Second, we asumme that $p \approx 2^\tau$. (For all the NIST prime except the 256-bit prime, the difference is negligible.)

Let $N : \{0,1\}^{\tau-t} \to \mathbb{Z}$ be the function that counts half the number of preimages of bit strings under f, that is, $N(x) = \#f^{-1}(x)/2$. Note that $\#f^{-1}(x)$ will always be a multiple of 2: The curve used has no rational points of order two, therefore every x-coordinate results in zero or two points on the curve. We also note that for a random bit string x, the expected value of $N(x)$ is $p/2^{\tau-t+1} \approx 2^{t-1}$.

In general, the probability that a random number in $\{0, 1, \ldots, p-1\}$ is the x-coordinate of some point on the elliptic curve E is very close to $1/2$. We can consider this a Bernoulli trial. Even if two x-coordinates are related by some simple linear expression, it is reasonable to assume that there is little correlation between the outcomes of the Bernoulli trials. Therefore, we can assume that for a random bit string x, $N(x)$ will follow a binomial distribution with parameters $1/2$ and 2^t.

Because the binomial distribution is symmetric around the expected value, we expect the number of bit strings with $N(x) = 2^{t-1} + r$ to be equal to the number of bit strings with $N(x) = 2^{t-1} - r$ for any r.

For each non-zero point $R \in E(\mathbb{F}_p)$, we can associate the number $N(f(R))$. When R is sampled uniformly at random, what is the distribution of $N(f(R))$? From the binomial distribution

$$\tilde{g}(i; 2^t, 1/2) = \binom{2^t}{i} \frac{1}{2^{2^t}}, \quad 0 \le i \le 2^t,$$

we get the approximate probability distribution

$$g(i) = \frac{i\,\tilde{g}(i; 2^t, 1/2)}{\sum_{j=0}^{2^t} j\,\tilde{g}(j; 2^t, 1/2)}$$

Table 1: Computed variance for the distribution $g(i)$ for various t.

t	6	10	13	16
σ^2	15.75	255.75	2047.75	16383.75

for the value of $N(f(R))$, where R is a random point. Note that $\sum_{j=0}^{2^t} j\,\tilde{g}(j; 2^t, 1/2)$ is the expected value of the binomial distribution, that is 2^{t-1}, so

$$g(i) = \frac{i\,\tilde{g}(i; 2^t, 1/2)}{2^{t-1}}.$$

We compute the expected value, which is

$$\sum_{i=0}^{2^t} ig(i) = \frac{1}{2^{t-1}} \sum_{i=0}^{2^t} i^2 \tilde{g}(i; 2^t, 1/2).$$

Note that the sum is the second moment of the binomial distribution around the origin, so the expected value of $N(f(R))$ when R is a random point, is

$$\frac{1}{2^{t-1}} 2^t \frac{1}{2}(1 + (2^t - 1)/2) = 2^{t-1} + 1/2.$$

We can also compute the second moment of the distribution to get the variance. The results for various t are given in Table 1.

This gives us a distinguisher for bit strings derived from random point sequences versus random bit strings: We split the string into blocks $x_1, \ldots, x_r$ of length $2^{\tau - t}$ and compute $N(x_i)$. If it comes from a random point sequence, we expect the average value to be larger than 2^t.

More concretely, we can design a simple bit-predictor. Given $\tau - t - 1$ bits of output, we can derive two $\tau - t$ bit strings x_0 and x_1 by setting the missing bit to 0 or 1. Then we compute $N(x_0)$ and $N(x_1)$ and guess b such that $N(x_b) > N(x_{1-b})$ if they are different.

The basic assumption is that for the correct guess b', $N(x_{b'})$ will be distributed as if $x_{b'}$ comes from a random point, while $N(x_{1-b'})$ will be distributed as if $x_{1-b'}$ was a random bit string.

The success probability μ of this bit predictor is then

$$\mu = \sum_{i=0}^{2^t} \tilde{g}(i)(g(i)/2 + \sum_{j=i+1}^{2^t} g(j)).$$

The predicted result for various t are given in Table 2.

Table 2: Bit predictor success probabilities for various t.

t	6	10	13	16
μ	0.5352	0.5088	0.5031	0.5011

Table 3: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(x)$ and curve p-196.

t	n	$\bar{\mu}$	$\bar{\sigma}^2$	$\lvert\bar{\mu}-\mu\rvert/(\sigma/\sqrt{n})$
6	100000	32.00558	16.00280889	0.441
10	110000	511.9450455	255.3650379	1.14
13	120000	4095.794708	2030.651852	1.57
16	146512	32767.90982	16366.00428	0.270

Table 4: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(f(R))$ and curve p-196.

t	n	$\bar{\mu}$	$\bar{\sigma}^2$	$\lvert\bar{\mu}-\mu\rvert/(\sigma/\sqrt{n})$
6	100000	32.50482	15.79931476	0.384
10	110000	512.5071545	255.3475611	0.148
13	120000	4096.252675	2042.637786	1.89
16	146386	32768.89841	16474.97224	1.19

3.2 Experimental results

The randomness source used for the experiments was NTL's random number generator, seeded with the output of FreeBSD's `/dev/urandom`. Most of the data comes from experiments with the two NIST curves p-196 and p-256.

