University of Patras, Electrical and Computer Engineering

Analysis of the Dual_EC_DRBG algorithm

Ilias Ouzounis
up1083749

December 23, 2024

Contents

1 Random Bit Generators

In cryptography it is essential to be able to generate random values. They are the most secure from attackers as any attempts to guess them are futile. From the passwords we all use to access our accounts to keys used to encrypt data, random values are needed everywhere to create robust and secure systems.

Unfortunately, humans are notoriously bad at creating random values and introduce a certain amount of bias [1]. This is why we had to rely on machines and algorisms to achieve high entropy on the values they generate. However, machines are completely deterministic and truly random values are impossible to create. To compensate for this, pseud-random number generators (PRNGs) were created. They work by utilizing a seed as the input to a deterministic function to generate a sequence of values that appear random. This function has to be chosen carefully to ensure the generated values are as close to random as possible and impossible to predict.

We can model the PRNG as a function f that maps an input from $\mathfrak{R}$ to a finite set of values S. [2]

$$f : \mathfrak{R} \to S \tag{1.1}$$
$$\tag{1.2}$$

The seed is the input to the function and the output is the generated value. The function f should be a one-way function [3] to prevent attackers from reconstructing the seed from the generated values. Additionally, there are other metrics to evaluate the suitability of the function f as a PRNG such as the period of the generated values and the correlation between them [2]. Nonetheless, PRNGs simply move the issue of creating random values to picking a random seed as the input to f, the main problem still remains.

The method of selecting a seed is equally important as the function f itself. Nowadays, most computer systems use the inherent randomness found in nature as the seeds to their PRNGs. Examples include the time of day, system temperature, cache lookup times or even user mouse movements. These values are rich in entropy and are additionally passed through a hash function to ensure they are more evenly distributed [2]. Even temperature values that are simar to each other will be transformed into completely different seeds. However, generating seeds this way is slow and not suitable for application stat require a high amount of random values. To combat this issue, a seed is not only used to generate a random value as the output of f but is also transformed through a similar function into a new seed that is used for the next value of the sequence. Periodically, when the truly random values from nature are available, they overwrite the seed to start a new chain of pseudo-random values.

$$f : S_1 \to S_2 \tag{1.3}$$
$$g : S_1 \to S_1 \tag{1.4}$$
$$s_0 = \text{seed} \tag{1.5}$$
$$v_1 = f(s_0), s_1 = g(s_0)$$
$$v_2 = f(s_1), s_2 = g(s_1)$$
$$\vdots$$

Here, S_1 is the set of all random seeds and S_2 is the set of all generated values. s_i are the seeds used in each step and v_i are the generated values.

There are a plethora of well constructed PRNGs that are used in practice. The Linear Congruential Generator (LCG) and the Mersenne Twister are some of the most widely used PRNGs for general purposes. For cryptography though, some more specialized and more advanced algorisms have been developed. The focus of this report is on the Dual_EC_DRBG algorithm. An Elliptic Curve based algorithm that was proposed by the National Institute of Standards and

Technology (NIST) in 2006 [4]. Before analyzing the algorism and the controversy surrounding it, we will first examine the the theory behind Elliptic Curves and how they are used in cryptography.

2 Elliptic Curves

2.1 Definition

Elliptic curves are a fascinating family of algebraic curves. They are defined by a simple equation (2.1) but have a very complex structure and, unpredictably, a lot of applications in number theory.

$$y^2 = x^3 + \alpha x + \beta \tag{2.1}$$

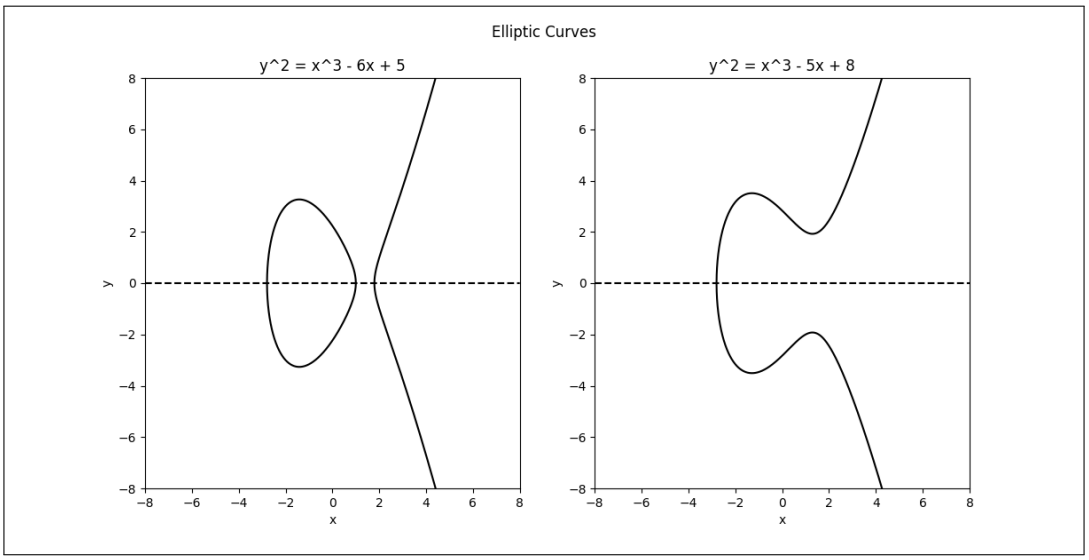

Figure 2.1: The two forms of an elliptic curve.

