Elliptic Curve Cryptography and Government Backdoors

Ben Schwennesen

Duke University

Math 89S (Mathematics of the Universe)

Professor Hubert Bray

April 24, 2016

Introduction

For as long as humans have roamed the Earth, they have kept secrets. Further still, as long as secrets have been withheld, there have been people attempting to expose them. Continual advancements in technology have had complicated implications for the ability of people to keep such secrets hidden. For example, while keeping one's secrets on a piece of paper might sound unwise, doing so would almost certainly be more secure than keeping them in an unprotected text document (to steal your piece of paper, an attacker would have to physically approach you, while a sophisticated adversary could easily hack you from the other side of the Earth). In other words, while technology has allowed for more secure means of storing data, finding a way to access that data without permission has become simpler, for largely the same reasons.

The reasons why governments have a vested interest in cryptography—broadly speaking, the study and implementation of techniques in math and computer science which allow for secure storage and/or transmission of data—are not difficult to see. Recent news has been saturated by reports of the public debate raging between the FBI and Apple, with the former arguing that it should be possible to issue warrants dictating that the latter unlock the encrypted iPhones of felony suspects [14]. For a while the FBI even pushed for a "golden key" for enforcement agencies like the NSA and FBI, which could be used to unlock any iPhone, but could only ever be utilized for government purposes. Setting aside whether a key of this sort is even possible, should citizens and corporations like Apple even trust the government with such a powerful tool?

There are compelling reasons why one would be hesitant to bestow U.S. law enforcement agencies with a "golden key." One of these reasons is the main topic of this paper; in 2013, (in)famous whistleblower Edward Snowden leaked a large collection of Top Secret documents and memos from the NSA (National Security Agency), which he had access to as a federal contractor working primarily within the CIA [10]. The leaked documents cover a vast array of projects, but most crucially for our purposes, (according to the New York Times [10]) they reveal in quite "striking detail" how the NSA ensured that it would be able to read data collected in its mass surveillance programs (about which the Snowden leak also revealed a great deal of information). In short, this was done by releasing an encryption standard through the National Institute of Standards and Technology (NIST), deemed the NIST SP 800-90A. Among the algorithms included in the standard was the "Dual_EC_DRBG" (Dual Elliptic Curve Deterministic Random Bit Generator); as we will explore, the implementation of this algorithm had made professional cryptographers suspicious for years leading up to the Snowden leak [19]. Among the documents Snowden released were memos that purportedly confirm the existence and use of a backdoor—a subtle way to easily break through the encryption—by the NSA. Crypto experts note, however, that the *Times* has not released these memos, and direct quotes from the leaked documents do not directly mention a backdoor, only imply its use. Regardless of the evidence provided by the *Times*, cryptographers like Bruce Schneier and Paul Kocher argue, the way Dual EC was implemented was such poor cryptography that "none of it makes sense if there isn't a backdoor in [it]" [19].

Groups, Rings, and Fields

[1] In order to understand how elliptic curve cryptography works (and in-turn how the NSA allegedly exploited it to create a backdoor), we should first briefly delve into the mathematics of groups, rings, and fields. These objects are, generally speaking, algebraically structured sets that are equipped with specific operations. We note prior to providing precise definitions of these objects that—generally speaking—all fields are rings, and all rings are, in-turn, groups; that is, if we let $\mathcal{S}$ denote some set [9],

$$\mathcal{S} \text{ is a field} \implies \mathcal{S} \text{ is a ring} \implies \mathcal{S} \text{ is a group.}$$

[1]Note that these concepts have been somewhat simplified, for the audience's sake as well as the author's.

Implication in the reverse direction, however, is not generally valid:

$$\mathcal{S} \text{ is a group } \not\Longrightarrow \mathcal{S} \text{ is a ring } \not\Longrightarrow \mathcal{S} \text{ is a field.}$$

Definition 0. A *binary operation* is a function $* : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, where $\mathcal{S} \times \mathcal{S}$ denotes the cartesian product of set $\mathcal{S}$ with itself (often simply written as $\mathcal{S}^2$). For example if $\mathcal{S} = \{a, b, c\}$, where $a, b, c \ \in \mathbb{R}$, then

$$\mathcal{S}^2 = S \times S = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}.$$

So, a binary operation takes a pair of elements from the set on which it is performed and produces a new object [12, p. 19].

