Dual EC: A Standardized Back Door

Daniel J. Bernstein[1,2], Tanja Lange[1], and Ruben Niederhagen[1]

[1] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
tanja@hyperelliptic.org, ruben@polycephaly.org

[2] Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607–7045, USA
djb@cr.yp.to

Abstract. Dual EC is an algorithm to compute pseudorandom numbers starting from some random input. Dual EC was standardized by NIST, ANSI, and ISO among other algorithms to generate pseudorandom numbers. For a long time this algorithm was considered suspicious – the entity designing the algorithm could have easily chosen the parameters in such a way that it can predict all outputs – and on top of that it is much slower than the alternatives and the numbers it provides are more biased, i.e., not random.
The Snowden revelations, and in particular reports on Project Bullrun and the SIGINT Enabling Project, have indicated that Dual EC was part of a systematic effort by NSA to subvert standards.
This paper traces the history of Dual EC including some suspicious changes to the standard, explains how the back door works in real-life applications, and explores the standardization and patent ecosystem in which the standardized back door stayed under the radar.

Keywords. Random-number generation, back doors, NSA, ANSI, NIST, ISO, RSA, Certicom, undead RNGs.

1 Introduction

The story of the Dual EC standard is one of the most interesting ones in modern cryptography.

This work was supported by the European Commission through the ICT program under contract INFSO-ICT-284833 (PUFFIN), by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005, and by the U.S. National Science Foundation under grants 1018836 and 1314919. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation." Permanent ID of this document: d3ueael2e7c4i2s3b7a0cek0d2o3o5r4e2d. Date: 2015.07.31.

Dual EC is a pseudorandom number generator. Soon after its publication it was criticized by experts for its poor design. It is thousands of times slower than alternatives; the numbers that it produces as output are biased, flunking the most basic requirement for a pseudorandom number generator; and, most importantly, it is mathematically guaranteed to have a skeleton key that makes the output entirely predictable to anyone in possession of the key. An honest designer would not have kept the key, but a pseudorandom number generator should not have a skeleton key in the first place.

Bruce Schneier wrote a damning article [34] about Dual EC in Wired Magazine. By the end of 2007, in the view of the public cryptographic community, Dual EC was dead and gone.

1.1. The awakening. On 5 September 2013, the New York Times [31], ProPublica [19], and The Guardian [2] reported on the "SIGINT Enabling Project". The New York Times wrote:

> Cryptographers have long suspected that the agency planted vulnerabilities in a standard adopted in 2006 by the National Institute of Standards and Technology and later by the International Organization for Standardization, which has 163 countries as members.
> Classified N.S.A. memos appear to confirm that the fatal weakness, discovered by two Microsoft cryptographers in 2007, was engineered by the agency. The N.S.A. wrote the standard and aggressively pushed it on the international group, privately calling the effort "a challenge in finesse."

The surprise for the public cryptographic community was not so much this confirmation of what had already been suspected, but rather that NSA's backdooring of Dual EC was part of an organized approach to weakening cryptographic standards. Not mentioned in the reports was the biggest surprise, namely that Dual EC was not dead at all: NIST's list of "DRBG validations" [21] showed that Dual EC was provided in dozens of commercial cryptographic software libraries. Dual EC was even the *default* pseudorandom number generator in RSA Security's BSAFE library.

How could an algorithm so thoroughly criticized in public by the experts be flourishing in fielded implementations? A partial explanation surfaced in December 2013, when Reuters [20] reported that NSA paid RSA "$10 million in a deal that set [Dual EC] as the preferred, or default, method for number generation in the BSafe software."

1.2. Contents. This article covers the history of Dual EC to the extent that it is known to the public, including some information that had not previously been brought to light. This article also explains technical aspects of how the back door works and how it can be exploited in practical applications.

Section 2 introduces the ecosystem that brings random numbers to cryptographic users. Section 3 tells the story of how Dual EC was standardized, including NSA's control over NIST and ANSI, and ANSI's control over ISO. Section 4 tells the story of how Dual EC escaped modifications that would have destroyed the back door. Section 5 explains the mathematical details of the back door,

including a March 2007 modification to the NIST standard that improved Dual EC's exploitability. Section 6 explains how to exploit the back door inside TLS. Section 7 describes "Extended Random", a TLS extension whose overt purpose lacks justification and whose covert effect is to further improve the exploitability of Dual EC. Section 8 describes Certicom's patents on Dual EC exploitation and Dual EC escrow avoidance.

We thank Jeff Larson for interesting discussions and for providing us with the public comments used in Section 3. We thank Bart Preneel for providing us with the change history for ISO used in Section 3.3. We thank another expert who chose to remain anonymous for support in the investigation and interpretation of Certicom's United States patent application.

We also relied on a repository of public documents OCRed and posted by Matt Green [11] as a result of two FOIA requests, one from Matthew Stoller and United States Representative Alan Grayson, the other from Andrew Crocker and Nate Cardozo from the Electronic Frontier Foundation. NIST subsequently posted higher-quality color copies of these documents [26], although without OCR.

Our website https://projectbullrun.org/dual-ec/ contains more detail regarding several aspects of Dual EC and its history and a collection of links to related documents.

2 Where do random numbers come from?

Random numbers are the most basic building block of cryptographic protocols. Some random numbers are secrets, used as keys that must never be guessed; security relies on these numbers not being predictable. Other random numbers are public "nonces", numbers that must be used just once by the sender and receiver and never used again.

Random-number generation normally starts with a limited amount of physical randomness harvested from unpredictable elements of the computer. This physical randomness is then cleaned from possible biases, resulting in an even smaller amount of "true randomness", which is then stretched into many random numbers using a cryptographic algorithm called a "pseudorandom number generator" (PRNG). Such an algorithm is "deterministic", meaning that anybody who knows the initial true randomness can predict all future outputs—but this true randomness is always kept secret. The most important design goal for a PRNG is that outputs should not be predictable from any *other* outputs. This implies that it should be impossible to learn anything about the internal state of the algorithm based on the outputs.

A cryptographic algorithm is simply a sequence of instructions. Dedicated users who need to protect high-value information in a world full of compromised computers occasionally follow cryptographic instructions using pencil and paper and dice, but normal users rely on their computers to run cryptographic software. This software comes from developers who have collected implementations of various cryptographic algorithms into "cryptographic software libraries",

such as the open-source OpenSSL library, RSA Security's BSAFE library, and Microsoft's SChannel library. Each of these libraries includes PRNGs, and uses the random numbers from those PRNGs to support advanced cryptographic operations such as Transport Layer Security (TLS), the security mechanism that defends HTTPS web pages against espionage and sabotage.

Where do software developers obtain the cryptographic algorithms that they decide to implement? The ultimate answer is cryptographic algorithm designers. Many designers have published cryptographic algorithms, and in particular PRNGs, allowing them to be freely used by software developers. However, there are also public evaluations showing that some of these designs are unsafe: the resulting random numbers are biased (think of loaded dice that roll 6 more often than 1), or have other detectable output patterns, or allow someone to figure out the true randomness that was used as input. Sometimes software developers quietly design their own PRNGs, but these PRNGs are usually shown to be unsafe as soon as they are exposed to public scrutiny.

Software developers can, in principle, read the entire public literature on designs and evaluations of PRNGs, and select safe PRNGs that have survived careful evaluation. However, this is time-consuming, so most software developers instead rely on standardization organizations to issue standards specifying trusted PRNGs. Noteworthy PRNG standards have been issued by the National Institute of Standards and Technology (NIST), part of the United States Department of Commerce; the American National Standards Institute (ANSI), a non-profit organization; and the International Organization for Standardization (ISO), a non-governmental organization whose members consist of ANSI and the national standards institutes of 163 other countries.

To summarize, there is a large ecosystem of people and organizations involved in designing, evaluating, standardizing, selecting, implementing, and deploying PRNGs. Available documents and news stories strongly suggest that Dual EC was part of a deliberate, coordinated, multi-pronged attack on this ecosystem: designing a PRNG that secretly contains a back door; publishing evaluations claiming that the PRNG is more secure than the alternatives; influencing standards to include the PRNG; further influencing standards to make the PRNG easier to exploit; and paying software developers to implement the PRNG, at least as an option but preferably as default.

