An Examination of Various Random Number Generator Weaknesses

Ernesto Martínez Simon Lammer

Cryptanalysis ST 2023

Abstract

This report aims to showcase the importance of Random Number Generators (RNGs) in the cryptographic stack by showing a set of three real world examples where the cryptosystems could be broken due to weak RNGs. First scenario shown is the naive "Sony's PS3 ECDSA" misunderstanding of random number generation and how it leaked it's PS3 ECDSA key. Secondly the case "A Novel Related Nonce Attack for ECDSA" is shown and demonstrates how the kind of random number generator used matters and could break ECDSA signatures in real world implementations such as the Bitcoin network. Finally the infamous "Dual Elliptic Curve Deterministic Random Bit Generator" or `DUAL_EC_DRBG` is explained cryptographically and historically, alongside it's non-provable backdoor which could in practise break most of the cryptosystems where this RNG was used.

Keywords: RNG · PRNG · ECDSA · DUAL_EC_DRBG

1 Introduction

Random Number Generators are nowadays commonly found in cryptosystems, from key generation to signatures and encryption. It's a widely researched topic in cryptography and involves multiple interesting aspects such as entropy sources and how to safely use the generated randomness.

Due to their necessity for modern cryptography, our **motivation** was to research and study real world situations where RNGs caused a major breakage in the cryptosystems they were used for. Our main goal was to select such situations and understand what and how happened.

The **main questions** we aimed for the report were, for each topic, what was the mathematical root cause that induced the fault in the algorithm, and how it could be prevented. We also wanted to know, naively, the impact of the improper or weak RNG usage had in a real world application. We **contribute** by providing a summary and in-depth explanations of multiple cases of RNG-related weaknesses that happened in practise with critical impacts.

Outline. In Section 2, Section 3 and Section 4 we examine the main topics of research of the report: PS3 ECDSA Nonce, "Polynonce" and `DUAL_EC_RBG`. In each section, the introductory content appears in order to understand the main topic before showing it's studied weaknesses. Finally, in Section 5 we conclude the report with a summary of our findings and thoughts.

2 PS3 ECDSA Nonce Misuse

2.1 History [EDN11]

The Playstation 3 (PS3) was released by Sony in 2006. It's main purpose is to serve as a game console. Additionally it also enabled media playback. A more niche function was to load a different operating system using the PS3's OtherOs feature. However, in April 2010, Sony decided not only to refrain from including this feature in the PS3 Slim (which they already announced in 2009), but also to remove it retroactively and unilaterally from the whole PS3 lineup via an automated firmware upgrade. This sparked motivation of some groups to find a way of restoring the ability to use operating systems different from the default one.

In this quest, by December 2010 the hacker group "failOverflow" found - among several other exploits - a way to reveal the private key used for signing executables (other than games) by Sony. Even though they chose not to publish the recovered private key, this was done a year later by "geohot"; which caused Sony to file a law suit that was eventually settled without revealing its full terms (but including a permanent injunction preventing "geohot" from reposting the original security system information) [Hen13].

2.2 Attack details [fai10; EDN11]

To learn how the attack works, it is necessary to first understand how the used signature algorithm ECDSA works.

$k \xleftarrow{\$} [1, n-1]$	Randomly choose from uniform distribution.
$R = kG = (x_R, y_R)$	
$r = x_R \bmod n$	If $r = 0$ restart the algorithm.
$e = h(m)$	
$s = k^{-1}(e + d \times r) \bmod n$	If $s = 0$ restart the algorithm.

Figure 1: Creating an ECDSA signature (r, s) from message m, private key d, EC basis point G, order n of G, and hash function h.

In ECDSA it is of utmost importance, that the nonce be generated randomly per-signature in an unpredictable way from a uniform distribution over a certain interval

(see Figure 1) and immediately discarded afterwards. If the nonce k used in a signature leaks, the private key d used to create said signature can be trivially recovered, because

$$s = \frac{e + d \cdot r}{k} \quad \equiv \quad d = \frac{s \cdot k - e}{r} \mod n,$$

where s, e, r and n are publicly known.

So, how did failOverflow manage to reveal Sony's private ECDSA key? Well, let's put it this way: If a security flaw can be summarized by a *xkcd* comic, any system built upon it is not very secure.

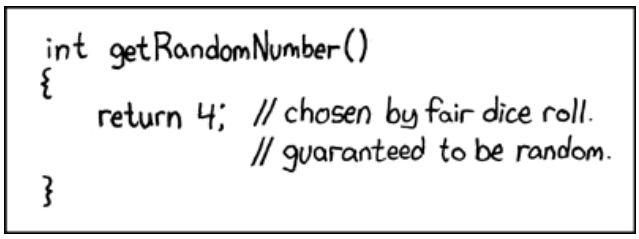


Figure 2: xkcd 221 − Random Number

Yes, they simply used a constant value for the - supposed to be random - nonce k. This allowed attackers to easily retrieve the private key d from two signatures:

$$\begin{aligned}
d &= \frac{s_1 \cdot k - e_1}{r_1} \\
d &= \frac{s_2 \cdot k - e_2}{r_2} \\
\frac{s_1 \cdot k - e_1}{r_1} &= \frac{s_2 \cdot k - e_2}{r_2} \\
k &= \frac{e_1 \cdot r_2 - e_2 \cdot r_1}{s_1 \cdot r_2 - s_2 \cdot r_1}
\end{aligned}$$

$$d = \frac{s_1}{r_1} \cdot \frac{e_1 \cdot r_2 - e_2 \cdot r_1}{s_1 \cdot r_2 - s_2 \cdot r_1} - \frac{e_1}{r_1} \mod n$$

2.3 Impact

The chain of trust system that was supposed to only allow valid operating systems to run on the PS3 does not support changing/revoking the root signature key (which is the one that was leaked). Since that key has been published, everyone could sign arbitrary software with it. Sony has not found a way to fix this issue in the PS3, but has surely moved on to a better pseudo-random number generator for other devices.