Definition 1. A *group* is an underlying set of elements together with a binary operation (called the *group law*, often simply addition) such that if the operation is performed on any two elements of the set, the result is always contained in the set. This property is deemed *closure* under the operation, and can be described as follows: let $\mathcal{G}$ denote the set underlying a group with operation $*$, then [12]

$$\alpha, \beta \ \in \ \mathcal{G} \implies \alpha * \beta \ \in \ \mathcal{G}.$$

In addition to closure, the following conditions must be satisfied for an object to qualify as a group [9]:

1. *Associativity:* $(a*b)*c = a*(b*c)$, for all $a, b, c \in \mathcal{G}$ (note: not necessarily commutative).
2. *Identity:* There exists element $1 \in \mathcal{G}$, such that $1 * a = a * 1 = a$ for all $a \in \mathcal{G}$.
3. *Inverse:* Given any element $a \in \mathcal{G}$, there exists $b \in \mathcal{G}$ such that $a * b = b * a = 1$.

Note that one may also define a *subgroup* to be a nonempty subset of the group which still satisfies the above properties [12, p. 32].

Definition 2. The simplest way to describe a *ring*—having already defined a group—is to say that a ring is a group with an additional binary operation. Usually, the group starts with an operation deemed addition $\oplus$ and an operator deemed multiplication $\odot$ is added; however, this may be reversed, or the operators could be entirely distinct from standard addition and multiplication [9].

An easily recognized example of a ring would be the set of integers $\mathbb{Z}$, with only the standard addition and multiplication operations.

Definition 3. Finally, proceeding from a ring to a field requires the *multiplicative inverse* property. That is, letting F denote a given field, if $a \in F$ and $a \neq 0$, then there exists $b \in F$ such that

$$a \odot b = b \odot a = 1.$$

Thus, equivalently, a field is effectively a ring endued with a division operation [9]. In more formal settings, a field is defined as a "commutative ring in which all nonzero elements are inveritble" [5].

The most familiar field to the vast majority of people is the field of real numbers, $\mathbb{R}$, under the typically defined addition and multiplication (and division) operations. The real numbers, along with the rational numbers $\mathbb{Q}$, are *infinite* fields (in the case of the reals, *uncountably* infinite), because the number of elements they contain are unbounded [5]. To understand the kind of fields essential to elliptic curve cryptography—finite fields—familiarity with modular arithmetic is indispensable.

Digression on modular Arithmetic

An everyday example of modular arithmetic (which has become somewhat trite) is the way hours are counted on a clock. An hour past midnight is not called 13:00 (nor 25:00 if one prefers military time), because clocks operate with the modulus 12 (24). In general applications, a modulus may be any positive integer. In the clock example, we would say that 13 o'clock and 1 o'clock are congruent modulo twelve.

Definition 4. Given integers a and b and a positive integer μ called the *modulus*,

$$a \equiv b \pmod{\mu} \iff a - b = \mu k \text{ for an integer } k,$$

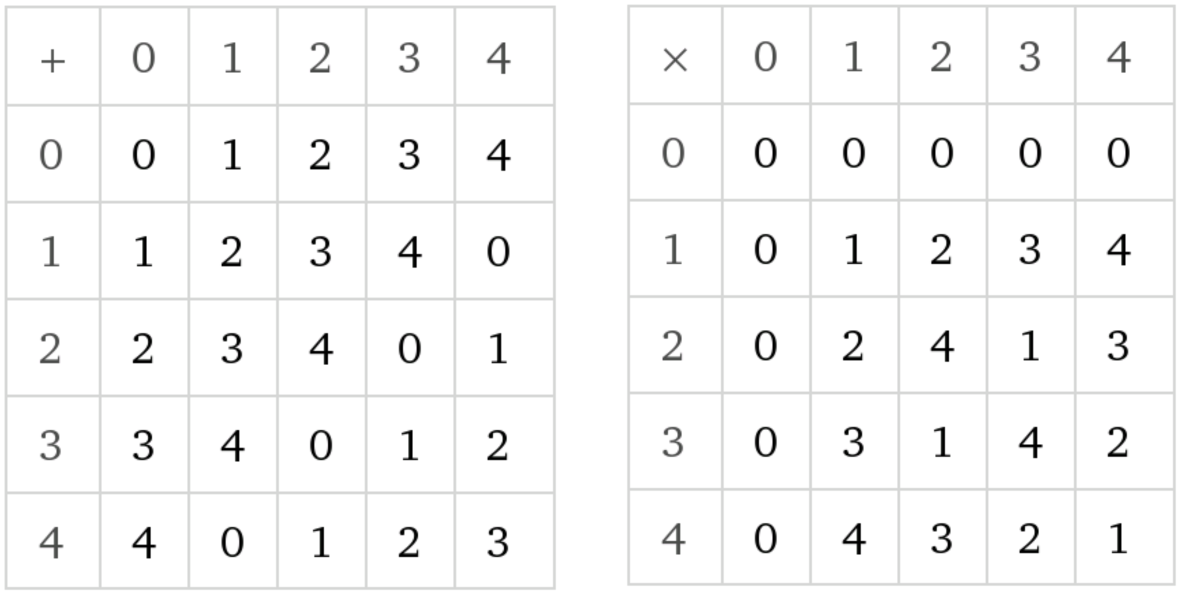
in which case a and b are said to be *congruent* modulo μ. Such an equation will normally have many solutions since it describes an equivalence, as opposed to an equality [8]. For example,

