Cryptographic Engineering

working draft

Matthew D. Green
Johns Hopkins University

Excerpt: Chapter 14, Random number generation

§14.3.1 Dual EC DRBG (book pages 25-30)

Posted for 601.445/601.645 Practical Cryptographic Systems, Fall 2026
Lecture 24, Wed Dec 2: cryptographic backdoors (Dual EC and Juniper)

`https://practicalcrypto.org`

This is an excerpt from an unfinished book draft: expect gaps, rough edges, and errors, and note that section numbers and page numbers may change in later drafts. Cross-references to other chapters point to material that is not included here. Copyright © 2026 Matthew D. Green. All rights reserved. Posted for the students of the course above; **please do not redistribute** — link to the course site instead. Built 2026-11-15.

mode does not inadvertently become active during production usage, since this can produce a major security incident (see §14.3.2 for an example of a manufacturer who used a test key in production.) A second challenge is that it is hard to build a testing pathway that guarantees correct behavior in normal operation. I am aware of at least one FIPS-certified module where the RNG failed to operate during normal usage, but still passed all FIPS KAT tests because the test harness bypassed a bug in the initialization logic [209].

Self tests. Many security standards also mandate that PRGs pass runtime "self tests", such as ensuring that the generator does not produce sequential repeated outputs. While this form of runtime testing makes sense for hardware RNGs, it probably offers much less utility when applied to PRG outputs. While I have no doubt that there exists bugs that can cause these tests to fail, the cryptographic nature of PRG algorithms ensures that they will often produce statistically "plausible" output even when they are unseeded or seriously malfunctioning.

14.3 RNG nightmares

The failure of an RNG can lead to catastrophe for all cryptographic software that uses it. In this section we give several examples of accidental and *engineered* RNG failures. While these specific vulnerabilities have since been mitigated, these failure modes can be instructive when thinking about contemporary RNG designs.

DRAFT

14.3.1 Dual EC DRBG

Beginning in the late 1990s, NIST and the National Security Agency began collaborating on a new set of pseudorandom generators, called *deterministic random bit generators* (DRBGs.) This work eventually resulted in ISO/IEC 18031 [167] and NIST's Special Publication 800-90A, initially published in 2006 [32]. Both of the standards incorporated a new, NSA-designed algorithm known as the *Dual Elliptic Curve DRBG*, or Dual_EC_DRBG for short. The unusual aspect of this generator was its use of mathematics (elliptic curve operations) that are traditionally associated with public-key cryptosystems. This aspect of the generator is widely believed to have enabled a sophisticated cryptographic backdoor.

Overview of Dual EC. The generation process in Dual EC follows the basic framework shown in Figure 14.8. The standard specifies an elliptic curve subgroup of prime order q, as well as two hard-coded generator points P, Q that are explicitly specified in the standard. To generate bits, an initial seed s_0 is recursively processed by a function F to produce new internal states $s_1, s_2, s_3, \ldots$, while a second (distinct) function G produces pseudorandom output bits from each internal state. The internal state is represented as an integer used in scalar multiplication. After each application of F, it is obtained from an x-coordinate: for a

curve over $\mathbb{F}_p$, this coordinate is represented by an integer in $[0, p)$. The implementation of both functions uses elliptic curve scalar multiplication as follows:

1. The internal state update function $F : s_{i-1} \rightarrow s_i$ is implemented by computing $S \leftarrow s_{i-1}P$, and setting s_i to be the x-coordinate of S.

2. The output generation function $G : s_i \rightarrow r_i$ takes an internal state s_i and computes $R = s_iQ$. It then truncates the most significant bits off of the x-coordinate of R, and outputs the remaining bits as r_i.[22]