3 A Novel Related Nonce Attack for ECDSA

The elliptic curve digital signature algorithm (ECDSA) is popular and often used to create digital signatures. Among its use cases are TLS, SSL, and several blockchain protocols including Bitcoin and Ethereum. When creating such signatures, the signer has to choose a secret per-message nonce k (see Figure 1). Leakage of this secret nonce k allows recovery of the private key d (as explained in Section 2.2).

There has been some work to choose k deterministically [Por13], but it is usually chosen as a per signature (pseudo-)random value [Mac23, p.5]. If the supposedly random choices were not unpredictably selected from a uniform distribution (i.e., because of a flawed PRNG), the signature's security guarantees break down (for the used private key d).

In this novel related nonce attack for ECDSA [Mac23], Macchetti presents a way to recover the nonces used in a small number of signatures under certain conditions.

3.1 Attack Details [Mac23]

Every signature can be written as

$$k_i = \frac{h_i}{s_i} + \frac{r_i}{s_i} \cdot d \mod n.$$

The point $R_i = k_i G$ can be recovered from the it's x-coordinate r_i that is the first half of the signature. However, retrieving k_i from R_i is (assumed to be) hard - as hard as solving the discrete logarithm problem over the elliptic curve. Nevertheless, a relationship between nonces of consecutive messages can be found:

$$\begin{aligned} k_i &= \frac{h_i}{s_i} + \frac{r_i}{s_i} \cdot d \\ d &= \frac{k_i s_i - h_i}{r_i} \end{aligned}$$

$$\begin{aligned} \frac{k_0 s_0 - h_0}{r_0} &= \frac{k_1 s_1 - h_1}{r_1} \\ (k_0 s_0 - h_0) r_1 &= (k_1 \cdot s_1 - h_1) r_0 \\ k_1 &= \frac{\frac{(k_0 \cdot s_0 - h_0) r_1}{r_0} + h_1}{s_1} \\ &= \frac{k_0 r_1 s_0 - h_0 r_1 + h_1 r_0}{r_0 s_1} \\ &= \frac{r_1 s_0}{r_0 s_1} k_0 + \frac{h_1 r_0 - h_0 r_1}{r_0 s_1} \end{aligned}$$

Naming these coefficients $u = \frac{r_1 s_0}{r_0 s_1}$ and $v = \frac{h_1 r_0 - h_0 r_1}{r_0 s_1}$, the relationship between signatures can be stated as

$$R_1 = k_1 G = (u k_0 + v) G = u R_0 + v G.$$

Therefore, finding the nonces of N points R_i is only one instance of the DLP instead of N instances.

If the nonces used for N signatures obey a multivariate polynomial equation, that equation can be rewritten as a univariate polynomial dependent only on the private key and public values. If coefficients a_i and exponents e_i in

$$a_0 k_0^{e_0} + a_1 k_1^{e_1} + a_2 k_2^{e_2} + \cdots + a_N = 0$$

are known, then this can be reformulated as

$$a_0 \left(\frac{h_0}{s_0} + \frac{r_0}{s_0} d\right)^{e_0} + a_1 \left(\frac{h_1}{s_1} + \frac{r_1}{s_1} d\right)^{e_1} + a_2 \left(\frac{h_2}{s_2} + \frac{r_2}{s_2} d\right)^{e_2} + \cdots + a_N = 0.$$

Computers can quickly find this polynomial's roots, wherein the private key d will appear.

The new attack works if the PRNG used to generate the nonce k uses arbitrary-degree recurrence relations modulo the order n of the elliptic curve's generator point G. In that case, only the very first nonce k_0 is chosen with proper randomness. For example, a Linear Congruential generator (LCG) with multiplier a_1 and increment a_0, where $k_i = a_1 \cdot k_{i-1} + a_0 \mod n$; or a Quadratic Congruential generator (QCG) with coefficients a_i, where $k_i = a_2 * k_{i-1} + a_1 * k_{i-1} + a_0 \mod n$. Note that nonce-reuse is a special case of these relations, where all coefficients but a_0 are non-zero (and a_0 is equal to the repeated nonce). In general, the nonces generated by such a PRNG relate as follows:

$$\begin{aligned}
k_1 &= a_{N-3} k_0^{N-3} + a_{N-4} k_0^{N-4} + \cdots + a_1 k_0 + a_0 \\
k_2 &= a_{N-3} k_1^{N-3} + a_{N-4} k_1^{N-4} + \cdots + a_1 k_1 + a_0 \\
k_3 &= a_{N-3} k_2^{N-3} + a_{N-4} k_2^{N-4} + \cdots + a_1 k_2 + a_0 \\
&\cdots \\
k_{N-1} &= a_{N-3} k_{N-2}^{N-3} + a_{N-4} k_{N-2}^{N-4} + \cdots + a_1 k_{N-2} + a_0
\end{aligned}$$

The goal is to produce a polynomial which only depends on the nonces and not on the (unknown) coefficients a_i. Therein the nonce dependency can then be substituted with a dependency on the private key d using the previously shown relation $k_i = \frac{h_i}{s_i} + \frac{r_i}{s_i} \cdot d$, after which the roots of the polynomial can be found to reveal the private key d in little time.