Figure 2.1 shows the two forms of an elliptic curve. It is important to note the lateral symmetry of the curves over the x-axis. Furthermore, depending on the values of α and β, the curve can have either one or two components. It is dependant on the discriminant of the curve calculated as $\Delta = -16(4\alpha^3 + 27\beta^2)$. If $\Delta < 0$, the curve has two disconnected components and when $\Delta > 0$, the curve is one continuous line. When $\Delta = 0$, the elliptic curve is called degenerate and it loses many of its useful properties. We won't be focusing on elliptic curves with $\Delta = 0$.

2.2 Point Addition

We can define an operation on the points of an elliptic curve called point addition. Given two different points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ on the curve we can define $R = P + Q$, another point on the curve as follows:

Draw the line that passes through P and Q. This line will intersect the curve another point $-R = (x_R, -y_R)$. We take $R = (x_R, y_R)$ as the reflection of $-R$ over the x-axis.

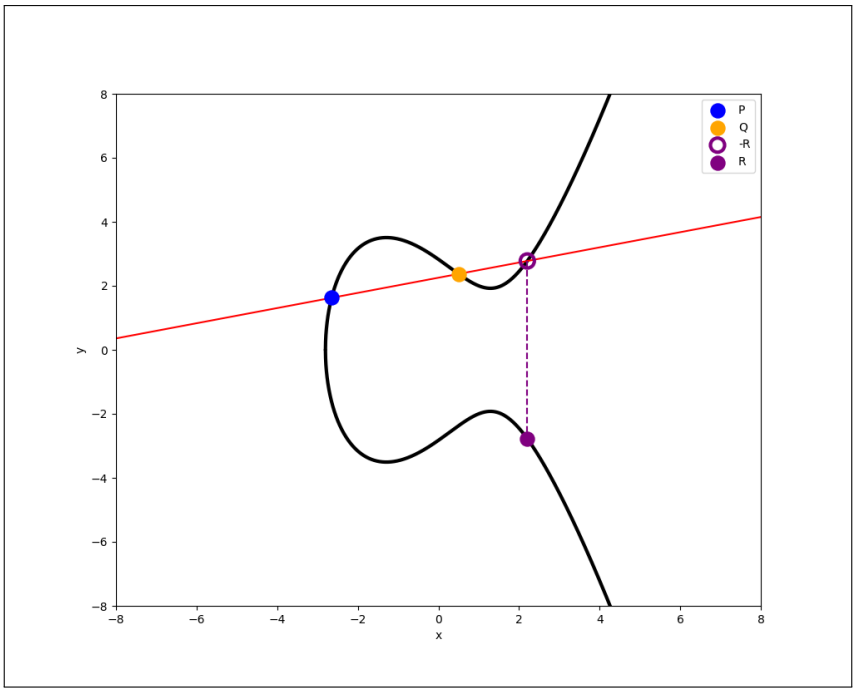

Figure 2.2: Point addition on an elliptic curve.

More formally, the slope of the line passing through P and Q is:

$$m = \frac{y_Q - y_P}{x_Q - x_P} \tag{2.2}$$

Then the third point on the line that intersects the elliptic curve has coordinates [5]:

$$x_R = m^Q - x_P - x_Q \tag{2.3}$$

$$-y_R = m(x_P - x_R) - y_P \tag{2.4}$$

Finally, we reflect that point over the x-axis to get $R = P + Q$.

$$y_R = -(-y_R) \tag{2.5}$$

If $Q = -P$, then we can't calculate m as $x_Q = x_P$ and the line that passes through them is parallel to the y-axis and does not intersect the curve at any other point. In this case, we define $P + (-P) = \mathcal{O}$, the point at infinity.

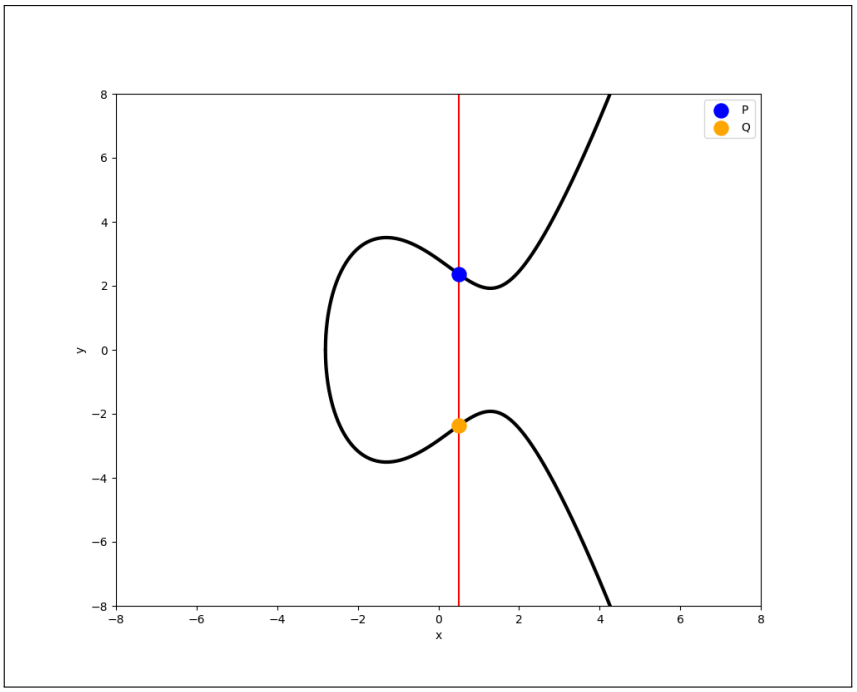

Figure 2.3: Point at infinity.