$$z \equiv 13 \pmod 9$$

has solutions $z = \ldots, -14, -5, 4, 13, 22, \ldots$. The notation MOD (or % in computer programming) is often used to indicate the smallest positive integer solution to such equivalences; in our example, 13 (MOD 9) $= 4$ is this solution, which should make it clear that this number really represents the *remainder* from integer division as taught in primary schools [8].

While division is technically not a valid operation in modular arithmetic, the *multiplicative inverse* is defined and is largely equivalent to division. Specifically, if we want to find the inverse of α modulo p, we seeks β such that $\alpha\beta \equiv 1 \pmod p \iff \alpha^{-1} = \beta$ (MOD p). For example, the modular inverse of 16 modulo 37 is 7 since $16 * 7 \equiv 1 \pmod{37}$ since $16 * 7 = 112 = 111 + 1 = 3(37) + 1$. Furthermore, a number only has a modular inverse if it is coprime to the modulus, i.e., the greatest common divisor of modulus p and the number for which we seek an inverse must be one [8]. Hence why prime p makes modular arithmetic more powerful. Now having a basic understanding of modular arithmetic, we are equipped to study the fields on which elliptic curves have cryptographic utility.

Definition 5. A *finite field* (also called a *Galois field*, after French mathematician Evariste Galois) is simply a field containing a finite number of elements. The number of elements in the field is deemed its *order*; for any prime number p, an integer ring modulo p (i.e., mod p is applied to all calculations performed on the field's elements) qualifies as a finite field with order p. (Pause to reflect on why this makes sense: if we are only interested in the



+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

(a) Addition table, $GF(5)$

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

(b) Multiplication table, $GF(5)$

Figure 1: Arithmetic in a finite field of order 5.

remainder of our calculations after division by p, it makes sense that all calculations can yield only values in $\{0, 1, 2, \ldots, p-1\}$). The only composite numbers on which a finite field may be defined are powers of prime numbers (i.e., we can define a finite field on $2^n, 3^n, 5^n$, etc. for any integer n, but not on $6, 10, 12, 14, 15$, etc.). The *characteristic* of a field has a precise definition that is not necessary here, but note that it must always be either zero (infinite fields) or a prime number (p, including when p is raised to a power). There are a number of ways to denote a finite field (perhaps *too many*), most commonly $GF(p)$, $\mathbb{Z}_p$, $\mathbb{F}_p$, or $\mathbb{Z}/p$, where p denotes the modulus; here, we will use the notation $GF(p)$ [5].

Elliptic Curves

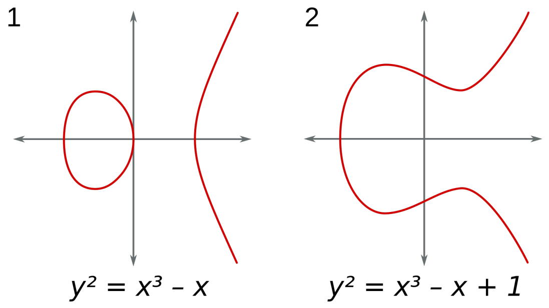
An elliptic curve [2] may be (informally) defined as the set of points satisfying an equation of two variables, where one of the variables has degree three and the other has degree two (a

[2]Despite the similarity in their names, ellipses and elliptic curves bear only a distant relationship.