Let's show the simplest case, which uses a LCG:

$$
\begin{aligned}
k_0 &\xleftarrow{\$} [1, n-1] \\
k_1 &= a_1 k_0 + a_0 \\
k_2 &= a_1 k_1 + a_0 \\
k_3 &= a_1 k_2 + a_0
\end{aligned}
$$

$$
\begin{aligned}
k_1 - k_2 &= a_1(k_0 - k_1) \\
a_1 &= \frac{k_1 - k_2}{k_0 - k_1} \\
k_2 - k_3 &= a_1(k_1 - k_2) \\
a_1 &= \frac{k_2 - k_3}{k_1 - k_2}
\end{aligned}
$$

$$
\begin{aligned}
\frac{k_1 - k_2}{k_0 - k_1} &= \frac{k_2 - k_3}{k_1 - k_2} \\
0 &= (k_1 - k_2)^2 - (k_2 - k_3)(k_0 - k_1)
\end{aligned}
$$

Similar equations can be generated for higher-order congruential generators when more signatures were observed; e.g. N=5 signatures for a QCG yields a 4-degree polynomial, or N=6 signatures for a Cubic Congruential Generator (CCG) yields a 7-degree polynomial.

The paper also describes how a set of $N-1$ signatures with truly random nonces can be extended with a *rogue nonce* to have at least one ordering of signatures that is vulnerable to the attack described above. Alas, finding the correct ordering is no faster than brute-force.

3.2 Impact

Using the provided implementation, private keys from vulnerable sets of signatures can be found very quickly (within one second for low-degree generators; and about 6.5 s for N=16, which yields a 92-degree polynomial) [Mac23]. If the nonces are truly generated randomly, any polynomial relation between them should have a very high degree - which would render this attack infeasible to apply in practice.

Signatures of the bitcoin blockchain were tested for this vulnerability. Creating a workable input dataset for the attack was not trivial since the original messages signed by the listed signatures are not included, but instead expected to be recomputed by signature verifiers. Retrieving the original messages for all signatures from the raw blockchain took 24 hours. The blockchain contained signatures from roughly 424 million unique public keys, yet the number of signatures per key declines rapidly as can be seen in Figure 3. [Ami23]

Signatures that stem from public keys with at least N=5 signatures were tested (this could exploit up to quadratic polynomial nonce relations) on a 128-core VM in under

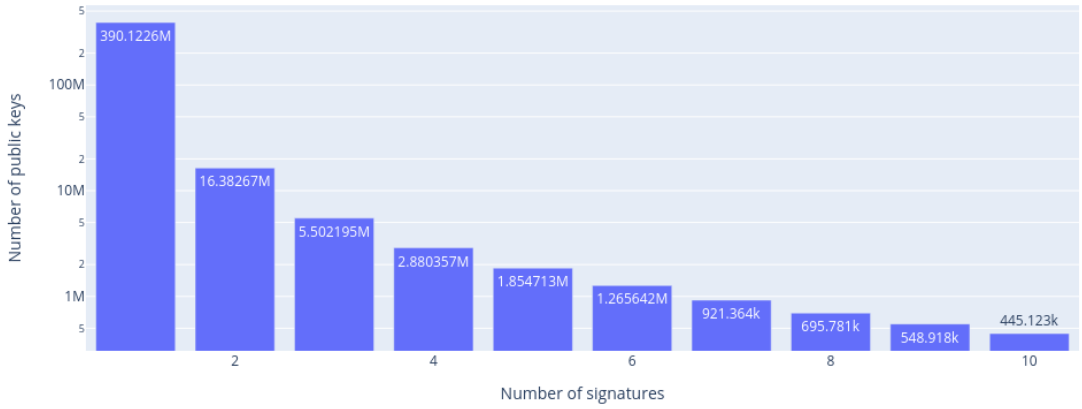


Figure 3: Amount of public keys by number of signatures in the bitcoin blockchain (until 2022-09-05). [Ami23]

3 days. The estimated cost of this analysis is just under 300 USD. 762 unique bitcoin wallets were broken in this procedure. The attack was rerun with N=4 to find another 11 weak wallets. Alas, all of these 773 wallets had a zero balance and reused nonces (the simplest polynomial relation). About 144 BTC (9.4 million USD) were possibly previously stolen from these wallets by exploiting their reused nonces. [Ami23]

The attack was also applied to the Ethereum blockchain and server TLS connections without practical success. However, there are many more use cases for ECDSA that have not been tested. The authors released their code in addition to the information, to enable others to create more robust systems. [Ami23]

4 Dual Elliptic Curve Deterministic Random Bit Generator

Dual Elliptic Curve Deterministic RBG (Random Bit Generator) or `DUAL_EC_DRBG` was[1] a cryptographically secure pseudo random number generator (CSPRNG) developed by the NSA [Kel14, p.3] that used elliptic curves to generate a deterministic sequence of random bits based on a seed. The algorithm got standardized by ANSI, ISO and NIST [BLN15, p.1] during seven years[2], and even got to be the default algorithm for DRBG in the "RSA BSAFE" cryptography library from RSA Security [BLN15, p.2]. Besides it's apparent success, the algorithm was heavily criticised by the cryptography community [Kel14; Gre13b; Ada14] generally due to a RNG bias, poor performance, lack of reduction proof and the suspects of a possible NSA kleoptographic[3] backdoor.

[1] We took the liberty of removing the secure attribute, assuming that the problems found and stated in this report, are enough justification to discard the qualifier.

[2] In the specific case of NIST

[3] The term kleptographic backdoor was introduced by Adam Young and Moti Yung in 1996 [YY97a]. Refers to a subtle and secure method of stealing information commonly using asymmetric cryptography.

4.1 Background

Before starting with the contents of the section, this small subsection aims to provide a starter mathematical background in order to be able to properly understand the rest of the contents. If the reader is confortable with basic concepts from elliptic curves such as it's order, scalar multiplication, operators and the Weierstrass Form, it can freely skip the subsection.

"Elliptic Curve Cryptography" is a subfield of cryptography (or a type of cryptosystems) that makes use of Elliptic Curves in order to achieve a certain functionality. For example, they are commonly use due to their discrete-log number-theoretic hard problem.

An Elliptic Curve is formed by solutions (X, Y) of an equation in Weierstrass Form Equation 1, Equation 2 (simplified form for $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ or any $\mathbb{F}_{p^m} (p \neq 2, 3)$) and Equation 3 (simplified form for $\mathbb{F}_{2^m}$).