3 Standardizing Dual EC

Dual EC is known as a NIST standard for the simple reason that NIST standards are freely available online. Dual EC was also standardized by ANSI and by ISO, and those standards are published in the sense that anyone can buy copies of the standards, but the costs are high enough to interfere with public evaluation. As NIST cryptographer John Kelsey put it in 2014 [18, page 15], "public review" for ANSI standards was "not very public".

Dual EC was publicly presented at a NIST workshop on random number generation in July 2004. NIST posted the workshop slides [23] and has kept the

slides online since then. NIST also received special permission from ANSI to post a June 2004 draft of ANSI standard X9.82 "Random Number Generation" before the workshop, but NIST took the draft down after the workshop.

Several NSA employees participated actively in the workshop, but they did not present Dual EC. Instead Dual EC was described as part of a presentation "Number Theoretic DRBGs" [16] by Don Johnson from Entrust. Dual EC was obviously a very slow PRNG, but Johnson's presentation claimed that this was justified because Dual EC provided "**increased assurance**" (boldface and underline in original) compared to other PRNGs. Dual EC appeared in full detail in the June 2004 ANSI X9.82 draft.

NIST developed its own Special Publication (SP) 800-90 in parallel with ANSI X9.82, with essentially the same text. NIST published a draft of SP 800-90 on 16 December 2005, asking for comments by 1 February 2006, six and a half weeks later. The draft specified 4 PRNGs; one of those PRNGs was Dual EC.

3.1. Ignoring biases. Kristian Gjøsteen from the Norwegian University of Sciences and Technology sent NIST a paper [10] in March 2006 objecting to Dual EC as being "flawed". Johnson's slides, after presenting schematics for Dual EC, had discussed biases in the resulting bit strings and made recommendations of how to deal with these biases; but Gjøsteen's paper showed that the bit strings were even more strongly biased.

"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," Gjøsteen wrote. "They should not be accepted in a generator based on number-theoretic assumptions."

Gjøsteen's attack was improved in a May 2006 paper [35] by Berry Schoenmakers and Andrey Sidorenko from Technische Universiteit Eindhoven. "Our experimental results and also empirical argument show that [Dual EC] is *insecure*," Schoenmakers and Sidorenko wrote.

The most obvious way to stop the attacks would have been to modify Dual EC to output far fewer bits per step, but Schoenmakers and Sidorenko emphasized that this would not make Dual EC "provably secure" and that there would still be "no reasons to use this generator". In retrospect it is easy to see that this modification would also have closed the back door in Dual EC; but at that time the existence of a back door had not yet been publicly announced.

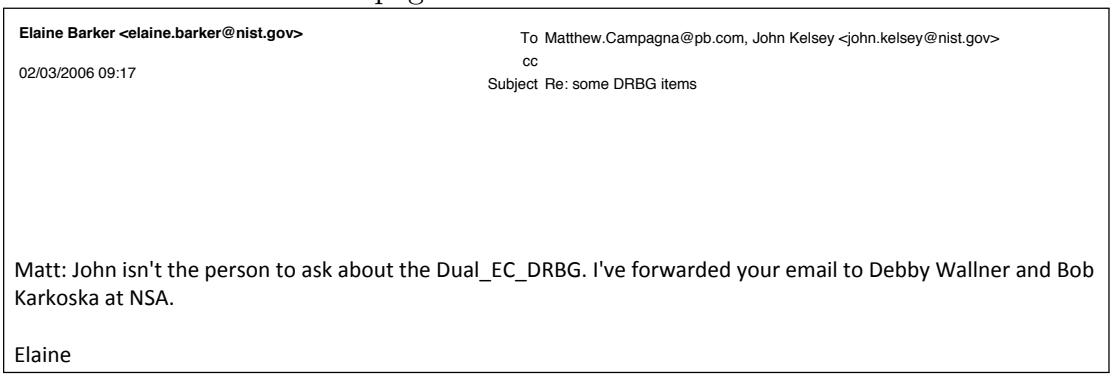
NIST's retrospective April 2014 online compilation of public 800-90 comments [25] does not include either of these papers. Obviously NIST had received Gjøsteen's paper after its 1 February 2006 deadline for comments. On the other hand, it turns out that NIST did not take this deadline seriously: NIST considered and acted upon comments that it received from Matt Campagna at Pitney Bowes on 3 February 2006, and comments that it received from Johnson on 31 March 2006, also not included in the online compilation.[3] NIST was continuing to actively edit SP 800-90 into May 2006; see [11].

[3] These extra documents were obtained by journalist Jeff Larson in January 2014. We are indebted to Larson for allowing us to present this new information here.

NIST standardized SP 800-90 in June 2006. Despite the objections from Gjøsteen, Schoenmakers, and Sidorenko, this final version of SP 800-90 included Dual EC, with its full bias.

3.2. Hiding behind NIST. None of the NIST workshop documents listed any author or designer of Dual EC. The authors listed for SP 800-90 were both from NIST: Elaine Barker and John Kelsey. As far as we know, none of the documents published at the time pointed to NSA as the source of Dual EC.

However, NIST's private response to Campagna sheds interesting light on the authorship of Dual EC.[4] Campagna had sent email to Kelsey, cc'ing Barker. Barker wrote back to Campagna as follows:



Elaine Barker <elaine.barker@nist.gov>

02/03/2006 09:17

To Matthew.Campagna@pb.com, John Kelsey <john.kelsey@nist.gov>
cc
Subject Re: some DRBG items

Matt: John isn't the person to ask about the Dual_EC_DRBG. I've forwarded your email to Debby Wallner and Bob Karkoska at NSA.

Elaine

This appears to confirm public statements from Kelsey in December 2013 [17] and May 2014 [18] referring to "designers of Dual EC DRBG at NSA" and saying that "NSA provided Dual EC DRBG".

Further confirmation that Kelsey was not the designer of Dual EC appears in one of Kelsey's internal drafts [22] of ANSI X9.82 Section 9.12, "Choosing a DRBG algorithm". The notes signed "JMK" (brackets in original) show that Kelsey did not even feel competent to comment on Dual EC's security, and that he was relying critically on advice from NSA. See [22, p. 9]:

> X.3 DRBGs Based on Hard Problems
> [[Okay, so here's the limit of my competence. Can Don or Dan or one of the NSA guys with some number theory/algebraic geometry background please look this over? Thanks! --JMK]]

See also a bit later in the document [22, p. 10]:

> X.3.1 Dual_EC_DRBG
> The DUAL_EC_DRBG relies for its security on the difficulty of the elliptic curve discrete log problem--given (P,xP), determine x. Widely used signature and key agreement schemes are based on this problem, as well. A very conservative system design which had few performance requirements on its random number generator mechanism might thus choose the DUAL_EC_DRBG as its DRBG. This would ensure that the security of the whole application or system relied very cleanly on the difficulty of this one problem.

[4] As above, we are indebted to Larson for tracking down this information.

> [[I'm really blowing smoke here. Would someone with some actual understanding of these attacks please save me from diving off a cliff right here? --JMK]]

3.3. Taking control of ISO. By summer 2003, ISO/IEC Joint Technical Committee 1 Subcommittee 27 was far into its own multi-year process of standardizing random-number generators, not including Dual EC. Its internal 2003 draft received more than 150 comments, but most of those comments were minor corrections and clarifications such as changing "Any secure hash" to "Any secure hash function". The draft was approved by 19 of the 24 countries that voted, including half of the countries that had sent in comments.