2.3 Point Doubling

If $P = Q$, we can't uniquely define a line that passes through a single point. Instead we take the tangent line at P and find the R as the reflection of the intersection of the tangent line with the curve over the x-axis in a similar way as before.

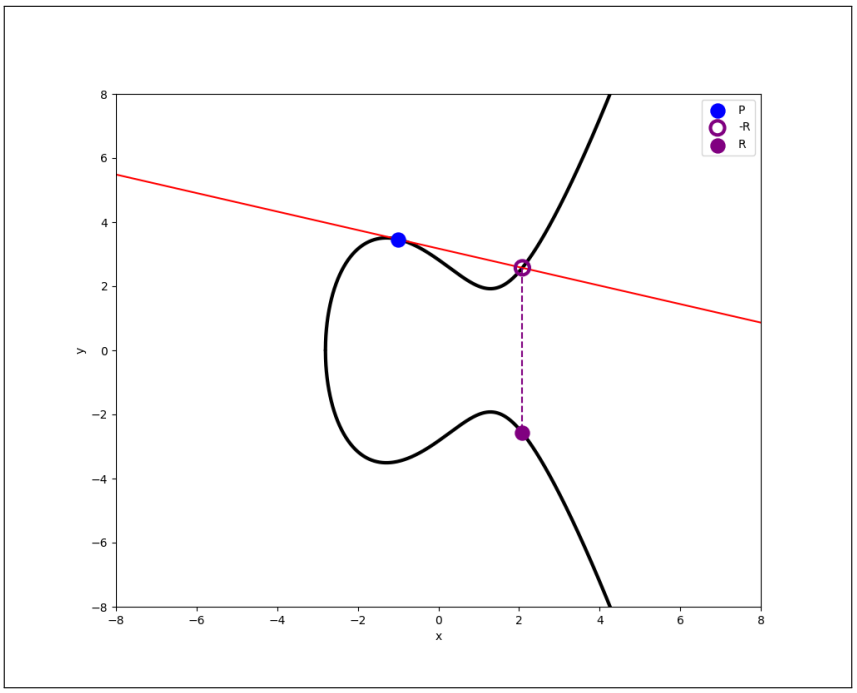

Figure 2.4: Point doubling on an elliptic curve.

The slope of the tangent line at P is [5]:

$$m = \frac{3x_P^2 + \alpha}{2y_P} \tag{2.6}$$

The coordinates of the third point on the line that intersects the curve are:

$$x_R = m^2 - 2x_P \tag{2.7}$$

$$-y_R = m(x_P - x_R) - y_P \tag{2.8}$$

And finally, we reflect that point over the x-axis to get $R = P + P$.

$$y_R = -(-y_R) \tag{2.9}$$

From these two definitions we can now define multiplication of a point by a scalar k as a series of point additions and doublings

$$kP = P + P + ... + P \tag{2.10}$$

2.4 Elliptic Curves over Finite Fields

Elliptic curves are normally defined over the real numbers but the true power of the properties of the elliptic curves is revealed when we examine them under the lens of a finite field $\mathbb{F}_p$.

A finite field $\mathbb{F}_p$ is a set of integers under addition and multiplication modulo p where p is a prime number. This field, along with the points operations defined in the previous section, form a group $E(\mathbb{F}_p)$. The members of this group are all the points with integer coordinates that satisfy the elliptic curve equation under the operations of the field $\mathbb{F}_p$.

$$y^2 \equiv x^3 + \alpha x + \beta \mod p, \quad x, y \in \mathbb{F}_p \tag{2.11}$$

As a group, $E(\mathbb{F}_p)$ needs to have an identity element, $P + e = P$ for all $P \in E(\mathbb{F}_p)$. We take the point at infinity $\mathcal{O}$ as the identity element of the group. This way we can also define the inverse of a point P as $P + (-P) = \mathcal{O}$.