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

$$y^2 = x^3 + ax + b \tag{2}$$

$$y^2 + xy = x^3 + ax^2 + b \tag{3}$$

In an elliptic curve, a point P it's represented by it's coordinates (X, Y). We can apply operations to the points such as the inverse, addition, double, scalar multiplication, etc.

In cryptography we don't use "nice" curves as the one shown in Figure 4, which are in $\mathbb{R}$. We operate over finite fields (Figure 5) which make some operations hard to compute, as in exponentiation with modular arithmetic.

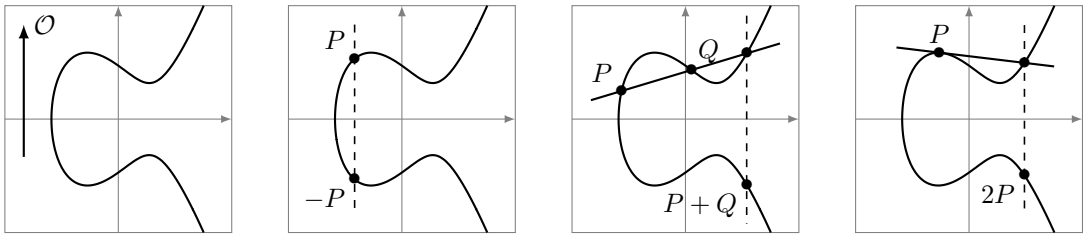
For this section, the most important concept to understand is that elliptic curves have a number-theoretic problem similar to the discrete-log in diffie hellman with exponentiation. If we take a scalar d and we multiply it by a point G, the resulting point $P = \sum_1^k G$ along G are not enough information to recover the secret d. This is called the elliptic curve discrete log problem. We can translate discrete log problems like the diffie hellman key generation to it's elliptic curve version by changing the group elements to elliptic curve points.

Other concept of an elliptic curve is it's order. The order of an elliptic curve is the number of points (X, Y) including $\mathcal{O}$ (neutral element). As points are the possible solutions to the elliptic curve equation, it's basically all it's possible solutions.

For understanding `DUAL_EC_DRBG` you have to know that it involves two elliptic curve points P and Q which belong to the same predefined elliptic curve. You can take an overview of this parameters in Figure 6.

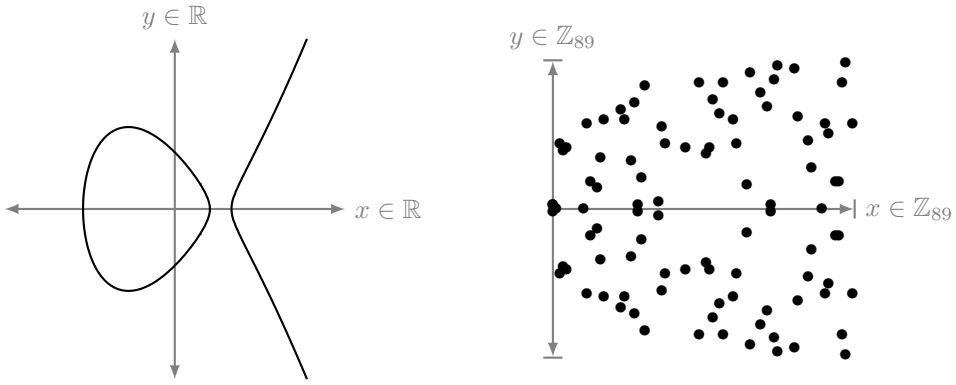
4.2 History & Standarization Process

Understanding `DUAL_EC_DRBG`'s history is the keystone to understand the nature and criticism of it's algorithm. The origins of it, along with the standarization process, are filled with irregularities and overlookings that point to the possible NSA backdoor. As the existence of a backdoor cannot be mathematically proven −reasons shown in Section 4.4− history, accompanied with the critical flaws that we'll show, are the way to suspect of the existence of a NSA kleptographic backdoor.



Neutral element $\mathcal{O}$ | Inverse element $-P$ | Addition $P+Q$ "Chord rule" | Doubling $P+P$ "Tangent rule"

Figure 4: Elliptic Curve Operations. Source `www.iacr.org/authors/tikz/`



$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$ | $y^2 = x^3 - 2x + 1$ over $\mathbb{Z}_{89}$

Figure 5: Elliptic Curves over $\mathbb{R}$ and $\mathbb{Z}_{89}$. Source: `www.iacr.org/authors/tikz/`

1997 Adam Young and Moti Young present two papers at Eurocrypt 1997 and Crypto 1997 showing how cryptography could be used as an attack resource [YY97a] instead of a defensive one by building a Diffie Hellman kleptographic backdoor [YY97b]. Years later it's shown that the `DUAL_EC_DRBG` backdoor is similar to the Diffie Hellman backdoor presented.

2000 ANSI X9.82 standarization process starts and the NSA pushes to include `DUAL_EC_DRBG` in ANSI X9.82 standard [Gre13a]. This shows that the algorithm originated as an ANSI standard, then NIST and ISO adopted it too [Gre13a].

early 2004 A draft of ANSI.X9.82 is published and includes `DUAL_EC_DRBG`. Also, RSA Security makes `DUAL_EC_DRBG` the default CSPRNG in the BSAFE cryptographic library. Ten years later, Reuter reports that this decision came from a 10 million USD deal that NSA had with RSA Security *"... RSA received $10 million in a deal that set the NSA formula as the preferred, or default, method for number generation in the BSafe software,*

according to two sources familiar with the contract. ... [$10 million] represented more than a third of the revenue that the relevant division at RSA had taken during the entire previous year ...” [Men13].