We would like to test the predictions we made above by sampling from the various distributions and comparing the sample mean with the expected mean. Remember that the average of n iid. random variables with variance σ^2 has variance σ^2/n.

First, we sampled $N(x)$ for uniformly random bit strings x, $N(f(R))$ for uniformly random points R, both for various t and curves. The results are given in Tables 3–8.

It seems reasonable to use a normal approximation for the average estimator for the mean, which has variance σ^2/n. The probability is roughly 99% that a sample is within three standard deviations of the mean, and we see that this holds. For $t = 6$ and $t = 10$, we see from Tables 3–6 that the experimental results provide very strong evidence for the claim that the expected value of $N(f(R))$ is larger than $N(x)$. For all t, the sample means and variances are consistent with the predictions.

Tables 7 and 8 suggests that these results hold equally well for all curves, so even though p-196 should not be used with the generator, experiments with curve can still be used to validate the heuristics in the previous section.

Note that Tables 5 and 6 suggest that $N(x)$ is *larger* than $N(f(R))$, but this is statistically insignificant.

Figure 1 shows the results for the bit-predictor. For low t, the results clearly show that the bit-predictor has a success rate significantly different from one half, while for larger t, the results are not statistically significant, though they are consistent with the predictions.

Table 5: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(x)$ and curve p-256.

t	n	$\bar{\mu}$	$\bar{\sigma}^2$	$\lvert\bar{\mu}-\mu\rvert/(\sigma/\sqrt{n})$
6	30000	31.98556667	16.03589288	0.625
10	40000	511.928375	257.3605789	0.895
13	40000	4095.825525	2055.367818	0.771
16	50000	32768.70112	16349.64758	1.22

Table 6: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(f(R))$ and curve p-256.

t	n	$\bar{\mu}$	$\bar{\sigma}^2$	$\lvert\bar{\mu}-\mu\rvert/(\sigma/\sqrt{n})$
6	30000	32.49006667	15.5692203	0.434
10	40000	512.625475	255.280188	1.57
13	40000	4096.8141	2061.115169	1.39
16	50000	32768.16766	16218.09703	0.581

Table 7: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(x)$ and $t = 6$.

Curve	n	$\bar{\mu}$	$\bar{\sigma}^2$
p-224	30000	32.0215	16.02857204
p-256	30000	31.98556667	16.03589288
p-384	30000	31.98953333	16.04369191
p-521	30000	32.01976667	16.01510978

Table 8: Sample mean $\bar{\mu}$ and sample variance $\bar{\sigma}^2$ for $N(f(R))$ and $t = 6$.

Curve	n	$\bar{\mu}$	$\bar{\sigma}^2$
p-224	30000	32.521	15.59854562
p-256	30000	32.49006667	15.5692203
p-384	30000	32.52756667	15.54489158
p-521	30000	32.5018	15.64978509

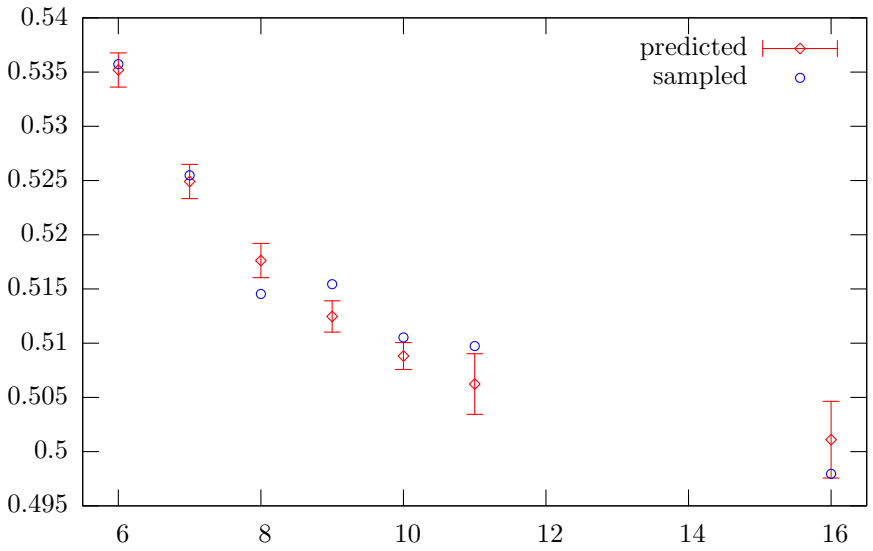

Figure 1: Predicted success rates, standard deviation for sample average, and sample average for bit predictor.

4 Conclusions

We have shown that while the point sequence generation is cryptographically sound, the way a bit string is derived from the point sequence is flawed. We have given heuristic arguments that there exists a distinguisher for the output bit strings and calculated its effectiveness. Our heuristics are validated by experimental evidence.

While the practical impact of these results are modest, it is hard to see how these flaws would be acceptable in a pseudo-random bit generator based on symmetric cryptographic primitives. They should not be accepted in a generator based on number-theoretic assumptions.