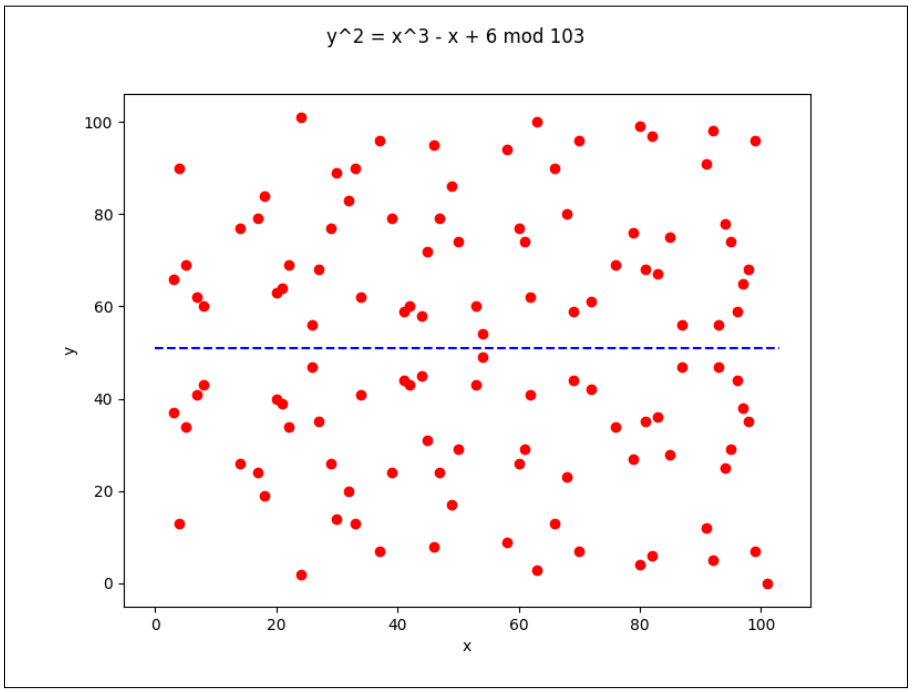

Figure 2.5: Points of $E(\mathbb{F}_p)$ with $p = 103$.

As we can see from Figure 2.5, even though the points are distributed in a seemingly random way, we still have a symmetrical structure, now over the line $y = \frac{p-1}{2}$.

2.5 Discrete Logarithm Problem on Elliptic Curves

With an elliptic curve over a finite field defined, we can now understand how elliptic curves are used in cryptography. The basis for the usage of elliptic curves is that given a point Q, it is really hard to find the scalar k such that $Q = kP$, where P is a publicly known point of the curve. As shown in Figure 2.6, the changes in kP as k varies are sporadic and unpredictable.

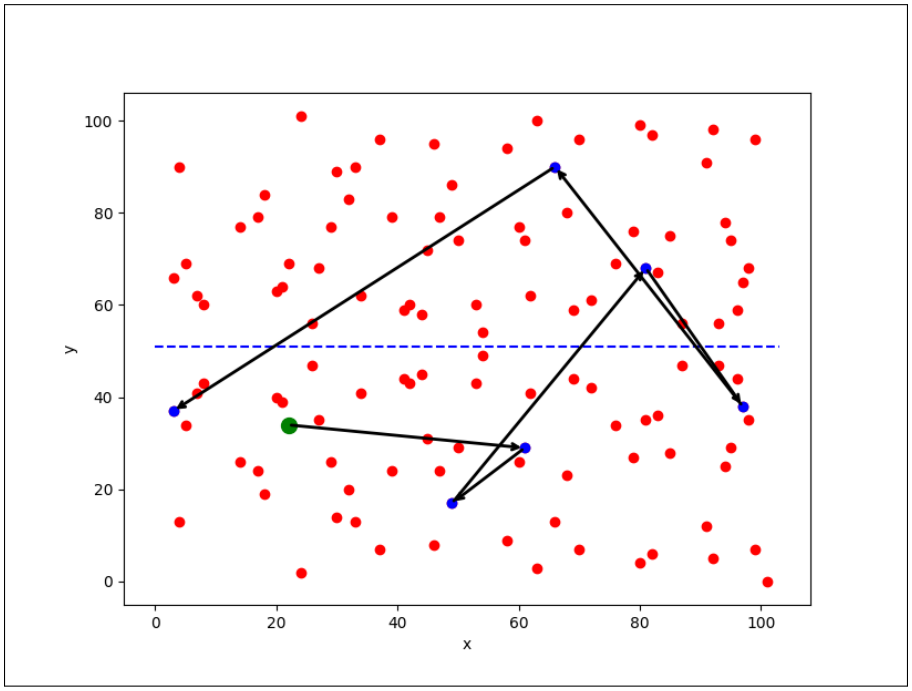

Figure 2.6: Points for increasing values of k. Point P is shown in green

This is called the discrete logarithm problem on elliptic curves and is what makes elliptic curves a good candidate for systems where key security is paramount, such as in bitcoin [6]. The elliptic curve, the field $\mathbb{F}_p$ and the generator point P are all publicly known and the key is the scalar k that is kept secret.

The best known algorithms for solving the discrete logarithm problem on elliptic curves are all $O(\sqrt{n})$ where n is the order of the cyclic group $\{P, 2P, ...nP\}$ where $nP = \mathcal{O}$. The elliptic curve and P are usually selected in a way that $n \approx p$. For the large primes, on the order of 2^{256}, this is a computationally infeasible problem to solve. Because the best known algorithms are only $O(\sqrt{n})$, elliptic curves can be as secure as other cryptographic systems such as RSA, but with much smaller key sizes. In fact, a 256-bit elliptic curve key is considered to be as secure as a 3072-bit RSA key [7].

With all these tools at our disposal, we can now move on to the analysis of the Dual EC DRBG algorithm.

3 Dual EC DRBG

3.1 The Dual EC DRBG algorithm

The Dual EC DRBG algorithm was a pseudo-random bit generator that was included in the NIST SP 800-90 standard [4]. It utilizes elliptic curve cryptography to generate the random bits as well as updating the seed of the generator. This algorithm was designed in part by the National Security Agency (NSA) of the United States as a secure PRNG. It is built on an elliptic curve over a finite field, with parameters $\alpha, \beta, \mathbb{F}_p$ selected by the designers. The size of the resulting

elliptic curve group is massive, in the order of 2^{256} for the smallest of the three curves proposed.