late 2004 NIST RNG Workshop starts. First question of P and Q appears: John Kelsey asks Don Johnson (NIST) where Q came from in `DUAL_EC_DRBG` [Kel14]. NIST response is *“Q is (in essence) the public key for some random private key. It could also be generated like a(nother) canonical G, but NSA kiboshed this idea, and I was not allowed to publicly discuss it, just in case you may think of going there.”*

early 2005 Certicom (now Blackberry) fills a patent showing a method to build a deliberate backdoor in dual EC (and how to prevent it) by having the controll of P and Q generation [BLN15, Chapter 8]. They explain the importance of this backdoor and how it could be used to decrypt traffic: *“in this case, trusted law enforcement agents may need to decrypt encrypted traffic of criminals, and to do this they may want to be able to use an escrow key to recover the encryption key”*. It's clear that the authors were aware of the Dual EC backdoor and how to exploit it by January 2005. The patent was referred to the Department of Defense (DoD) of the United States and the NSA (in 2007), which recommended against a secrecy order [BV05, p.48].

late 2005 ISO/IEC 18032:2005 and NIST SP 800-90A draft are published, both include `DUAL_EC_DRBG`. The possibility of a trapdoor by controlling (P, Q) in `DUAL_EC_DRBG` is formally commented in X9. The NSA's response was *“NSA generated (P, Q) in a secure, classified way.”*. NSA also pointed out that DRBG was originally created for the national security community, and that they wanted to get FIPS validated devices that used it. They suggested that it would be reasonable to let other users to generate their own (P, Q) [Kel14, p.24]. Years later, it's shown that almost all the implementations used the default NSA parameters and the usage of custom values was discouraged as you couldn't get FIPS[4] 140 certification [BK12, p.77].

early 2006 Kristian Gjøsteen publishes a comment [Gjo05] on the NIST SP 800-90A stating that there is a bias in the random bits produced by `DUAL_EC_DRBG` that could allow predicting bits with a certain advantage [Gjo05, p.8]. Kristian clearly states that this sucess rate would not be acceptable in PRBG based on symmetric primitives, and neither should be based on number-theoretic assumptions. Berry S. and Andrey S. improve Kristian's attack afterwards [SS06]. Daniel R. L. Brown publishes a paper with extensions to the previous one and anticipating the findings of Shumow and Ferguson's [SF07] that a backdoor could possibly exist.

mid 2006 NIST SP 800-90A is published including `DUAL_EC_DRBG` without any previously stated problem fixed. Three other algorithms are presented (`HASH_DRBG`, `HMAC_DRBG`,

[4]Federal Information Processing Standards. It's a U.S. goverment computer security standard that specifies certain requirements for cryptography modules

CTR_DRBG) as an alternative, one of them also from the NSA (HASH_DRBG) but with NIST modifications. Years later, after the discovery of the backdoor, it's known that the solutions proposed to reduce the bias problem (unfixed in the standard) would have (partially) closed the backdoor [BLN15, p.5].

2007 Dan Shumow and Niels Ferguson give a presentation [SF07] that demonstrates how an attacker can include a backdoor in DUAL_EC_DRBG and recover the full internal state with the control of (P, Q). An article [Sch07] is published in Wired suggesting that the NSA could have a backdoor in DUAL_EC_DRBG based on this presentation. NIST rejected retiring DUAL_EC_DRBG stating that they had *"... no evidence that anyone has, or will ever have, the secret numbers for the backdoor ..."* [BLN15; Kel14, p.8, p.32].

2013 Edward Snowden's NSA leaks are published. NSA's Bullrun program existence is revealed, the program aimed *"to covertly introduce weaknesses into the encryption standards ..."*. NSA Security advises it's customers to stop using DUAL_EC_DRBG in BSAFE cryptography library.

april 2014 NIST removes DUAL_EC_DRBG [NIS14]. Checkoway et al. shows the possiblity of building a SSL/TLS backdoor with DUAL_EC_DRBG [Che+14].

may 20214 Richard George, NSA's Technical Director of "Information Assurance Directorate" made the following statement: *"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."* [BLN15, p.10].

4.3 How DUAL_EC_DRBG Works

In this section we will explain the inner workings of DUAL_EC_DRBG from a general perspective. We won't detail it's security and full reduction proof, but we'll study how the states are updated and which number-theoretic problem defends the algorithm, also we'll study how additional input can be fed into it. In the following Section 4.4 we'll show the problems this algorithm has and how they work.

The general overview of the algorithm can be seen in Figure 6 and Figure 7. Figure 7 shows the separation between the inner state and the output "state", it also shows the order in which the operations are applied and to which operands. Figure 6 is a more detailed diagram showing how the state feedbacks it's own output by itself, it also details more visually which operations are applied in each state.

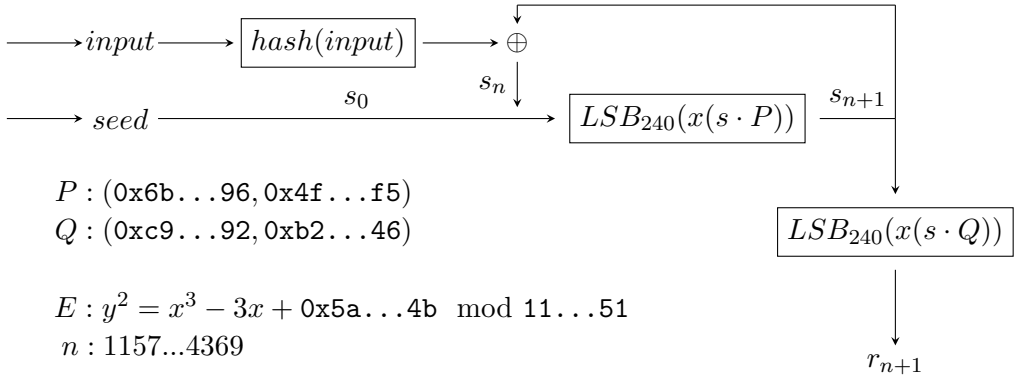


Figure 6: DUAL_EC_DRBG detailed diagram focusing on the inner state feedback *seed* is only used once. *input* is optional. r is the random output. $x(s \cdot P)$ represent the x coordinate of the $s \cdot P$ operation result in the curve. $LSB_{240}(x)$ represents the least significant 240 bits of x. Constants are for demonstration and come from NIST 800-90A Appendix A.1.1 Curve P-256