The United States comment [14] was strikingly different, the only comment objecting to the "whole document":

> The U.S. National Body has reviewed ISO/IEC 2 CD 18031, N3578. We feel that this document is lacking sufficient depth in many areas and simply is not developed enough to be an ISO standard which encompasses both Non-deterministic and Deterministic Random Bit Generation. We do feel that ANSI X9.82 Random Bit Generation standardization work is much further developed and should be used as the basis for this ISO standard.
>
> To make ISO/IEC 18031 consistent with X9.82 would require extensive commenting and revisions. To better progress this standard, the U.S. has instead developed a contribution for ISO that is consistent with ANSI X9.82, but written in ISO format. Furthermore, we believe this contribution will also be complementary to ISO/IEC 19790.
>
> We provide this contribution as an attachment, and propose that ISO further develop this contribution as their standard.
>
> Additionally, the U.S. recognizes that ANSI X9.82 is not an approved standard and still requires further work. As ANSI X9.82 develops, the U.S. will contribute these changes to ISO.

The attachment, 153 pages, was an early version of ANSI X9.82, including a full description of Dual EC (using the same points P and Q that were later standardized by NIST for the curves P-256, P-384, and P-521; see Section 4 for discussion of the importance of P and Q).

ISO did what the United States told it to do. After two years it released standard ISO 18031:2005, including Dual EC. This rather blunt takeover of the ISO standard is, presumably, what NSA internally referred to as a "challenge in finesse".

4 Minding the P's and Q's

Every summer hundreds of cryptographers gather in Santa Barbara for the annual Crypto conference. The highlight of the conference is the three-hour "rump

session" on the evening of the second day, featuring a series of short talks on very recent results.

Dan Shumow and Niels Ferguson, cryptographic researchers at Microsoft, announced at the Crypto rump session in August 2007 that there was a "possibility of a back door" in Dual EC. The name Dual EC refers to two "elliptic curve points" P and Q used inside the algorithm; what Shumow and Ferguson explained [36] was a way for whoever had generated the points P and Q to start from one random number produced by Dual EC and predict all subsequent random numbers. Recall that some PRNG outputs are made public, while others are secret; releasing just one public output would allow the attacker to predict all subsequent secret outputs, obviously a security disaster.

"Break the random-number generator, and most of the time you break the entire security system. Which is why you should worry about a new random-number standard that includes an algorithm that is slow, badly designed and just might contain a backdoor for the National Security Agency," Bruce Schneier wrote in a November 2007 article [34] for Wired Magazine. "My recommendation, if you're in need of a random-number generator, is not to use Dual_EC_DRBG under any circumstances."

However, NIST did not withdraw Dual EC from its standard. NIST sent Schneier a letter [3] saying "We have no evidence that anyone has, or will ever have, the 'secret numbers' for the back door ... For this reason, we are not withdrawing the algorithm at this time."

4.1. Behind the scenes. Kelsey had already been asking questions about P and Q as early as October 2004, as shown by the following email exchange [15] between Kelsey and Johnson, made public in 2014:

```
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
```

The "random private key" mentioned in Johnson's message is the simplest way that the Dual EC authors could have generated Q. However, from the perspective of the Shumow–Ferguson attack, this private key is exactly the secret information needed to unlock the back door in Dual EC.

The alternative mentioned in Johnson's message, generating Q "like a(nother) canonical G", would have made it much more difficult for the Dual EC authors to know this "random private key". It is hard to imagine any legitimate reasons for NSA to have told Johnson not to talk about this idea. "I didn't catch why this was significant then," Kelsey wrote in 2014.

According to Kelsey [18, page 24], Ferguson reported the Shumow–Ferguson attack to ANSI in 2005. Two other participants in the ANSI discussions, Dan Brown and Scott Vanstone from Certicom, had discovered the same attack before 21 January 2005 (see Section 8), but apparently did not report it to ANSI.

NSA's response, according to Kelsey, was that NSA had "generated (P, Q) in a secure, classified way"; that NSA wanted to allow existing devices using this P and Q "to get FIPS validated" (i.e., certified by a testing laboratory to meet NIST standards); and that it "would be reasonable to allow other users to generate their own (P, Q)".

NIST could easily have generated a new Q "like another canonical G", and recommended this Q as a replacement for NSA's Q, which as noted above would have made Dual EC exploitation much more difficult. NIST could nevertheless have allowed NSA's Q as a non-recommended option, answering NSA's request for FIPS validation. NIST could also have entirely eliminated this problematic PRNG from the standard, ignoring NSA's request.

Instead NIST added an appendix to its draft of SP 800-90 explaining how users *could* generate their own P and Q, but ("to avoid using potentially weak points") specifically recommending *against* doing this. Furthermore, another paragraph in SP 800-90 prohibits FIPS validation for any users doing this: see [27, page 84] ("One of the following NIST approved curves with associated points shall be used in applications requiring certification under FIPS 140-2") and the detailed testing instructions [24, page 19] ("CAVS Dual_EC_DRBG tests use only the NIST Approved curves and associated points").[5] The rule forcing the pre-described points for FIPS validation was already in the June 2004 draft of

[5] CAVS stands for NIST's Cryptographic Algorithm Validation System. "Cryptographic algorithm validation is a prerequisite to the Cryptographic Module Validation Program (CMVP)." See http://csrc.nist.gov/groups/STM/cavp/.

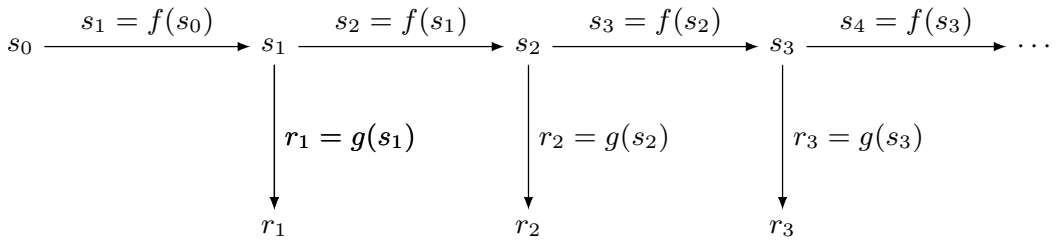


Fig. 5.1. General schematic of a state-based PRNG with functions f and g.

ANSI X9.82 and was not modified when a section on using alternative points was added.

SP 800-90 did not discuss the origin of P and Q, and did not explain the power available to whoever had generated P and Q. In hindsight it is quite amazing how blindly NIST trusted NSA.

4.2. NSA's public story. Richard George, who was Technical Director of the Information Assurance Directorate at NSA from 2003 until his retirement in 2011, made the following claims regarding Dual EC in a talk [9] at the Infiltrate conference in May 2014: "We were gonna use the Dual Elliptic Curve randomizer. And I said, if you can put this in your standard, nobody else is gonna use it, because it looks ugly, it's really slow. It makes no sense for anybody to go there. But I'll be able to use it. And so they stuck it in, and I said by the way, you know these parameters that we have here, as long as they're in there so we can use them, you can let anybody else put any parameters in that they want."

As far as we know, there have been no public comments from NSA regarding the prohibition on FIPS validation of alternative P and Q; NSA instructing Johnson not to talk about generating a new Q; NSA paying RSA Security to implement and use Dual EC; and NSA internally advertising its PRNG standardization as part of a systematic effort to weaken cryptographic standards.

5 How the Dual EC back door works

This section digs into mathematical details: how PRNGs work in general; how Dual EC works; and how the back door works.

SP 800-90 allows users to refresh the internal state of PRNGs with some additional input. This complicates the definition of Dual EC. There are actually two slightly different versions of Dual EC in two releases of SP 800-90: the June 2006 version [27], now called Dual EC 2006, and an updated March 2007 version, now called Dual EC 2007. Additional input breaks the back door in Dual EC 2006 in many cases, often preventing the attacker from predicting the Dual EC output. Fortunately for the attacker, Dual EC 2007 repairs the back door, allowing the back door to be used in all cases, even when additional input is provided.