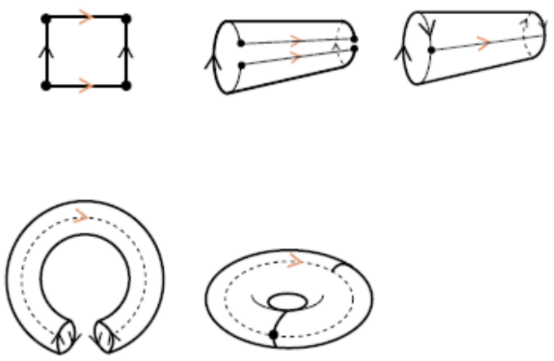
formal definition requires deep knowledge of complex analysis, topology, algebraic geometry, and other upper-level fields of mathematics). For our purposes, these equations will be of the form

$$y^2 \equiv x^3 + Ax + B \pmod{p},$$

where A and B are scalars in the field $GF(p)$. This is a version of the so-called *Weierstrass*



(a) Elliptic curves over $\mathbb{R}^2$.



(b) Topologically, an EC over $\mathbb{C}^2$ is a torus.

Figure 2: Plots of elliptic curves over common fields.

normal form [4]. Note that the equivalence symbol $\equiv$ is often (but not always) used in place of the traditional equals sign, to ensure that it is understood the relation only holds due to the modulus p. Further note that there is an additional requirement on the scalars in order for the mathematics of the curve to be useful in cryptography: the *discriminant*, $\Delta \equiv -16(4A^3 + 27B^2)$ must not be equivalent to zero (the curves when this holds are called *nonsingular*, since $\Delta = 0$ results in a singularity on the curve) [13]. Figure 3 demonstrates why this is the case. With this additional requirement, as well as a "point at infinity", $\mathcal{O}$ (necessary for reasons that will be explained), we may define an elliptic curve $\mathcal{E}$ using set notation over a general field F as follows [4]:

$$\mathcal{E} = \{\ (x, y) \in F^2 \mid y^2 = x^3 + Ax + B,\ 4A^3 + 27B^2 = 0\ \} \ \cup\ \{\mathcal{O}\}$$

Though it is far easier to visualize the curves over common fields (compare figures 2 and 4), the key property of elliptic curves that makes them useful in cryptography, the group law of point addition, applies only over finite fields [17].

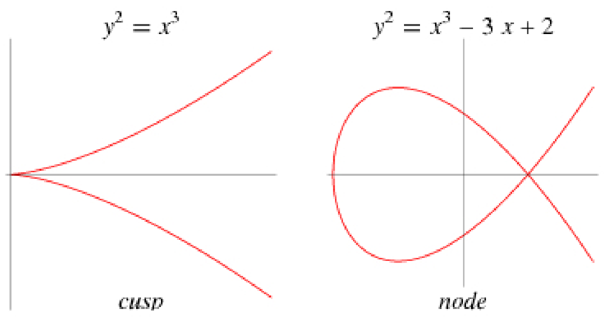


Figure 3: (*Left*) When $\Delta = 0$ and $A = 0$, the plot has a cusp singularity. (*Right*) $\Delta = 0$ and $A \neq 0$, the plot is self-intersecting. From [13].

Point Addition

The set of points defining an elliptic curve may be transformed into a group by defining the point addition group law, called the *chord-and-tangent method* by some mathematicians [3]. We begin by supposing that the set of points E meets the definition of an elliptic curve and is generally described by the equation $y^2 = x^3 + Ax + B$ [3]. Define point addition for three *collinear* points P, Q, and R on E as $P + Q + R = 0$. By this definition, for two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ on E, we can naturally write $P + Q = -R$. This amounts to saying the the addition of P and Q is done by drawing a line through the two points, and due to the requirement that $\Delta = -16(4A^3 + 27B^2) \neq 0$, the line is guaranteed to intersect E at another point, R. The reflection of this point across the curve then corresponds to the sum of P and Q, i.e., $P + Q = -R$, where $-R$ is the second point intersecting the curve on the vertical line through R (see Figure 5(a)) [17, 4].