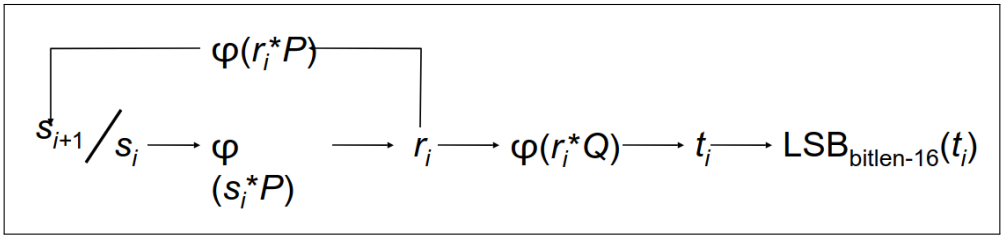

Figure 3.1: Diagram of the Dual EC DRBG algorithm

In Figure 3.1 the basic structure of the Dual EC DRBG algorithm is shown. The algorithm consists can be split in two parts, the generating and the reseeding part.

Starting from a seed s_i, the algorithm performs a scalar-Point multiplication with a specified point P, $s_i P$. Afterwards, the x coordinate of the resulting point is kept while the y coordinate is discarded,

$$r_i = (s_i P)_x \tag{3.1}$$

Afterwards, this new scalar value is used in another scalar-Point multiplication with another selected point on the curve and the x coordinate is kept again.

$$A = r_i Q \tag{3.2}$$
$$t_i = A_x \tag{3.3}$$

Finally, form the resulting x coordinate, the output bits are generated by removing the 16 most significant bits of t_i.

$$o_i = t_i \mod 2^{16} \tag{3.4}$$

As for the new seed, it is calculated by performing a third scalar-Point multiplication and keeping the x coordinate, between r_i and P.

$$s_{i+1} = (r_i P)_x. \tag{3.5}$$

For the values of the parameters of the elliptic curve used in the algorithm, the NIST SP 800-90 specifies them as follows for the curve P-256 [4]:

$$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$$
$$\alpha = -3$$
$$\beta = 5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b_{16}$$

This elliptic curve creates a group of order

$$n = 115792089210356248762697446949407573529996955224135760342422259061068512044369$$

well within the required size to be secure to brute force attacks.

As for the points P and Q used in the algorithm, the NIST specifies them as [4]:

$$P_x = 6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296_{16}$$
$$P_y = 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5_{16}$$
$$Q_x = c97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c59852018192_{16}$$
$$Q_y = b28ef557ba31dfcbdd21ac46e2a91e3c304f44cb87058ada2cb815151e610046_{16}$$

3.2 Controversy regarding the Dual EC DRBG algorithm

In theory the algorithm is effective in providing secure and random bits at the output as the parameters are sufficiently large and create prime-sized groups that are safe from some other known methods that exploit properties of elliptic curves [8]. However, when it was first announced, there were some concerns over the selection of the P and Q points [9]. It is customary for algorithm designers to provide some explanation of their derivation of the values used, but none were provided for the Dual EC DRBG algorithm. They are often generated by a verifiable seed or by some other method involving mathematical constants, like the digits of pi. This created some suspicion that the NSA had maliciously selected the points to create a backdoor in the algorithm. This was further fueled by the fact that the NSA had previously been caught inserting backdoors in other cryptographic algorithms, like the DES encryption standard, and that they required the use of these specific points. Any other points were deemed insecure by the NSA, but no explanation was given as to why.

Not long after the algorithm was announced, the cryptographic community voiced their concerns over the algorithm and the NSA's involvement in its design. Firstly, it was criticized for being slower than other know algorithms at the time [9], but the majority of the criticism was targeted on the lack of transparency in the choices of P and Q. As many cryptographic experts pointed out [10] [11], by selecting P and Q in a specific way, the NSA could have access to the current random seed of the generator just by looking at the outputs. This would allow them to predict the output of the generator and thus break the security of the algorithm.

3.3 The Dual EC DRBG backdoor

How could someone select P and Q in such a way that it would be possible to predict the value of the new seed s_{i+1} from the output o_i? The answer lies in the generator point P. As the order of P is prime, and equal to n (3.1), it must mean that the group generated by P is the entire group of the curve. This means that for any point Q on the curve, there exists a scalar k such that $Q = kP$. Calculating the value of k just by knowing Q and P is not realistic as it would require to solve the discrete logarithm problem, which, as has been stated before, is computationally infeasible. If the NSA chose Q by knowing the value of k, it would be possible, almost trivial, to find an e such that $P = eQ$ [12].

$$\begin{aligned} Q &= kP \\ Q &= k(eQ) \\ 1 &= ke \\ e &= k^{-1} \mod n \end{aligned} \tag{3.6}$$

With this value of e, an attacker could reverse the algorithm and retrieve the value of the seed s_{i+1} from o_i by skipping the intermediate value r_i which is computationally impossible to find.

$$\begin{aligned} s_{i+1} &= (r_i P)_x \\ s_{i+1} &= (r_i(eQ))_x \\ s_{i+1} &= (e(r_i Q))_x \\ s_{i+1} &= (eA)_x \end{aligned} \tag{3.7}$$

Recall however, that the final step before the output is truncating the first 16 bits of t_i, meaning the attacker, by knowing the output o_i cannot be certain of the value of t_i and thus the next seed s_{i+1}. Nonetheless, the size of the possible values is relatively small, $|S| = 2^{16}$, and thus reasonable to brute force. Add in the fact that we know that t_i is the x coordinate of a point on the elliptic curve and the size of S is reduced massively, to only a handful of possible points. This was also pointed out at that time [12] and suggestions were made to increase the truncated bits to 128. This would make the algorithm less efficient, as it would generate fewer random bits per iteration, but even an attacker with knowledge of the backdoor key k would struggle massively to find the correct seed s_{i+1} through all the possible values of t_i. The NSA never implemented this change though.