A textual example of usage with an inital seed and 240 output bits:

1. User starts by providing a *seed*. This will be the input to the first inner state loop
2. Inner state calculates $s = LSB_{240}(x(seed \cdot P))$ and passes it to the output part and also feedbacks it to it's input again.
3. Output bits are calculated as $r = LSB_{240}(x(s \cdot Q))$
4. Next inner state input is s, so we re-calculate $s = LSB_{240}(x(s \cdot P))$. If the user would like to enter additional input, s would, after that step, get xor-ed as $s = s \oplus hash(input)$

We can see how the inner state can be viewed as a chain of operations (Figure 7, Equation 4), in our case $f_P()$ involves the multiplication of a scalar with a point P in a eliptic curve. This makes returning back to the previous state impossible (would mean the ability to break the discrete log problem in eliptic curves).

$$s_0 = seed \rightarrow s_1 = f_P(s_0) \rightarrow \cdots \rightarrow s_n = f_P(s_{n-1}) \tag{4}$$

The output applies a non-reversible function to preclude calculating the inner state s_n based on r_n, in this case is also the multiplication of a scalar (the state) and a point Q in the curve. The definition of $g()$ doesn't matter for the discrete-log based "protection", g discards Y coordinate from the output point and takes LSB_{240} bits form the resulting X. See how this operation is represented in the green part of Figure 7.

$$r_n = g(s_n \cdot Q) \tag{5}$$

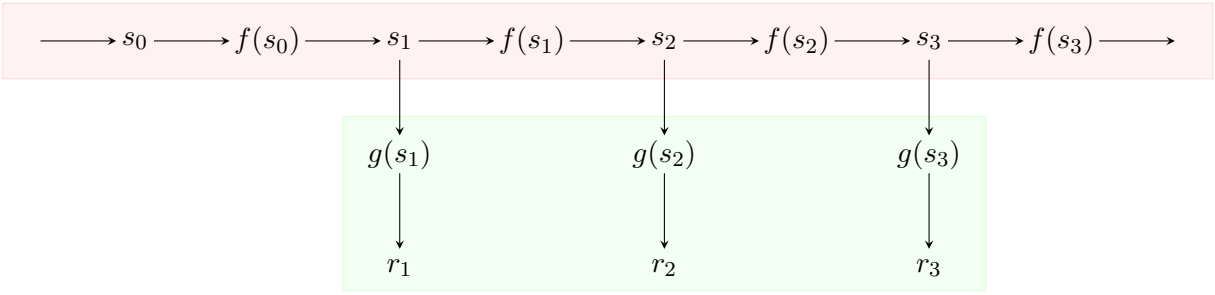


Figure 7: Sequence diagram for generating three 240 bit values of randomness in `DUAL_EC_DRBG` (without additional input). s_0 is meant to be the seed. $f(s)$ is a function that multiplies the scalar s by a curve point P and returns the 240 least significant bits for the X coordinate of the resulting point. $g(s)$ is an equivalent function but operating with a curve point Q. The inner part is shown in *red* and the outer output part is shown in *green*.

As we can see, Dual Eliptic Curve DRBG security is based on the intractability of the $P = bG$ scalar finding problem in secure elliptic curves (elliptic curve discrete log problem) as previously explained (Section 4.1). The inner state is protected by the stated elliptic curve discrete log problem. Recovering the inner state would imply the ability of predicting all future outputs and in consequence, breaking the algorithm. The green part of Figure 7 is the one that really protects the algorithm, the red part makes impossible to learn previous values, but as you can compute forward, the algorithm would already be broken as you could predict, with a 100% success rate, all future generated bits samples.

4.4 The Backdoor

Although the existence of a NSA backdoor cannot be mathematically proved nowadays, the mathematical possibility of introducing a backdoor, along with the history of standarization irregularities it had, make the NSA backdoor in `DUAL_EC_DRBG` a likely scenario.

`DUAL_EC_DRBG` includes two crucial parameters, the points in the curve P, Q. P has to be our generator for the choosen curve [BK12, p.77], while Q holds no special relation to P, it has to be a point on the curve and it's used to protect the inner state.

If Q is randomly choosen in a secure way, there is no possibility of a backdoor and the algorithm is safe from backdoors. The problem comes where P, Q come predefined (as it's indirectly enforced). A malicious attacker could craft P, Q and a secret as following:

Attacker selects a secret scalar d

Now there is a hidden relation between P and Q

$$d \cdot P = Q \rightarrow e \cdot d \cdot P = e \cdot Q \rightarrow P = e \cdot Q \tag{6}$$

Finds e such that $e \cdot d = 1 \mod r$ (r is order of EC)

Now based on Equation 6, if an attacker gives you the resulting (P, Q) pair, you cannot prove the existence of a d or e that relates P and Q because that would mean solving the discrete log problem in eliptic curves: $P = e \cdot Q$ (find scalar e based on (P, Q)).

Now if you run `DUAL_EC_DRBG` with the attacker-given parameters and the attacker also has access to the random bits r of output (something very common, for example in some padding scheme), it can run the following attack:

1. Bruteforce 2^{16} bits combination to recover the X coordinate of the elliptic curve point. Remember the output is 256 bits and we truncate it to 240 bits only.

2. Solve the elliptic curve equation for candidates to get Y, now an attacker would have (X, Y) candidates. Discard any guess that cannot belong to the curve.