By this definition of point addition, we encounter an issue when attempting to add a point and its inverse, since a vertical line will only intersect the curve twice (at most). This explains the necessity of our point at infinity, $\mathcal{O}$:

$$P + (-P) = (-P) + P = \mathcal{O} \iff P + \mathcal{O} = \mathcal{O} + P = P, \quad \forall \text{ (means “for all”) } P \in E.$$

Figure 5(c) shows another special case of point addition, occurring either where a point is

[3]For clarity's sake, we describe the law over the real numbers instead of over a finite field.

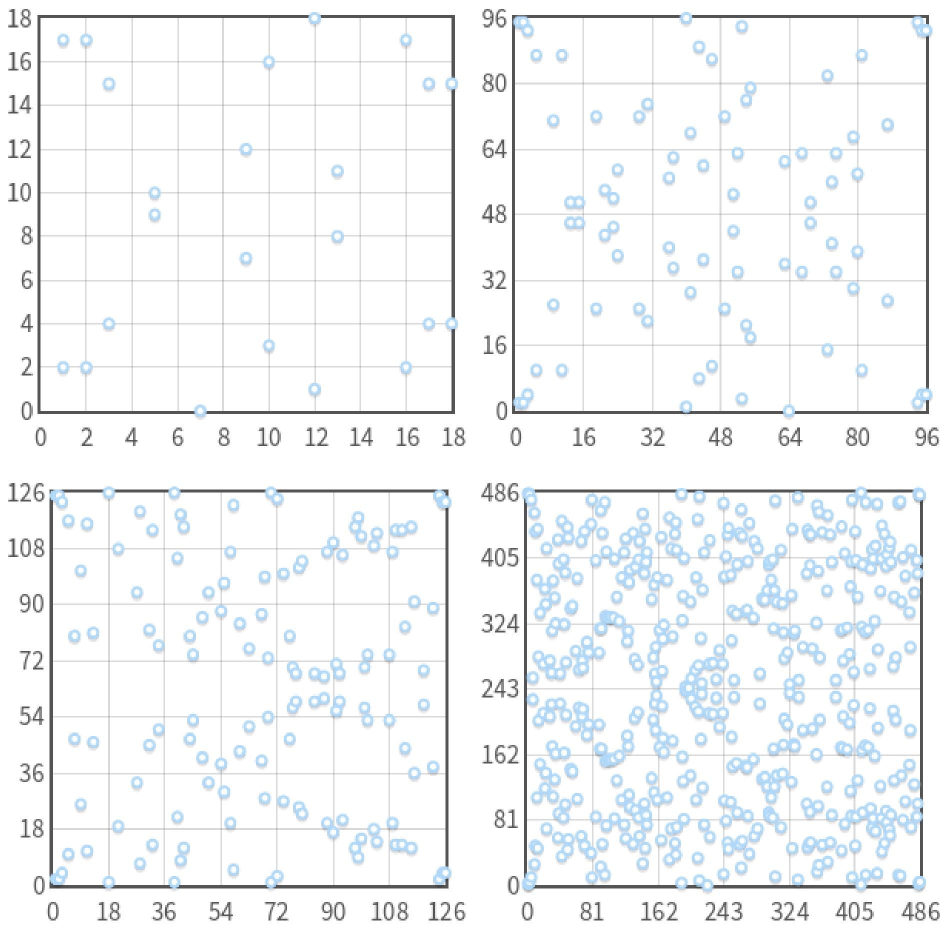
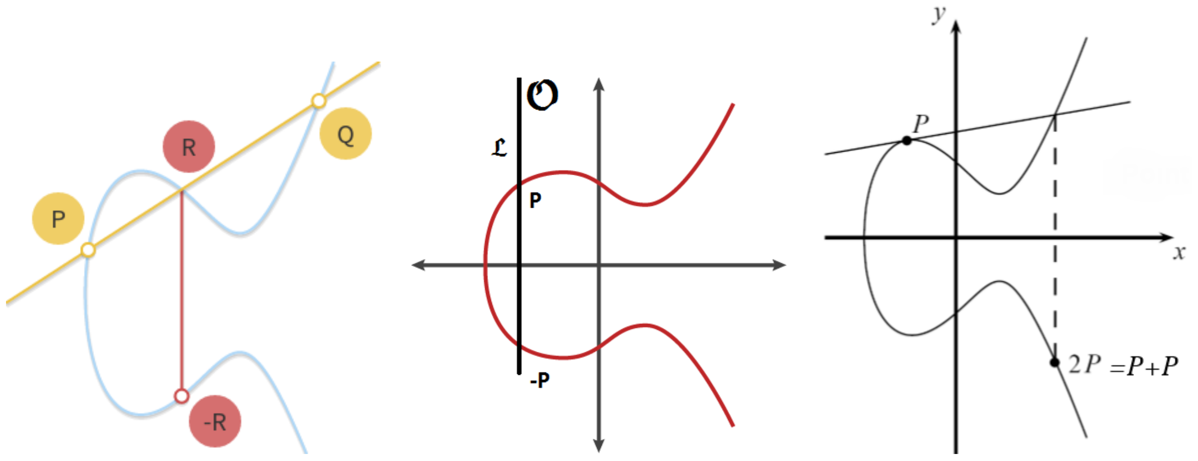