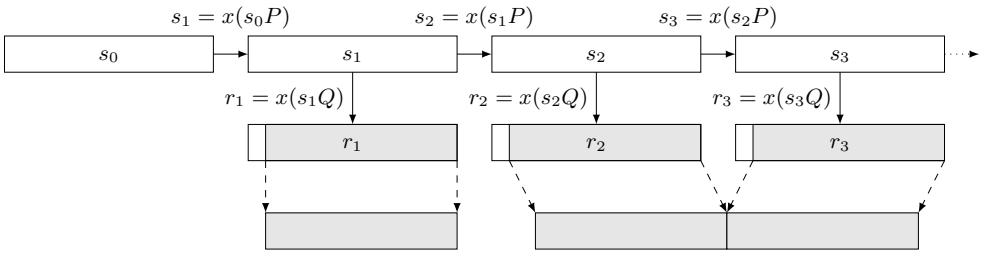


Fig. 5.2. Basic Dual EC algorithm using points P and Q on an elliptic curve.

5.1. PRNG structure: state updates and output functions. Figure 5.1 shows a general schematic of a PRNG. The PRNG maintains an internal state s_i; the initial state s_0 is initialized from an entropy source. Each time some random output is requested from the PRNG, the internal state is updated from s_{i-1} to s_i using a function f such that $s_i = f(s_{i-1})$. After the internal state is updated, the PRNG derives a certain number of random bits using another update function g by computing $r_i = g(s_i)$ and returning some bits of r_i. If more bits are requested than available from r_i, the PRNG updates the state again by computing $s_{i+1} = f(s_i)$. Then $r_{i+1} = g(s_{i+1})$ is computed, and some bits from r_{i+1} are appended to the previously generated bits. This process is repeated until the requested number of bits have been generated.

For this process to be secure, it is crucially necessary that the internal state is not learned by an attacker. An attacker who knows some internal state s_i is able to compute all following states $s_{i+1}, s_{i+2}, \ldots$ and to reproduce all output bits by computing $r_i, r_{i+1}, r_{i+2}, \ldots$. Therefore, the function g must be a one-way function; otherwise an attacker who learns some random output is able to compute the internal state of the PRNG. If the function g has a back door that allows an attacker to compute the internal state of the PRNG, then the complete PRNG is insecure.

5.2. Basic Dual EC algorithm. Dual EC follows the general PRNG scheme described above. Dual EC specifies two points P and Q on the standard NIST P-256 elliptic curve. The internal state of Dual EC is a 256-bit integer s. The function f for updating the internal state is defined as $f(s) = x(sP)$, computing the sth multiple of P and returning the x-coordinate of the resulting point. The function g for deriving some random output is defined as $g(s) = x(sQ)$.

Both functions f and g are cryptographically secure one-way functions. It is computationally hard to compute s given sP and P, i.e., to solve the elliptic-curve discrete-logarithm problem (ECDLP).

Figure 5.2 illustrates the Dual EC algorithm. Given an initial state s_0, the PRNG updates the internal state by computing $s_1 = x(s_0P)$ when some random bits are requested. Then $r_1 = x(s_0Q)$ is computed and the most significant 16 bits of r_1 are discarded. Finally up to 30 random bytes are returned. If more than 30 bytes are required, the process is performed repeatedly, each time dropping the most significant 16 bits of r. The output bits are concatenated and finally returned.

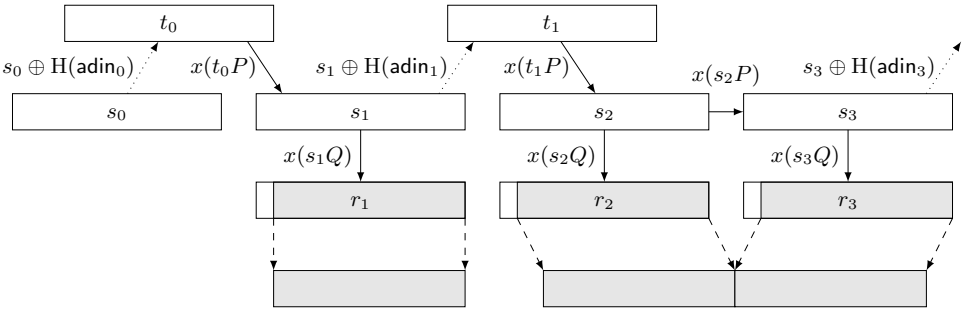


Fig. 5.3. Dual EC 2006 with additional input. Compare Figure 5.2.

Dual EC also allows larger variants using the P-384 and P-521 curves. The state sizes are 384 and 521 bits respectively. The outputs are correspondingly larger: 46 bytes (368 bits) and 63 bytes (504 bits).

5.3. Basic Dual EC back door. The Shumow–Ferguson attack works as follows. Assume that the attacker knows a scalar d such that $P = dQ$, and sees the random output r_1 (e.g., when r_1 is used as a public nonce). Now he can recompute a y-coordinate corresponding to the x-coordinate r_1 using the curve equation and obtains $R = (r_1, y_{r_1}) = s_1Q$ for some s_1 unknown to the attacker. Finally, he computes $d \cdot R = d \cdot s_1Q = s_1dQ = s_1P$ and learns the internal state s_2 as the x-coordinate $x(dR) = x(s_1P)$. Now, the attacker can reproduce all the following Dual EC output of the victim. Thus, the knowledge of the scalar factor d with $P = dQ$ provides a back door for the attacker to the internal state of Dual EC.

As described above, the most significant 16 bits of r_i are discarded. However, the attacker is able to recover the missing bits easily: the attacker requires at most 2^{16} attempts in order to find the missing bits.

5.4. Dual EC 2006. As mentioned earlier, NIST allows users to enter additional input into Dual EC. The additional input string may be, e.g., the current system time, some counter value, some high-entropy randomness, or simply 0 if no refresh is desired.

When random output is requested from Dual EC 2006, the hash of some additional input string "adin" is `xor`'ed into the state before the basic state update is performed. For example, in Figure 5.3, 30 bytes are requested when the state is s_0, 60 bytes are requested when the state is s_1, and further bytes are requested when the state is s_3, so hashes of additional inputs are `xor`'ed into s_0, s_1, and s_3.

5.5. The partially broken Dual EC back door. If additional input is used in Dual EC 2006, and the attacker sees random output of at most 30 bytes, then the back door does not work any more. Assume the attacker observes r_1. Since s_1 has been modified with some additional input string and since there is no more known relation between r_1 and s_2, the attacker can no longer apply his

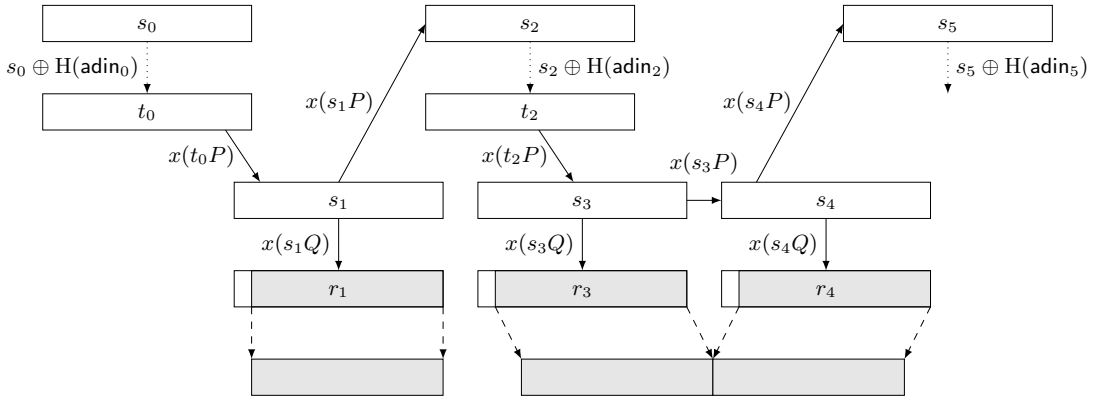


Fig. 5.4. Dual EC 2007 with additional input. Note the additional state update compared to Figure 5.3.

back-door computation. Even if the attacker can guess adin_1, he is not able to recover the internal state from r_1.