3. Knowing that $X = r_n = s_n \cdot Q$, an attacker could multiply X (which is the random output intercepted r_n) by e to build the following equation:

$$\overbrace{e}^{\text{Multiply by hidden calculated } e} \cdot X = \overbrace{e}^{\text{Multiply by hidden calculated } e} \cdot s_n \cdot Q \rightarrow s_n \cdot \underbrace{eQ}_{\text{The attacker nows the hidden } e \cdot Q = P \text{ relation!}} \rightarrow \underbrace{s_n \cdot P}_{\text{Next RNG inner state!}} \tag{7}$$

4. See how the attacker "bypassed" the discrete log elliptic curve problem by previously knowing the relation. Now the attacker knows $s_n \cdot P$ which is the internal state.

5. An attacker can compute all future random bits. In case the user introduces random input (see Figure 6), the attacker would have to re-run the attack or to guess the input of the user (could be doable if naive inputs such as time are used).

Again, demonstrating the existence such relation between P and Q is imposible nowadays as it would mean breaking the elliptic curve discrete log problem. Admire[5] the niceness of a backdoor that, even with clear evidence, cannot be demonstrated and in consecuence, legally atributed to any group.

Now a lot of questions arise regarding the choices of P, Q and the possibility of making the bruteforcing step intractable. They were indirectly answered in Section 4.2 but we'll summarize them here explicitly.

How default (P, Q) were generated? Textually: *"NSA generated (P, Q) in a secure, classified way."* [BLN15, p.9].

Why not discard more bits? Discarding more bits would harden the bruteforce step, even could solve the random bias [Gjo05] problem as stated in [Kel14, p.21]. The idea was never implemented, the performance of the algorithm was bad enough that discarding bits would imply more "rounds" to generate the same ammount of random bits.

[5] As much as we do.

Why not randomly generating P, Q? It was an available option lightly explained in NIST's standard appendix. By using custom (P, Q) pairs, the device wouldn't get FIPS 140 certification [BK12, p.77]. At the end, most libraries used standard, NSA-provided, (P, Q) values.

4.4.1 Real Life Exploitability

Even if at this point `DUAL_EC_DRBG` is completely broken, depending on the design and/or implementation of certain protocols can make the exploitation a bit harded. In this small section we'll give a small showcase of an example, and how again, RSA Security, is involved.

Assumming we want to intercept TLS traffic from machines that are using `DUAL_EC_DRBG`, we would, as the NSA, need 240 consecutive bits of randomness coming from `DUAL_EC_DRBG` for which we have the kleptographic backdoor.

The problem is that implementations of TLS often release only 224 bits of consecutive randomness, which would make the attack 2^{32} complex, which nowadays is tractable.

During $2006, 2008, 2009$ and 2010 there were four proposals of TLS extensions that increased the amount of PRNG output [BLN15, p.18]. None of them got accepted, but RSA BSAFE library implemented "Extended Random" extension [Res08] as an option. A 2012 summary of TLS monitoring shows that one each 77000 connections request this extension.

All the TLS proposals that wanted to increase the PRNG output would have made `DUAL_EC_DRBG` easier to exploit (even if they were unrelated). They all got discard but still, we can see a non-oficial extension implemented in BSAFE that made `DUAL_EC_DRBG` easier to exploit.

4.4.2 Other Problems

Even with the possible kleptographic backdoor that `DUAL_EC_DRBG` can have, the algorithm design is poor in other aspects completely unrelated to this trapdoor. Two of the main critics were it's statistical bias and performance, there was no apparent reason to push this algorithm (beside the requested defense niche usage) as it performs worse than symmetric-based PRNGs and it's statistical bias is much higher.

Statistical Bias Berry Schoenmakers and Andrey Sidorenko formulated in a paper [SS06] how `DUAL_EC_DRBG` output bits were biased and how they could differentiate random output with the RNG output with a significant (compared to other RNGs) success rate. Based on Brown's paper [Bro06] findings that s_iQ acts as a random point in the curve, they defined $\phi(r)$ (being $r = LSB_{240}(x(s_iQ))$) as the number of points of the curve that, after the LSB operation, match to the same bit block. They found this values are higher than the expected 2^{16}, introducing a bias. If the probability of a certain output block appearing is $\frac{\phi(r)}{\#E(\mathbb{F}_p)}$ and $\#E(\mathbb{F}_p)$ is constant, the probability is tied to $\phi(r)$. They calculated the expected $\phi(r)$ and an experimental one and the latest was higher. This created a distinguisher that could predict the PRNG with success rate of 0.50078. The

process would be calculating $\phi(r)$ for the shown block and if $\phi(r) > 2^{16}$ conclude that the block was coming from `DUAL_EC_DRBG` and not a random uniform distribution.

Performance By relying on rather complex elliptic curve operations, `DUAL_EC_DRBG` had a very poor performance [Kel14, p.46]. Even if elliptic curve operations can be simple and can be computed "apparently fast" in modern devices, they might not be the best choice for embedded devices or when thousands of operations are required in short periods of time to generate large random bitstrings. There are much better solutions which even include CPU support for parts of their computations, which accelerates the proces. We've seen that research papers show it's performance to be "thousands of times slower than alternatives".

5 Conclusion

In this final section we want to share our findings and conclusions we've taken during the development of the report. Even if Random Number Generations can be seen important for certain crypto-related tasks such as key generation, we have to be careful regarding their usage on nonces too. Needless to say, non-cryptographers should never build (or implement) a RNG on their own, but instead use some good implementation of a well-known algorithm that has been put under appropriate scrutiny. As we've seen in the three shown scenarios, misused or bad/weak RNGs broke the entire security of the systems even in constrained enviroments. The **takeaway** from this report should be to put as much attention to RNGs as one puts on encryption modes, primitives and protocols, because weak RNG can have disastrous to an (otherwise) flawless cryptosystem.