3.4 Adoption and removal of the Dual EC DRBG algorithm

Despite the many criticisms from the community, for some reason, the Dual EC DRBG algorithm was adopted by some corporations like RSA Security and Microsoft. RSA Security even made it the default PRNG in their BSAFE cryptographic library. This was a very controversial move as the algorithm was not only slower than other known algorithms, but had been known to be potentially vulnerable [13].

Years later, in 2014 when Snowden leaked many classified documents from the NSA, it was revealed that the NSA most likely had knowledge of the backdoor in the Dual EC DRBG and intentionally placed it there. To push for its adoption, it was even leaked that they paid RSA Security 10 million dollars to make it the default PRNG in their BSAFE library. This revelation caused a massive uproar in the cryptographic community and many companies immediately removed the algorithm from their products. The NIST also removed the algorithm from the SP 800-90 standard and advised against its use. The NSA has never confirmed nor denied the existence of the backdoor, but the evidence is overwhelming that it was intentional.

After these events, the NSA had completely lost the trust of the scientific community and even now, any attempts to create new cryptographic standards are met with skepticism and scrutiny.

4 Custom Dual EC DRBG Backdoor

4.1 Custom Backdoor

To showcase the severity of the backdoor analyzed in the previous sections, we can alter the parameters of the Dual EC DRBG algorithm to insert our own backdoor. By selecting a specific k, we can construct our own $Q = kP$ point and utilize it as our backdoor.

For example, we select $k = 1234567890abcdef_{16}$. Then $Q = kP$ is set to:

$$Q_x = 9fad84aeae08bbef7f010014d82cef6a09de2b0cf871b5ce0c4f1d13a59a5934_{16}$$
$$Q_y = 7cb45769f1070e2c2470fe5b1bfe63133c0b0cdc64ea4bf3791a8ec2a07fd4f_{16}$$

Additionally, we need to compute e such that $P = eQ$ to be able to reverse the process and find the seed s_{i+1} from an output.

$$e = k^{-1} \mod n$$
$$e = cadfd3dfd056ea711348f294745e8a9644e7e4d7d3d90b744781eac483b929fe_{16}$$

With this "new" Dual EC DRBG algorithm, we can generate a random value and recreate it using the backdoor.

For example, let's seed the generator with $s_0 = 42_{16}$. The output o_0 is:

$$t_0 = 7a8ac253a35c39494fff582e706fb8f815c7571e183c068f99449c6c2291_{16}$$

and the new seed s_1 is:

$$s_1 = b819dc3f7ac644a4edadaa7e65e520bcbe8f494d29e0fbfb5b96ebf10baf2a28_{16}$$

but we will consider it hidden to an outside attacker. The attacker has only access to the output bits o_0 and has knowledge of the parameters of the elliptic curve as well as the backdoor Q and e.

By having the truncated bits of the output, the attacker can try all possible values of $t_0 = x_i|o_0$ to try and fill the truncated bits. x_i takes all the possible values from 0 to $2^{16} - 1$ and is considered the truncated bits of the output. There are 2^{16} total values to be checked. However, the attacker knows that the output is the x coordinate of a point on the elliptic curve. This means

that if $x_i|o_0$ does not correspond to the x coordinate of an elliptic curve point, it can be discarded. This halves the number of values needed to check from 2^{16} to $32722 \approx 2^{15}$ in this example. From these candidate points, A_i, the possible seeds are then calculated as $s_{i+1} = (eA_i).x$. The attacker is left with a small number of possible seeds that can be utilized to predict candidate future outputs.

In fact, if the attacker also has access to the next output o_1, they can use the candidate seeds from o_0 to try and generate o_1' themselves. If the generated o_1' matches the actual o_1, they can narrow the possible seeds even further. For 16 truncated bits of output, it is actually enough to narrow the possible seeds down to just one. This means that the attacker can predict all future outputs of the generator.

The argument to increase the truncated bits, stems from the computational increase that is introduced with each new bit cut off. Trying different truncation sizes and plotting the time needed to find all the possible seeds in a logarithmic scale, we can clearly see that the complexity increases exponentially as shown in Figure 4.1. This makes sense since removing one bit from the output increases the number of possible values to check by a factor of 2.