The back door still works in case the attacker observes some random output that is longer than 30 bytes. For example, if the attacker observes combined output from r_2 and r_3, he can simply compute $R_2 = (r_2, y_{r_2})$ and obtain the internal state $s_3 = x(dR_2) = x(ds_2Q) = x(s_2P)$. In order to compute the following output values, the attacker needs to correctly guess the following additional input strings.

The ability of an implementation of Dual EC to refresh the internal state using an additional input string breaks the back door for the attacker in many practical cases where the amount of randomness observed by the attacker is smaller than or equal to 30 bytes.

5.6. Dual EC 2007: the repaired back door. Dual EC 2007 demands an additional update step of the internal state at the end of each invocation. That means that after the requested number of random bits have been generated, the internal state is updated one more time by computing $s_{i+1} = x(s_iP)$. Figure 5.4 illustrates the modified algorithm.

Because of this additional state update, the attacker is able to apply his back-door computation. Given the random output r_1, the attacker computes $R_1 = (r_1, y_{r_1})$ and obtains the internal state $s_2 = x(dR_1) = x(ds_1Q) = x(s_1P)$. However, in order to compute the following random output, the attacker still has to guess any additional input strings (if the specific implementation of Dual EC used by the victim is making use of additional input).

5.7. Forward secrecy. The official reason for the change from Dual EC 2006 to Dual EC 2007 was to provide "backtracking resistance".

"Backtracking" does not mean working backwards from random outputs to earlier random outputs, which would be a serious security problem. It means working backwards from *the internal state* to earlier random numbers. For example, if f in Figure 5.1 is not one-way, the attacker can backtrack to all previous internal states and compute all previous random numbers.

Of course, PRNGs are designed to preserve the secrecy of the internal state. The idea of "backtracking resistance", also called "forward secrecy", is to reduce the damage in the extreme situation of an attacker somehow *stealing* the internal state: the attacker will be able to predict all future random numbers but will not be able to compute earlier random numbers.

The function f in Dual EC has always been one-way. The only issue with "backtracking resistance" in Dual EC 2006 with additional input is that an attacker who sees the current state s_1 can compute the *current* random number r_1. This issue disappears as soon as the next additional input is provided, allowing s_1 to be replaced with s_2; i.e., the current random number is protected against theft as soon as a new random number is generated.

The obvious way to fix theft of the current random number is to *first* output this number, *then* immediately absorb additional input, *then* apply the function f to update the state. This provides full "backtracking resistance"; it is also simpler and more efficient than Dual EC 2007. However, from the attacker's perspective, Dual EC 2007 is much more satisfactory because it fixes the back door.

6 Exploiting the back door in Dual EC implementations

> *Their eyes open wide when I talk about how how hard it is to really get the information they assume they just get to attack this thing. ... I've challenged any of them to actually generate their own parameters and show me that in real life they can recover that. No one has done it yet.* —Richard George, May 2014 [9]

The basic Dual EC attack allows the attacker to predict all subsequent Dual EC outputs, *assuming* that the attacker has seen a sufficiently long stretch of contiguous output bits. If all output bits were kept secret then the attack could not even get started. Some cryptographic protocols send random bits through the network, but it is not immediately obvious that Dual EC is exploitable inside real protocols, such as the widely deployed TLS standard, the primary encryption mechanism used for communication in the Internet today.

This section summarizes the results of a paper "On the practical exploitability of Dual EC in TLS implementations" [8] posted in April 2014 and presented at the USENIX Security Symposium in August 2014. This paper is joint work that we carried out with Stephen Checkoway, Matt Fredrikson, Adam Everspaugh, Matt Green, Tom Ristenpart, Jake Maskiewicz, and Hovav Shacham.

The paper shows that the basic Dual EC attack ignores critical limitations and variations in the amount of the PRNG output actually exposed in TLS, additional inputs to the PRNG, PRNG reseeding, alignment of PRNG outputs, and outright bugs in Dual EC implementations. The levels of Dual EC exploitability vary between RSA Security's BSAFE, Microsoft's SChannel, and OpenSSL.

However, in all analyzed situations where Dual EC was actually used in TLS, the Dual EC back door turned out to be exploitable by anyone who knows the secret d. TLS transmits enough random data in plain text during the TLS

handshake, while it uses other random data to generate secret keys; the paper showed how to recover those secret keys.

6.1. How targets vary. The paper investigated four TLS libraries offering Dual EC: OpenSSL-FIPS (a FIPS-validated version of OpenSSL), Microsoft's SChannel, and two versions of RSA Security's BSAFE, namely BSAFE-Java and BSAFE-C.

The key to the back door, i.e., the factor d such that $P = dQ$, is not publicly known, so the paper replaced (P, Q) in each library with a new (P, Q) using a known key. Replacing the points in SChannel, BSAFE-Java, and BSAFE-C required some reverse engineering. Replacing the points in OpenSSL-FIPS was relatively easy because OpenSSL is open-source.

OpenSSL-FIPS turned out to have a severe Dual EC bug, despite FIPS validation: the self-test of the library consistently failed when OpenSSL-FIPS was configured to use Dual EC. It is therefore reasonable to guess that nobody ever used Dual EC with OpenSSL. On the other hand, there is an obvious fix for the bug, producing a modified library "OpenSSL-fixed" studied in the paper; users of OpenSSL-FIPS might have silently fixed this bug without reporting it. The other libraries had functional Dual EC implementations, and Dual EC was the default PRNG for both BSAFE-Java and BSAFE-C.

OpenSSL-fixed was the only library that used an additional input string for each request for random output, thus increasing the cost of the basic attack by the need to guess the adin; the other three libraries did not use any adin. BSAFE-Java, BSAFE-C, and OpenSSL-fixed implemented Dual EC 2007, with the additional update step at the end of each invocation. It appears that SChannel tried to implement Dual EC 2007 but accidentally implemented something equivalent to Dual EC 2006 instead: SChannel computes the additional state update but discards the result and continues with the previous state. This does not hurt Dual EC exploitability since SChannel does not use adin. One other difference between the Dual EC implementations is that BSAFE-C buffered unused random bytes for consecutive invocations, reducing the computational cost of Dual EC.

Beyond the differences in the Dual EC implementations, the implementations of TLS varied in how they used random output from Dual EC. By default each library generated a different number of random values, used random values in a different order in the TLS handshake, and used a different cipher suite. The paper used each library as a TLS server, and investigated an ephemeral cipher suite.

In a typical example of a TLS handshake with an ephemeral key exchange, the server generates random numbers for the *session ID*, the *server random*, the ephemeral secret a, and (depending on the cipher suite) the *signature nonce*. Both server random and session ID are sent in plaintext over the wire and therefore can be used as entry points for the attack. If the cipher suite uses "ECDHE", an ephemeral elliptic-curve Diffie–Hellman key exchange, the ephemeral secret is used to compute aP; if the cipher suite uses "DHE", an ephemeral Diffie–Hellman key exchange (without elliptic curves), the ephemeral secret is used to

Attack	Total Worst Case Runtime (min)
BSAFE-C v1.1	0.04
BSAFE-Java v1.1	63.96
SChannel I	62.97
SChannel II	182.64
OpenSSL-fixed I	0.02
OpenSSL-fixed II	83.32
OpenSSL-fixed III	$2^k \cdot 83.32$

Fig. 6.1. Experimental timings for recovering the internal state of Dual EC from a TLS handshake for several implementations on a four-node, quad-socket AMD Opteron 6276 (Bulldozer) computing cluster.

compute g^a (for some integer g). The nonce is used to compute a signature. The value from the key exchange and the signature (if available) are sent over the wire as well and can be used to verify that the correct Dual EC internal state has been found. Finally, client and server compute their secret encryption key from the exchanged data; the attacker is able to recompute the same key once he has found the internal state and obtained the server's ephemeral secret.

6.2. Attack cost. Table 6.1 shows an overview of the worst-case runtimes for the attack. The attacker has to spend a different computational effort for each case, reflecting differences in the implementations of Dual EC and TLS.