Figure 4: Elliptic curve $y^2 \equiv x^3 - 7x + 10 \pmod{p}$ for $p = 19, 97, 127, 487$ (finite fields).

added to itself ($P + P$) or where P and Q are distinct, but there is no third point on the line connecting the two points in question. The latter case may only occur when the line intersecting P and Q is tangent to the curve, and so the sum becomes symmetric to the point where the tangency occurs (i.e., if the line is tangent at P, then $P + P = -Q$ and if the line is tangent at Q, then $P + Q = -Q$). The former case (the one that Figure 5(c) specifically addresses) is calculated by a similar procedure: if we attempt to add P to P, then we pick two points around P and let them approach each other, resulting in a line tangent to the curve at P [17]. This procedure generalizes to adding a point P to itself any



(a) Typical addition of two distinct points

(b) Addition of a point and its inverse

(c) Point doubling via a tangent line

Figure 5: Various cases of point addition on elliptic curves over $\mathbb{R}$.

number of times, which amounts to scalar multiplication:

$$kP = \underbrace{P + P + \ldots + P}_{\text{k times}} \qquad \text{where } k \text{ is a natural number.}$$

The procedure for point addition may be summarized by the following formulas, described over a finite field of order p (the equations also hold over other fields when mod p is removed). For $P = (x_P, y_P), Q = (x_Q, y_Q) \in E$, where m denotes the slope of the line through P and Q, $P + Q = -R = -(x_R, y_R)$ may be calculated as follows[4]:

$$\begin{aligned} m &= \begin{cases} (y_Q - y_Q)(x_P - y_P)^{-1} \pmod p & \text{if } P \neq Q \\ (3x_P^2 + A)(2y_P)^{-1} \pmod p & \text{if } P = Q \end{cases} \\ x_R &= (m^2 - x_P - x_Q) \pmod p \\ y_R &= [y_p + m(x_R - x_P)] \pmod p \\ &= [y_Q + m(x_R - x_Q)] \pmod p \qquad [4] \end{aligned}$$

[4]Note that the equation for the line's slope is written to reflect the modular multiplicative inverse.

Discrete Logarithm Problem

Now that we have defined the point addition group law for elliptic curves, we are equipped to describe why elliptic curves are useful in cryptography. Elliptic curve cryptography (ECC) uses a system of public and private keys to encrypt data. This system is quite common in modern cryptography [11] and is implemented by one of the most well-known encryption algorithms, RSA (after *R*ivest, *S*hamir, and *A*dleman, the algorithm's designers). Both ECC and RSA are dependent on a *trapdoor function* to prevent adversaries from decrypting the data.

Definition 6. A *trapdoor function* is a function τ from some domain X to some target Y. The key idea behind the function is that, given $x \in X$, calculating $\tau(x) = y \in Y$ is simple, while calculation in the reverse direction is astronomically more difficult [11]. That is, using some lightly abusive notation,

$$\begin{aligned} \text{For function} \quad & \tau : X \to Y \text{ and } x \in \tau, \tau(x) \to y, \\ \text{but} \quad & \tau^{-1}(y) \not\to x. \end{aligned}$$

In the RSA cryptosystem, the trapdoor function is multiplication of prime numbers: though you might easily be able to calculate 37×59, it is not immediately obvious that the prime factors of 2183 are 37 and 59. Obviously you still could factor 2183 with relative ease, but now imagine that the product is a 2048-bit binary number. The difficulty of this factorization (which, with modern computational power, would take longer than the life of the universe for a long enough number) is what makes RSA encrypted data so secure and reliable [11].