The security argument behind the Dual EC design was outlined in an analysis by Brown and Gjøsteen in 2007 [79]: provided that s is a random scalar and P, Q are random generators of the curve subgroup, then the resulting points (sP, sQ) should themselves indistinguishable from two randomly-sampled points in the subgroup.[23] In this argument, both the output of the generator and the next internal state are pseudorandom. Of course, a limitation of this analysis is that while indistinguishability holds for the output values viewed as *points* in the subgroup, but does not necessarily hold for the *bits* that are output by the generator. The problem is that there are many values for x in NIST's Weierstrass curves that will not satisfy the curve equation $y^2 = x^3 + ax + b \ mod \ p$, which has the effect of producing a measurable *bias* if the full x-coordinate is output at each cycle. Dual EC attempts to compensate for this issue by discarding a small number of the most significant bits of each x-coordinate before producing the remainder as output, although even this approach leaves a small but detectable bias in the generated bits.[24]

A backdoor in Dual EC? In 2007, the Microsoft cryptographers Dan Shumow and Neils Ferguson stunned the research community by presenting a short talk at a major cryptography research conference entitled "On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng" [291]. The talk made the remarkable case that the *designers* of Dual EC could potentially hold a master "trapdoor' that would allow them to compromise the security of any Dual EC implementation.

Shumow and Ferguson's concern was the provenance of the specific elliptic curve points P and Q that are specified in the NIST standard. They observed that, while security might hold if these points were generated at random, the NIST standard did not actually contain any evidence that they were. Without this assurance, it was possible that the points could be generated so that the NSA, as an attacker, would know a *discrete logarithm relationship*

[22] Using the standard's maximum output lengths, it retains the least significant 240, 368, or 504 bits of this coordinate for P-256, P-384, or P-521 respectively, discarding the most significant 16, 16, or 17 bits.

[23] This argument relies on at least two critical assumptions. The first is that the elliptic curve decisional Diffie-Hellman problem (§4.4.3) is hard in the given elliptic curve subgroups. The second, more problematic assumption, is that no party knows the discrete log relationship between P and Q.

[24] This bias was acknowledged by the generator's designers, who conducted a detailed security analysis and ultimately argued that it would not cause a problem for protocol implementations.

DRAFT

between P and Q.[25] They argued that this secret knowledge would form a master "trapdoor" for the generator. An attacker with knowledge of the trapdoor could use it to recover the internal state of the generator given only a single short output, which would allow the attacker to *predict all future outputs of the generator*, at least until it was reseeded.

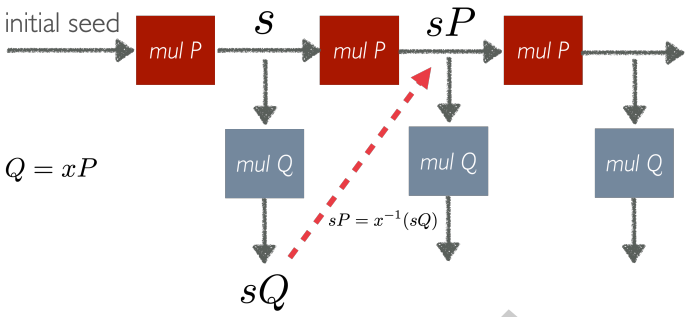


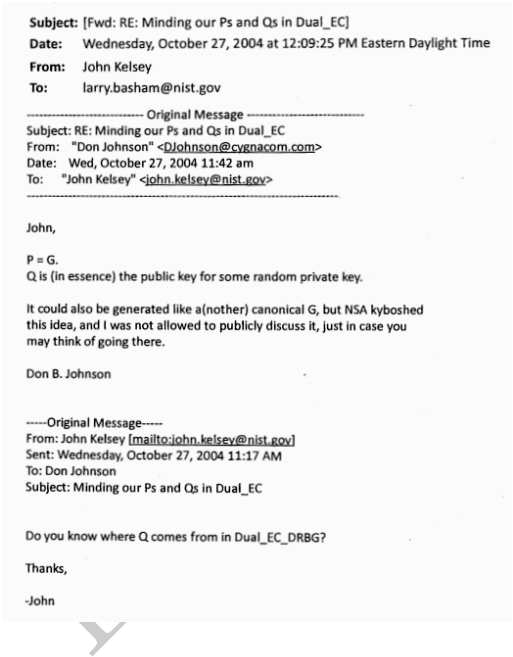
Figure 14.12: Illustration of the Dual EC generation process (grey arrows) with the backdoor illustrated (red arrow.) Knowledge of the discrete log relationship x such that $Q = xP$ allows an attacker to transform the output point sQ into the next internal state sP. This diagram shows only the point multiplications and omits the truncation of the output values, but this can be inverted by brute-force guessing the truncated bits.