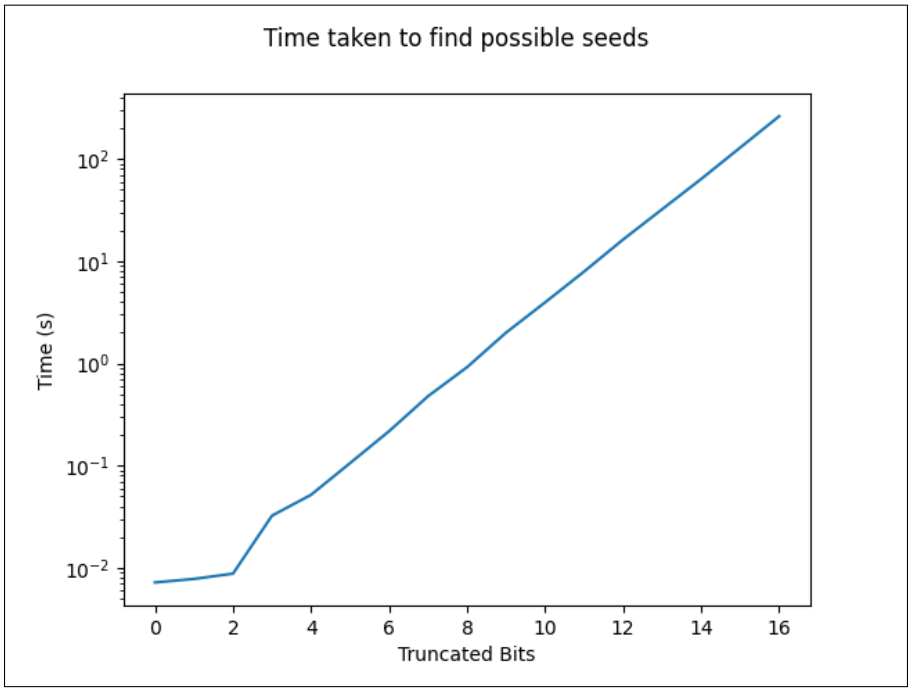

Figure 4.1: Time taken to find all possible seeds for different truncation sizes

Just 3 minutes are enough for a standard commercial laptop to find all the candidate seeds. For a larger truncation, 128 bits for example as has been proposed in [12], the attacker would need to check $2^{127} \approx 10^{38}$ seed values which would take an enormous amount of time and would make the backdoor practically useless. By increasing the percentage of bits truncated, it also harder to pinpoint the exact seed, even if the attacker knows the next output.

To confirm this, we construct a smaller elliptic curve over a field with $p = 10^9 + 7$, to make the computations faster and be able to truncate and test a larger percentage of bits. All the numbers in the field $\mathbb{F}_p$ are smaller than p and as such can be represented with 30 bits. This is the maximum size of the output.

Additionally, to define the elliptic curve, we set $\alpha = -3$ and $\beta = 123456789$. The order of the curve is $n = 1000008295$, calculated using sagemath [14] and the selected generator point $P = (31415926, 725920279)$ has the same order. The backdoor is set with $k = 42 * 69 * 111$, $e = k^{-1} = 99451207$ and $Q = kP = (657403560, 363572734)$. As the truncated bits increase,

we can see the same exponential time complexity in Figure 4.2. The number of possible seeds even if the attacker knows the next output also increases exponentially after a certain point as shown in Figure 4.3. This is the reason the backdoor is considered secure for a large enough truncation size.

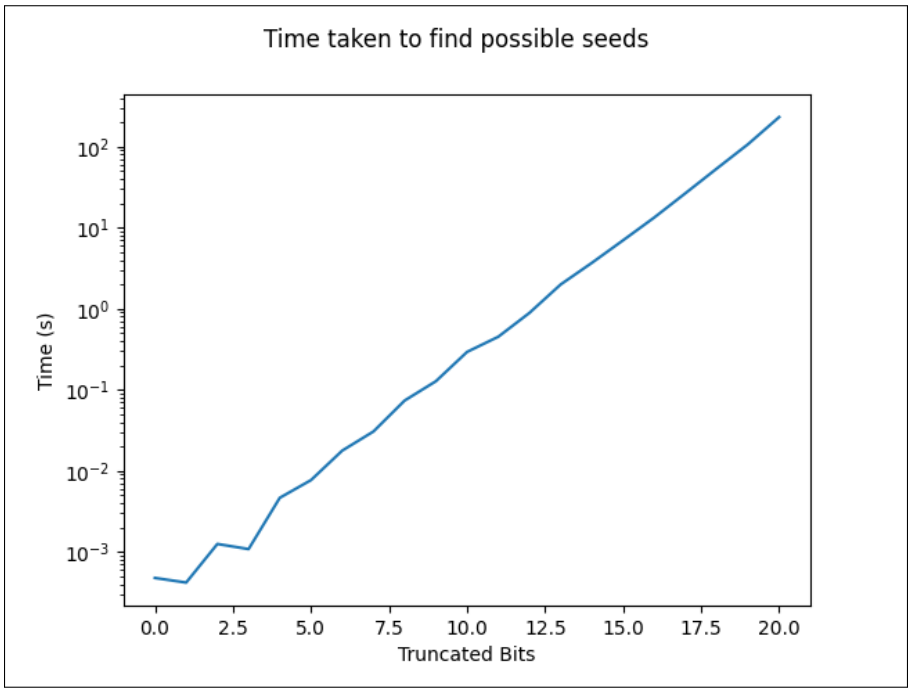

Figure 4.2: Time taken to find all possible seeds for different truncation sizes on a smaller elliptic curve