The attack on BSAFE-C is the cheapest. For a TLS connection using DHE, the server draws in consecutive order 32 bytes for the session ID, 28 bytes for the server random, and 20 bytes for the ephemeral secret. The internal buffering of random bytes reduces the number of bits that need to be guessed to only 16, because the consecutively drawn random values "session ID" and "server random" can simply be concatenated to obtain 30 bytes of a single invocation output. Therefore, at most 2^{16} bit combinations need to be checked. A small 16-CPU research cluster was able to recover the internal state and break the connection within 0.04 minutes.

Attacking the BSAFE-Java version is more expensive. Here, the TLS implementation (using an ECDHE cipher suite) does not obtain the session ID from a call to Dual EC; the first value drawn from the PRNG is 28 bytes of the server random. This is followed by a call for 32 random bytes for the secret DH key and finally an ECDSA nonce. Thus, using the 28 bytes of the server random, 32 bits need to be guessed to recover the 32-byte internal state, resulting in 2^{32} possible combinations. The research cluster used at most 64 minutes for the attack.

SChannel requests random data for an ECDHE cipher suite in a different order. The TLS implementation first requests 32 bytes for the session ID, but does not make all of these available to the attacker: it reduces the top four bytes modulo 20000 before transmission. It then requests 40 bytes for the secret ephemeral key; 28 bytes for the server random; and, finally, 32 bytes for the secret ECDSA nonce.

For SChannel, there are two cases. "SChannel I" means that a sequence of TLS handshakes is available to the attacker, so he can use the server random from a previous TLS handshake in order to compute the internal state. In this case, the attacker needs to guess the missing 32 bits from the 28-byte server random. This requires up to 2^{32} operations. The research cluster used less than 63 minutes.

"SChannel II" means that only data from a single TLS handshake is available to the attacker in order to compute the internal state. In this computationally more complex case, the back door computation must use the session ID as entry point for computing the ephemeral key. This requires about 2^{18} guesses to recover the original value before the modulo operation; each guess, in turn, requires checking 2^{16} possibilities since the most significant 16 bits were discarded. Testing a possibility means recomputing the ECDHE public key. In total, this case requires up to 2^{34} recomputations. The research cluster used a bit more than three hours.

OpenSSL-fixed requests first 32 bytes for the session ID, then 28 bytes for the server random, and finally 32 bytes for the ephemeral key. The randomness for the session ID is obtained in two 30-byte pieces, so it is particularly easy to recompute the internal state. Only the first 16 "discarded" bits need to be guessed; the correct state can quickly be verified by comparing the corresponding 16 bytes of the second piece with the last two bytes of the session ID.

Recall, however, that OpenSSL uses an additional input string in each request for random data, to refresh the randomness of the internal state. This adin is a concatenation of the current system time in seconds, the current system time in microseconds, a monotonically increased 32-bit counter, and the process ID. The current system time in seconds is known to the attacker since it is contained in the TLS handshake. However, the remaining data of the adin needs to be guessed.

Table 6.1 shows three cases. "OpenSSL-fixed I" assumes that the entire adin is known to the attacker; thus he only needs to guess the 16 missing bits for the session ID and is able to recompute the state and follow all internal state updates. The cluster used 0.02 minutes to recompute the state on 16 CPUs in parallel.

"OpenSSL-fixed II" assumes that the attacker knows the counter (because he may be attacking the very first TLS handshake when the counter is 0) and the process ID (because it may be determined depending on the order in which systems services are started during boot time). Then the attacker only needs to recompute the current microsecond in which the adin was computed; this requires at most 1,000,000 guesses. The research cluster recomputed the internal state within 84 minutes.

Finally, if counter and/or process ID are not known, a multiple of the time for OpenSSL II is required. "OpenSSL III" in the table includes a factor 2^k, giving an idea about the required time in case k bits of unknown adin need to be guessed.

The attacks are easy to parallelize and scale well on a large number of CPUs. Thus, an attacker who can afford a large CPU cluster is able to compute the

internal state in a much shorter time than the 16-CPU research cluster used in the experiment. About 1,000 CPUs are sufficient to finish most of the attacks (all except SChannel II and OpenSSL-fixed III) within 1 minute. A computing cluster of the size of the Tianhe-2 supercomputer with 70,000 CPUs computes most of the attacks in under one second.

6.3. Attack scenarios. An attack on a TLS connection does not need to be done "online", while the communication between the client and the server is ongoing. It is possible to use the back door to retroactively break into a recorded TLS connection at any time. The attack on TLS connections works as well when the client instead of the server is targeted.

If the server is targeted, the recovered internal state can be used not only to compute the encryption keys of the targeted connection but also of all future connections to the server by any client (see the discussion of SChannel I). If a signature scheme based on the digital signature algorithm (DSA, designed by NSA) is used for server authentication, the knowledge of just one signature nonce enables the attacker to compute the server's secret identity key and thus to impersonate the server.

7 Extended Random

The basic Dual EC attack requires the attacker to see at least a block of 30 Dual EC output bytes (and, of course, to know the back door for the Dual EC parameters). As shown in the previous section, implementations of TLS often make only 28 consecutive bytes public. This increases the cost of using the back door from about 2^{15} to about 2^{31}.

A single 2^{31} computation is not a problem, but if the same attack is carried out many times, say 2^t times, then the attack costs are increased from 2^{15+t} to 2^{31+t}. This is a serious issue when t is large. Even worse, if Dual EC is used with P-384 or P-521 instead of P-256, then there is a critical gap between the standard 224 bits revealed by TLS and the 368 or 504 bits in a Dual EC output block, and the attack becomes infeasible.

There are four proposals of TLS extensions that increase the amount of PRNG output visible to an attacker: "Opaque PRF" [32] from 2006, "Extended Random" [33] from 2008, "Additional PRF Inputs" [13] from 2009, and "Additional Random" [12] from 2010. None of these extensions were standardized, but BSAFE implements Extended Random as an option, and a 2012 summary of TLS monitoring [1] reveals that occasionally, about once in every 77000 connections, clients actively requested Extended Random.

7.1. How Extended Random affects Dual EC exploitation. A client that supports "Extended Random" sends more random data in its initial "client random" in its TLS handshake: instead of 224 bits it sends a string of random bits at least "twice as long as the security level", i.e., $\geq$256 bits. If the server also supports "Extended Random" then the server responds with its own "server random" of the same length that the client chose.

Extended Random reduces the Dual EC attack cost from 2^{31} to 2^{15}, since the attacker no longer needs to guess 16 extra missing bits. Extended Random also simplifies the attack, because it includes more than one block of output: the attacker easily and efficiently confirms guesses for internal Dual EC states by comparing the potential next output to the next bits of the extended randomness. A 512-bit Extended Random also makes attacks feasible against, e.g., the P-521 variant of Dual EC.

To summarize: From the perspective of a Dual EC attacker, there are obvious benefits to Extended Random.

7.2. The official reason for Extended Random. Extended Random was proposed in an Internet-Draft by Eric Rescorla (RTFM, Inc.) and Margaret Salter (NSA) in April 2008. The latest version, draft 02, is from 2 March 2009:

```
Network Working Group                                      E. Rescorla
Internet-Draft                                              RTFM, Inc.
Intended status:  Informational                             M. Salter
Expires:  September 3, 2009                   National Security Agency
                                                          March 02, 2009


                    Extended Random Values for TLS
                draft-rescorla-tls-extended-random-02.txt
```

Section 6 of the document acknowledges a funding source: “This work was supported by the US Department of Defense.”

The Internet-Draft states the following rationale for extended randomness:

```
The United States Department of Defense has requested a TLS mode
which allows the use of longer public randomness values for use with
high security level cipher suites like those specified in Suite B
[I-D.rescorla-tls-suiteb].  The rationale for this as stated by DoD
is that the public randomness for each side should be at least twice
as long as the security level for cryptographic parity, which makes
the 224 bits of randomness provided by the current TLS random values
insufficient.
```

“Cryptographic parity” is not a common phrase among cryptographers. It is not defined in the document, and its intended meaning is highly unclear. Furthermore, there is no known attack strategy that comes even close to exploiting the 224 bits of randomness used in TLS.