DRAFT

The impact of such a vulnerability would be devastating to any cryptographic system built using Dual EC. For example, it could potentially allow a passive eavesdropper to decrypt every TLS and IPSec connection made or received by a device, thus rendering all communications transparent. This is because most implementations of these protocols will use the same generator to compute public nonces, such as the "Server Random" value used in TLS connections, prior to producing secret key material such as Diffie-Hellman ephemeral secrets. A passive eavesdropper with the trapdoor could simply observe a nonce from a TLS connection, then use the trapdoor to efficiently recover the secret keys and then decrypt the session (see [84] for a detailed impact analysis on several TLS libraries.)

The challenge for Shumow and Ferguson was that there was no way to definitively *prove* that the NSA had kept such a backdoor. Of course, any thoughtful security expert would view this standard as absurd — the mere suspicion of a master backdoor should have been sufficient to immediately revise the standard. Unfortunately, NIST leadership stonewalled critics, claiming that it had "no evidence that anyone has, or will ever have, the 'secret numbers' for the back door that were hypothesized." To the great shame of everyone involved,

[25] This implies that the attacker knows the (elliptic curve) discrete logarithm of Q base P, for example. It is easy to produce a pair of values with a known relationship: an attacker can select P at random, and then sample a random scalar x, setting $Q = xP$ (here multiplication indicates scalar multiplication of elliptic curve points.) There is no way to determine from the given points whether the author retained knowledge of x.



Subject: [Fwd: RE: Minding our Ps and Qs in Dual_EC]
Date: Wednesday, October 27, 2004 at 12:09:25 PM Eastern Daylight Time
From: John Kelsey
To: larry.basham@nist.gov

---------- Original Message ----------
Subject: RE: Minding our Ps and Qs in Dual_EC
From: "Don Johnson" <DJohnson@cygnacom.com>
Date: Wed, October 27, 2004 11:42 am
To: "John Kelsey" <john.kelsey@nist.gov>

John,

P = G.
Q is (in essence) the public key for some random private key.

It could also be generated like a(nother) canonical G, but NSA kyboshed this idea, and I was not allowed to publicly discuss it, just in case you may think of going there.

Don B. Johnson

-----Original Message-----
From: John Kelsey [mailto:john.kelsey@nist.gov]
Sent: Wednesday, October 27, 2004 11:17 AM
To: Don Johnson
Subject: Minding our Ps and Qs in Dual_EC

Do you know where Q comes from in Dual_EC_DRBG?

Thanks,

-John

Figure 14.13: An email exchange between NIST's John Kelsey and Don Johnson at Cygnacom, an NSA contractor. This conversation about the provenance of P, Q occurred during the Dual EC standardization process in 2004.

NIST's refusal to act succeeded in ending the discussion. In large part this was due to the fact that the research community was unaware of Dual EC actually being widely deployed, and so many researchers assumed the warning had been sufficient to address the matter.

The Snowden leaks. In 2013, a former NSA contractor named Edward Snowden revealed a series of classified documents describing various NSA programs. One major revelation was the existence of a $250 million/year effort to subvert cryptographic standards, known as the "SIGINT Enabling Project". This project explicitly sought to incorporate vulnerabilities in commercial encryption products and public standards, with the intent of "enabling" decryption by the NSA [259]. The New York Times article that reported on the leak specifically called out a classified reference to a successful tampering effort involving the 2006 ISO standard that contained Dual EC.

Events moved quickly following this leak. In late 2013, the security community began to take stock of the impact of a possible Dual EC compromise. Shockingly, they discovered that use of Dual EC was widespread due to the fact that it was included as the default generator in the popular RSA BSAFE library, which had been used in many FIPS-certified cryptographic products during the 2000s. This realization was followed by a Reuters report alleging that RSA Security had been received government contracts for incorporating Dual EC, even prior to the algorithm's standardization in 2006 [221]. Both NIST and RSA quickly acted to deprecate Dual EC, though this did little to undo any past damage, or to repair older versions of security products that included the library.