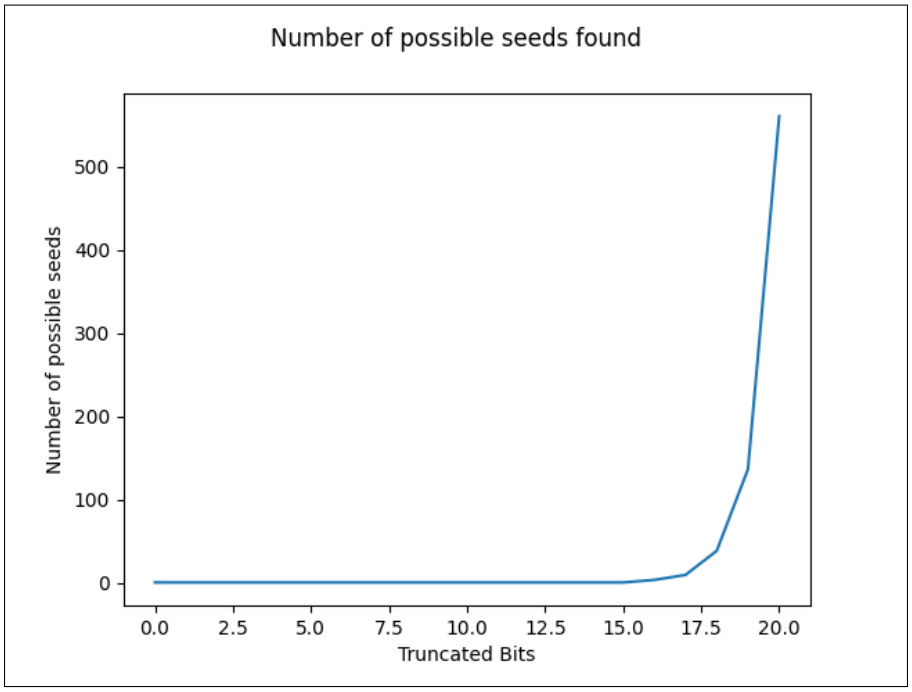

Figure 4.3: Number of possible seeds for different truncation sizes on a smaller elliptic curve

5 Conclusion

In this report, we examined the need for random number generators and looked at the theory behind one of the most infamous ones, the Dual EC DRBG algorithm. Dual EC DRBG was initially designed to provide high level security to the randomly outputted values. However, it came under huge scrutiny once a potential backdoor was first theorized. The backdoor would allow attackers to find the seed of the algorithm and predict future outputs. Something that is considered catastrophic for a random number generator. As shown in this report, the backdoor was very efficient and despite some suggestions from the community to nullify it, no changes were made and it was endorsed by NIST as a standard. Snowden's leaks in 2014 solidified the theory of the backdoor and the algorithm was finally removed from the NIST standards. Nonetheless, the algorithm is still used in many systems and some companies have yet to transition to a more secure alternative.

The reputation of NIST and the NSA were tarnished from this incident and the trust in the security of the systems they endorse was shaken. The community has since been more cautious and has been more critical of the algorithms that are proposed. The Dual EC DRBG algorithm is a prime example of how a backdoor can be inserted into a system and one is left to wonder how many other algorithms have similar vulnerabilities. The need for transparency and open source code is more important than ever and the governing bodies should be more communicative of their decisions and the reasons behind them. Any suspicious attempts have to be questioned by the community to ensure a secure and private society.

References

[1] Willem A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. Psychological Bulletin, 77(1):65–72, 1972.

[2] Tom Kennedy. Math 577-002 - monte carlo methods, 2016. Available at https://math.arizona.edu/~tgk/mc/book_chap3.pdf.

[3] Wikipedia contributors. One-way function. https://en.wikipedia.org/wiki/One-way_function, 2024. Accessed: 2024-12-20.

[4] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. Technical report, National Institute of Standards and Technology, 2007.

[5] Enigbe. About elliptic curves and dlp. https://enigbe.medium.com/about-elliptic-curves-and-dlp-ed76c5e27497#:~:text=Formally%2C%20the%20elliptic%20curve%20discrete,with%20e%2C%20i.e.%20eG%2C%20produces, 2024. Accessed: 2024-12-20.

[6] Jun Wang. Ecdsa and bitcoin ii, 2018. Accessed: 2024-12-20.

[7] Wikipedia contributors. Elliptic-curve cryptography. https://en.wikipedia.org/wiki/Elliptic-curve_cryptography, 2024. Accessed: 2024-12-20.

[8] Daniel R. L. Brown and Kristian Gjøsteen. A security analysis of the nist sp 800-90 elliptic curve random number generator. 2007.

[9] Matthew Green. The many flaws of dualECDRBG. https://web.archive.org/web/20160820174502/http://blog.cryptographyengineering.com/2013/09/the-many-flaws-of-dualecdrbg.html, 2013. Accessed: 2024-12-20.

[10] Kristian Gjøsteen. Comments on dual-ec-drbg/nist sp 800-90, draft december 2005. 04 2006.

[11] Bruce Shneier. Did nsa put a secret backdoor in new encryption standard?? https://web.archive.org/web/20140621062515/http://archive.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115, 2007. Accessed: 2024-12-20.

[12] Daniel Shumow and Niels Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng. https://web.archive.org/web/20160818212201/http://rump2007.cr.yp.to/15-shumow.pdf, 2007. Accessed: 2024-12-20.

[13] Wikipedia contributors. Dual EC DRBG. https://en.wikipedia.org/wiki/Dual_EC_DRBG, 2024. Accessed: 2024-12-20.

[14] The Sage Developers. Sagemath, the sage mathematics software system, 2024. http://www.sagemath.org.