TLS encrypts data using a “master secret” computed from the server’s 224-bit random value, the client’s 224-bit random value, and a “pre-master secret”. The pre-master secret is the foundation of TLS security: in ECC cipher suites it is obtained from a DH key exchange between client and server, and in RSA cipher suites it is chosen by the client and encrypted to the RSA key of the server. The pre-master key is already at least twice as large as the security level. Even if the client constantly reuses the same pre-master secret and random value, the server has negligible chance of ever repeating its 224-bit random value with a properly

functioning RNG: a server generating an incredible 1000000000000000000 random 224-bit values has chance below 0.00000000000000000000000000000001 of seeing the same value twice.

The Internet-Draft contains no further explanation to support its allegation of 224 bits being insufficient. As far as we know, NSA has not attempted to defend this allegation in other venues. Meanwhile, to the extent that Extended Random is supported, it has an undisputed impact on the exploitability of Dual EC.

8 Certicom patents

The Canadian company Certicom (now part of Blackberry) has patents in multiple countries on

- Dual EC exploitation: the use of Dual EC for key escrow (i.e., for a deliberate back door) and
- Dual EC escrow avoidance: modifying Dual EC to avoid key escrow.

The patent filing history also shows that

- Certicom knew the Dual EC back door by January 2005;
- NSA was informed of the Dual EC back door by April 2005 (even if they did not know it earlier); and
- the patent application, including examples of Dual EC exploitation, was publicly available in July 2006, just a month after SP 800-90 was standardized.

This section cites several documents related to the patent applications and patents. We have a web page [5] with more details and cached copies of the documents.

A short summary of the general patenting process is that a party seeking to protect its intellectual property files a patent application with a national patent office. Often the initial application is a so-called "provisional" application, meaning that the details or the claims are not fully worked out; this gives an official time stamp on the possible invention. Often patent applications are submitted for more than one country using the "Patent Cooperation Treaty" (PCT) to start with the national filing. Certicom submitted the patent in the US and also filed for patents in Canada, Europe, and Japan.

8.1. Publicity and the avoidance thereof. In early 2005, Certicom began trying to patent both Dual EC exploitation and Dual EC escrow avoidance. The patent application lists Daniel R. L. Brown and Scott A. Vanstone as "inventors".

Certicom never drew public attention to these patenting efforts, or to the possibility of a back door in Dual EC. Their actions went generally unnoticed until 28 December 2013, when Certicom's patent application was announced by Lange in a presentation [4] with Bernstein and Heninger at the 30th Chaos Communication Congress, crediting a tweet [28] by "nymble" earlier in the month.

"At some point, I clued into the possibility of a backdoor, and, among other things, tried to make sure the possibility was at least publicly known, at first quietly: with a patent, and with a comment within my March 2006 eprint," Brown wrote in email [6] to the CFRG mailing list a few days after the presentation. "Later, others raised much more publicity, which seemed sufficient to me. I had expected such publicity to cause the proposers, X9F1 and NIST to withdraw the default P&Q from the standard."

8.2. The provisional patent application. The provisional patent application does not claim to have invented Dual EC per se, and does not clarify who invented Dual EC. It cites ANSI X9.82 [29, page 2, paragraph 0003]:

> The American National Standards Institute (ANSI) has set up an Accredited Standards Committee (ASC) X9 for the financial services industry, which is preparing a [sic] American National Standard (ANS) X9.82 for cryptographic random number generation (RNG). One of the RNG methods in the draft of X9.82, called Dual_EC_DRBG, uses elliptic curve cryptography (ECC) for its security. Dual_EC_DRBG will hereinafter be referred to as elliptic curve random number generation (ECRNG).

The provisional patent application describes the Dual EC back door [29, page 4, paragraphs 0010–0013]:

> The applicant has recognised that anybody who knows an integer d such that Q = dP ... can compute U from R as U = eR. ... The truncation function means that the truncated bits of R would have to be guessed. ... The updated state is u = z(U), so it can be determined from the correct value of R. Therefore knowledge of r and e allows one to determine the next state to within a number of possibilities somewhere between 2^6 and 2^{19}. This uncertainty will invariably be eliminated once another output is observed, whether directly or indirectly through a one-way function. ... It has therefore been identified by the applicant that this method potentially possesses a trapdoor, whereby standardizers or implementers of the algorithm may possess a piece of information with which they can use a single output and an instantiation of the RNG to determine all future states and output of the RNG, thereby completely compromising its security.

The provisional patent application also describes ideas of how to make random numbers available to "trusted law enforcement agents" or other "escrow administrators". For example [29, page 9, paragraph 0039]:

> In order for the escrow key to function with full effectiveness, the escrow administrator ... needs direct access to an ECRNG output value r that was generated before the ECRNG output value ... which is to be recovered. It is not sufficient to have indirect access to r via a one-way function or an encryption algorithm. ... A more seamless method

> may be applied for cryptographic applications. For example, in the SSL and TLS protocols, which are used for securing web (HTTP) traffic, a client and server perform a handshake in which their first actions are to exchange random values sent in the clear.

The provisional patent application also describes various ways to avoid the back door, such as [29, page 7, paragraphs 0028 and 0031] choosing P and Q as hashes of random seeds in a way similar to ANSI X9.62 (the idea that NSA told Johnson not to talk about; see Section 4.1):

> An arbitrary string is selected ... the hash is then converted to a field element ... regarded as the x-coordinate of Q ... To effectively prevent the existence of escrow keys, a verifiable Q should be accompanied with either a verifiable P or a pre-established P.

It is clear that Brown and Vanstone were aware of the Dual EC back door, and ways to exploit it, by January 2005 when the provisional patent application was filed. Technically, the applications were filed by Certicom, but both Brown and Vanstone signed a "Declaration and Power of Attorney For Patent Application" document in April 2006 [30, pages 39–41] declaring that they were the "inventors" and had reviewed the 23 January 2006 patent application, which includes a priority claim to the January 2005 provisional. Furthermore, the 23 January 2006 patent application contains all of the quotes given above, except that instead of "verifiable" it used the phrase "verifiably random".

8.3. Secrecy-order review. The United States Patent and Trademark Office (USPTO) forwards patent applications to "appropriate agencies" [37] to decide whether to impose secrecy orders on those applications:

> [Applications] are screened upon receipt in the USPTO for subject matter that, if disclosed, might impact the national security. Such applications are referred to the appropriate agencies for consideration of restrictions on disclosure of the subject matter as provided for in 35 U.S.C. 181.
> If a defense agency concludes that disclosure of the invention would be detrimental to the national security, a secrecy order is recommended to the Commissioner for Patents. The Commissioner then issues a Secrecy Order and withholds the publication of the application or the grant of a patent for such period as the national interest requires.

The USPTO referred Certicom's provisional patent application to the Department of Defense (DoD) for review. Eventually, on 2 February 2006, DoD returned a "Department of Defense: Access acknowledgment/Secrecy order recommendation for patent application" form [29, page 19] recommending against a secrecy order.

According to the USPTO, the referral letter was mailed on 7 April 2005, and the response was entered into PAIR on 27 February 2006. The response

itself states that the referral was on 7 March 2005 and that the response was forwarded on 7 February 2006.

The patent application was referred to DoD on 13 March 2006. The Navy responded "No comments" on 15 March 2006. NSA recommended against a secrecy order on 16 April 2007; see [30, page 48].

8.4. International patent applications. Certicom filed its patent application internationally under the PCT in 2006. The international publication number is WO2006/076804. This filing alone does not lead to national patents: the applicant needs to request examination in the designated countries (and pay the applicable fees). Searching for WO2006076804 on http://patentscope.wipo.int shows applications filed in Canada, Europe, and Japan.

The PCT stipulates (with certain exceptions) that international patent applications are published 18 months after the priority date. WIPO, the World Intellectual Property Organization, published the patent application on 27 July 2006 in full length online [7]. This means that a clear explanation of the back door and its (ab-)use was publicly available as of July 2006.