The Juniper Dual EC discovery. In December 2015, Juniper Networks announced the discovery of two critical CVEs in their Netscreen line of firewall/VPN devices [331]. These vulnerabilities were remarkable for the fact that they were not accidental vulnerabilities, but had been added by an outside threat actor that had compromised and tampered with Juniper's codebase beginning in 2012. One of the two vulnerabilities, CVE-2015-7756, was described as potentially "allowing a knowledgeable attacker who can monitor VPN traffic to decrypt that traffic." Although the vulnerabilities were quickly patched, disassembly by the public research community quickly revealed several surprising facts about both the original Netscreen implementation and the modified one [227, 83].

First, firmware analysis revealed that the original (unmodified) version of the Netscreen codebase contained an undocumented instance of the Dual EC generator, which had been present in the code since 2008.[26] This is quite surprising, since Juniper's CMVP certification documents make no mention of Dual EC, and instead claim that the generator uses only ANSI X9.31 for deterministic bit generation. This undocumented instance of Dual EC has

[26] While the existence of Dual EC was not disclosed in FIPS documents, Juniper did quietly release a knowledge-based article in 2013, immediately following NIST's deprecation of Dual EC. At the time, the company acknowledged the existence of Dual EC in the code (and noted the use of non-standard curve points), but claimed that the output of Dual EC was post-processed by a second ANSI generator, thus removing any risk that the generator backdoor could be exploited.

DRAFT

two further unusual features: (1) it does not use the standard NIST Q point, instead using a new value Q' of unknown provenance, and (2) while all outputs from the Dual EC generator *appear* to be post-processed using ANSI X9.31 (which should make exploitation of a Dual EC backdoor infeasible) the code for the ANSI generator contains a bug (Figure 14.14) that completely disables this post-processing. The result is that in the default configuration, all Netscreen PRG outputs were produced using Dual EC. These findings have astounding implications: any attacker with knowledge of the discrete logarithm of Q' would have been able to efficiently decrypt Netscreen VPN traffic worldwide.[27]

```
1  void prng_reseed(void) {
2    blocks_generated_since_reseed = 0;
3    if (dualec_generate(prng_temporary, 32) != 32)
4      error_handler("FIPS ERROR: PRNG failure, unable to reseed\n", 11);
5    memcpy(prng_seed, prng_temporary, 8);
6    prng_output_index = 8;
7    memcpy(prng_key, &prng_temporary[prng_output_index], 24);
8    prng_output_index = 32;
9  }
10 void prng_generate(void) {
11   int time[2];
12
13   time[0] = 0;
14   time[1] = get_cycles();
15   prng_output_index = 0;
16   ++blocks_generated_since_reseed;
17   if (!one_stage_rng())
18     prng_reseed();
19   for (; prng_output_index <= 0x1F; prng_output_index += 8) {
20     // FIPS checks removed for clarity
21     x9_31_generate_block(time, prng_seed, prng_key, prng_block);
22     // FIPS checks removed for clarity
23     memcpy(&prng_temporary[prng_output_index], prng_block, 8);
24   }
25 }
```

Figure 14.14: Decompiled segment of code from the Juniper Netscreen 6.2 RNG. The loop in lines 19–24 implements the ANSI X9.31 PRG, but this loop never runs due to the fact that the loop variable prng_output_index has been set to 32 during a subroutine call at line 8. Source: [83].

These findings are so surprising on their own that the second portion of the story is only slightly remarkable. Sometime in 2012, unauthorized attackers gained access to the Netscreen ScreenOS codebase and replaced the original value Q' with a new value Q'' of their own devising. If we presume that the attackers knew the discrete logarithm of this new value, then this attack essentially "re-keyed" an existing backdoor to create one that the attackers could exploit for themselves. The vulnerability remained undiscovered within Netscreen routers until its discovery in late 2015. To this day we do not know why the attackers did this, and whether they used their capability to attacks some other target. The FBI investigation remains open as of this writing, and no further details have been released.

[27] For a detailed analysis and proof-of-concept exploit, see Checkoway *et al.* [83].