References

[Ada14] Aris Adamantiadis. *DUAL_EC_DRBG Backdoor: A proof of concept.* Feb. 2014. URL: `https://blog.0xbadc0de.be/archives/155`.

[Ami23] Nils Amiet. *Polynonce: A Tale of a novel ECDSA Attack and Bitcoin Tears.* Mar. 2023. URL: `https://research.kudelskisecurity.com/2023/03/06/polynonce-a-tale-of-a-novel-ecdsa-attack-and-bitcoin-tears/` (visited on 05/22/2023).

[BK12] Elaine Barker and John Kelsey. "NIST Special Publication 800-90A Revision 1". In: *NIST Technical Series* (Jan. 2012). URL: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-90a.pdf`.

[BLN15] Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. "Dual EC: A Standarized Back Door". In: *Independent* (July 2015).

[Bro06] Daniel R. L. Brown. *Conjectured Security of the ANSI-NIST Elliptic Curve RNG.* Cryptology ePrint Archive, Paper 2006/117. `https://eprint.iacr.org/2006/117`. 2006. URL: `https://eprint.iacr.org/2006/117`.

[BV05] Daniel R. L. Brown and Scott A. Vanstone. "Elliptic Curve Random Number Generation". 60/744,982. Jan. 2005. URL: https://projectbullrun.org/dual-ec/documents/11336814.pdf.

[Che+14] Stephen Checkoway et al. "On the Practical Exploitability of Dual EC in TLS Implementations". In: *USENIX* (Aug. 2014). URL: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-checkoway.pdf.

[EDN11] EDN. *The Sony PlayStation 3 hack deciphered: what consumer-electronics designers can learn from the failure to protect a billion-dollar product ecosystem.* May 2011. URL: https://www.edn.com/the-sony-playstation-3-hack-deciphered-what-consumer-electronics-designers-can-learn-from-the-failure-to-protect-a-billion-dollar-product-ecosystem/ (visited on 05/24/2023).

[fai10] failOverflow. *Console Hacking 2010.* Dec. 2010. URL: https://fahrplan.events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf (visited on 05/24/2023).

[Gjo05] Kristian Gjosteen. "Comments on Dual-EC-DRBG/NIST SP 800-90". In: (Dec. 2005). URL: https://web.archive.org/web/20110525081912/http://www.math.ntnu.no/~kristiag/drafts/dual-ec-drbg-comments.pdf.

[Gre13a] Matthew Green. *A few more notes on NSA random number generators.* Dec. 2013. URL: https://blog.cryptographyengineering.com/2013/12/28/a-few-more-notes-on-nsa-random-number/.

[Gre13b] Matthew Green. *The Many Flaws of DUAL_EC_DRBG.* Sept. 2013. URL: https://blog.cryptographyengineering.com/2013/09/18/the-many-flaws-of-dualecdrbg/.

[Hen13] Rick Henderson. *Sony agrees settlement with George Hotz (aka Geohot) in PS3 jailbreaking case.* Apr. 2013. URL: https://www.pocket-lint.com/games/news/playstation/109593-sony-agrees-ps3-geohot-settlement/ (visited on 05/24/2023).

[Kel14] John Kelsey. *Dual EC in X9.82 and SP 800-90.* Ed. by NIST. May 2014. URL: https://csrc.nist.gov/csrc/media/projects/crypto-standards-development-process/documents/dualec_in_x982_and_sp800-90.pdf.

[Mac23] Marco Macchetti. *A Novel Related Nonce Attack for ECDSA.* Cryptology ePrint Archive, Paper 2023/305. https://eprint.iacr.org/2023/305. 2023. URL: https://eprint.iacr.org/2023/305.

[Men13] Joseph Menn. *Exclusive: Secret contract tied NSA and security industry pioneer.* Dec. 2013. URL: https://web.archive.org/web/20140102045805/https://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220.

[NIS14] NIST. *NIST Removes Cryptography Algorithm from Random Number Generator Recommendations*. Apr. 2014. URL: https://www.nist.gov/news-events/news/2014/04/nist-removes-cryptography-algorithm-random-number-generator-recommendations.

[Por13] T. Pornin. *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. RFC 6979. http://www.rfc-editor.org/rfc/rfc6979.txt. RFC Editor, Aug. 2013. URL: http://www.rfc-editor.org/rfc/rfc6979.txt.

[Res08] E. Rescorla. "Extended Random Values for TLS". In: *RFC IETF* (Apr. 2008). URL: https://datatracker.ietf.org/doc/html/draft-rescorla-tls-extended-random-00.

[Sch07] Bruce Schneier. *Did NSA Put a Secret Backdoor in New Encryption Standard?* Ed. by Wired. Nov. 2007. URL: https://web.archive.org/web/20150608034624/https://archive.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115.

[SF07] Dan Shumow and Niels Ferguson. *On the Possibility of a Back Door in the NIST SP800-90 Dual EC PRNG*. Ed. by Microsoft. Aug. 2007. URL: http://rump2007.cr.yp.to/15-shumow.pdf.

[SS06] Berry Schoenmakers and Andrey Sidorenko. *Cryptanalysis of the Dual Elliptic Curve Pseudorandom Generator*. Cryptology ePrint Archive, Paper 2006/190. https://eprint.iacr.org/2006/190. 2006. URL: https://eprint.iacr.org/2006/190.

[YY97a] Adam Young and Moti Yung. "Kleptography: Using Cryptography Against Cryptography". In: vol. 1233. May 1997, pp. 62–74. ISBN: 978-3-540-62975-7. DOI: 10.1007/3-540-69053-0_6.

[YY97b] Adam Young and Moti Yung. "The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems." In: vol. 1294. Aug. 1997, pp. 264–276. ISBN: 978-3-540-63384-6. DOI: 10.1007/BFb0052241.