8.5. Resulting patents. Certicom received a European patent on 4 July 2012, and subsequently received patents in the United States, Japan, and Canada. The United States patent covers only escrow avoidance; the same seems to hold for the Canadian and Japanese patents. However, the European patent reaches farther than the United States patent: it covers both Dual EC exploitation and Dual EC escrow avoidance. The claims on escrow use are more refined than in the original patent application where they accounted for only one claim. The European patent lapsed in many countries because Certicom did not pay maintenance fees in those countries, but Certicom paid its January 2015 fees for France, Germany, the Netherlands, and Great Britain.

References

1. Bernhard Amann, Matthias Vallentin, Seth Hall, and Robin Sommer. Revisiting SSL: A large-scale study of the Internet's most trusted protocol, 2012. http://www.icsi.berkeley.edu/pubs/techreports/ICSI_TR-12-015.pdf.
2. James Ball, Julian Borger, and Glenn Greenwald. Revealed: how US and UK spy agencies defeat internet privacy and security. *The Guardian*, September 5 2013. http://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security.
3. Elaine Barker. Letter to Bruce Schneier, 2007. https://github.com/matthewdgreen/nistfoia/blob/master/6.4.2014%20production/109%20-%20Nov%2028%202007%20Letter%20to%20Bruce%20from%20Barker%20-%20Wired%20Commentary%20.pdf.
4. Daniel J. Bernstein, Nadia Heninger, and Tanja Lange. The year in crypto, 2013. Presentation at 30th Chaos Communication Congress, https://hyperelliptic.org/tanja/vortraege/talk-30C3.pdf.
5. Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. Certicom's patent applications regarding Dual EC key escrow, 2014. https://projectbullrun.org/dual-ec/patent.html.

6. Daniel R. L. Brown. Re: Dual_EC_DRBG, 2014. http://permalink.gmane.org/gmane.ietf.irtf.cfrg/2300.
7. Daniel R. L. Brown and Scott A. Vanstone. Elliptic curve random number generation, 2006. Patent application published by WIPO, http://tinyurl.com/oowkk36.
8. Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of Dual EC in TLS implementations. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 319–335. USENIX Association, August 2014. https://projectbullrun.org/dual-ec/documents/dualectls-20140606.pdf.
9. Richard George. Life at both ends of the barrel: an NSA targeting retrospective, 2014. http://vimeo.com/97891042, keynote talk at Infiltrate conference.
10. Kristian Gjøsteen. Comments on Dual-EC-DRBG/NIST SP 800-90, draft December 2005, 2006. http://www.math.ntnu.no/~kristiag/drafts/dual-ec-drbg-comments.pdf.
11. Matthew D. Green. Results of a recent FOIA for NIST documents related to the design of Dual EC DRBG, 2015. https://github.com/matthewdgreen/nistfoia.
12. Paul Hoffman. Additional random extension to TLS, February 2010. Internet-Draft version 01, http://tools.ietf.org/html/draft-hoffman-tls-additional-random-ext-01.
13. Paul Hoffman and Jerome Solinas. Additional PRF inputs for TLS, October 2009. Internet-Draft version 01, http://tools.ietf.org/html/draft-solinas-tls-additional-prf-input-01.
14. Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 27, IT Security techniques. US national body comments on ISO/IEC 2nd CD 18031. Attachment 10 to SC27 N3685, 2003. https://projectbullrun.org/dual-ec/documents/us-comment-to-iso.pdf.
15. Don Johnson. Minding our Ps and Qs in Dual_EC, 2004. http://csrc.nist.gov/groups/ST/crypto-review/documents/Email_Oct%2027%202004%20Don%20Johnson%20to%20John%20Kelsey.pdf.
16. Don Johnson. Number theoretic DRBGs, 2004. http://csrc.nist.gov/groups/ST/toolkit/documents/rng/NumberTheoreticDRBG.pdf.
17. John Kelsey. 800-90 and Dual EC DRBG, 2013. http://csrc.nist.gov/groups/SMA/ispab/documents/minutes/2013-12/nist_cryptography_800-90.pdf.
18. John Kelsey. Dual EC in X9.82 and SP 800-90, 2014. http://csrc.nist.gov/groups/ST/crypto-review/documents/dualec_in_X982_and_sp800-90.pdf.
19. Jeff Larson, Nicole Perlroth, and Scott Shane. Revealed: The NSA's secret campaign to crack, undermine Internet security. *ProPublica*, September 2013. https://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption.
20. Joseph Menn. Exclusive: Secret contract tied NSA and security industry pioneer. *Reuters*, December 2013. http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220.
21. National Institute for Standards and Technology. DRBG validation list. http://csrc.nist.gov/groups/STM/cavp/documents/drbg/drbgval.html.
22. National Institute for Standards and Technology. Internal draft of X9.82 section 9.12, 2004? https://github.com/matthewdgreen/nistfoia/blob/master/6.4.2014%20production/011%20-%209.12%20Choosing%20a%20DRBG%20Algorithm.pdf, received through FOIA.

23. National Institute for Standards and Technology. RNG workshop and standards development, 2004. http://csrc.nist.gov/groups/ST/toolkit/random_number.html#RNG%20WSD.
24. National Institute for Standards and Technology. The NIST SP 800-90A Deterministic Random Bit Generator Validation System (DRBGVS); current version from 2013, first version from 2009, 2013. http://csrc.nist.gov/groups/STM/cavp/documents/drbg/DRBGVS.pdf.
25. National Institute for Standards and Technology. Compilation of public comments on 2005 draft of SP 800-90, 2014. http://csrc.nist.gov/groups/ST/toolkit/documents/CommentsSP800-90_2006.pdf.
26. National Institute for Standards and Technology. NIST FOIA material released to COV: X9.82 and NIST SP 800-90 process (June 10, 2014), 2014. http://csrc.nist.gov/groups/ST/crypto-review/review_materials.html.
27. National Institute of Standards and Technology. Special Publication 800-90: Recommendation for random number generation using deterministic random bit generators, 2012. First version June 2006, second version March 2007, http://csrc.nist.gov/publications/PubsSPs.html#800-90A.
28. nymble. Interesting patent on use of ECC random number generator for 'escrow'. Designed as backdoor in 2005. Twitter post on December 3, 2013. https://twitter.com/nymble/status/408023522284285952.
29. Patent Application Information Retrieval (PAIR). Image file wrapper for provisional application 60644982, 2005. https://projectbullrun.org/dual-ec/documents/60644982.pdf.
30. Patent Application Information Retrieval (PAIR). Image file wrapper for patent application 11336814, 2006. https://projectbullrun.org/dual-ec/documents/11336814.pdf.
31. Nicole Perlroth, Jeff Larson, and Scott Shane. N.S.A. able to foil basic safeguards of privacy on web. *International New York Times*, September 2013. http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html.
32. Eric Rescorla and Margaret Salter. Opaque PRF inputs for TLS, December 2006. Internet-Draft version 00, http://tools.ietf.org/html/draft-rescorla-tls-opaque-prf-input-00.
33. Eric Rescorla and Margaret Salter. Extended random values for TLS, March 2009. Internet-Draft version 02, http://tools.ietf.org/html/draft-rescorla-tls-extended-random-02.
34. Bruce Schneier. Did NSA put a secret backdoor in new encryption standard?, 2007. http://archive.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115.
35. Berry Schoenmakers and Andrey Sidorenko. Cryptanalysis of the Dual Elliptic Curve pseudorandom generator. Cryptology ePrint Archive, Report 2006/190, 2006. https://eprint.iacr.org/2006/190.
36. Dan Shumow and Niels Ferguson. On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. CRYPTO 2007 Rump Session, August 2007. http://rump2007.cr.yp.to/15-shumow.pdf.
37. United States Patent and Trademark Office. Review of applications for national security and property rights issues, 2013. Manual of Patent Examining Procedure, Section 115, http://www.uspto.gov/web/offices/pac/mpep/s115.html.