ECC cryptosystems operate on a similar principle. Even if you know the start and end points of a series of calculations, due to the mod p operation performed every time the group law is applied to a point on the curve over $GF(p)$, deducing how many times the law was applied before reaching the end point is nearly impossible for well-chosen curves [4].

Consider an example with start point $P = (3, 6)$ and end point $Q = (3, 91)$ on the elliptic curve $y^2 \equiv x^3 + 2x + 3 \pmod{97}$. It would actually not be possible to determine how Q was calculated from P, since $4P = (3, 91)$, but so too does $9P = (3, 91)$. Further still, $4P = 9P = 14P = 19P = \ldots = (3, 91)$, or generally for any integer k, $(5k+4) = 4P = (3, 91)$ [4]. When we consider this same process being performed over the field $GF(p)$ where p is very large, it becomes clear that the problem is computationally complex.

This brings us to the *discrete logarithm problem* (DLP) [5] for elliptic curves (ECDLP): if given points P and Q on an elliptic curve E over $GF(p)$, what is k such that $kP = Q$? The smallest integer solution k to this problem is then called the discrete logarithm (or index) of Q with respect to P, denoted by $k = log_P(Q)$ (or by $k = ind_P(Q)$). Such a problem would be quite simple in certain fields, but over a finite field with very large p, it becomes intractable with modern computers [17]. The difficulty is so pronounced that it is actually why ECC started to become popular in the first place: the security provided by a key size of 256 bits in an ECC system is comparable to the security provided by a key size of *3072 bits* in an RSA system [4], and the difference between the two exponentially increases with key size. This means that ECC can be implemented far more efficiently than RSA and similar algorithms when attempting to meet the needs of modern data security [18].

Dual_EC_DRBG

Finally, we arrive back at the motivation of this inquisition: the Dual_EC_DRBG, or the dual elliptic curve deterministic random bit generator ("dual" refers to the two points P and Q) [2]. Dual_EC_DRBG (from here on, simply "Dual EC") is an algorithm intended to provide cryptographically secure pseudorandom bits; that is, though the mathematical nature of how computers work essentially make it impossible for software alone to generate *truly* random numbers, algorithms may be devised that generate numbers which would require near-infinite

[5]The term originates from an analogous problem in the multiplicative group $\mathbb{F}_q^*$: given $a \in \mathbb{F}_q^*$ and $b = a^k$, solve $k = log_a(b)$ [18].

computational power to predict. Dual EC first emerged in the early 2000s, eventually being standardized by the National Institue for Standards and Technology (NIST) in a public domain document titled "NIST Special Publication 800-90A", originally published in 2007 (though the standard was adopted as early as 2006) [2]. The document itself standardized four different pseudorandom number generators (PRNGs), with Dual EC being the only algorithm utilizing elliptic curves, as well as the only algorithm that has obvious potential for a backdoor.

At first glance, it may almost seem good that the Dual EC system was implemented as a PRNG, as opposed to directly encrypting transmissions; in fact, the opposite is true. Random number generators underpin the entirety of modern cryptographic systems. If you manage to gain a skeleton key to a generator—that is, a key to make the numbers predictable—you make useless all the other algorithms that utilize it, such as encryption key generators, random authentication challenges, key agreement schemes, prime number generators, and more [15]. One of the truly suprising things about Dual EC is that the vulnerablility of the algorithm is really quite obvious. The standard given by the NIST gives a list of explicit parameters (actually three different lists for different constant sizes, though we present the most commonly used one) describing the elliptic curve behind the algorithm. The equation defining the curve is given, along with the prime order p of the finite field $GF(p)$ [1, p. 77]:

$$\begin{aligned}
y^2 &= x^3 - 3x + b \pmod{p} \qquad \text{where} \\
p &= 115792089210356248762697446949407573530\\
&\qquad 086143415290314195533631308867097853951 \\
&= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
b &= \text{5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0} \\
&\qquad \text{cc53b0f6 3bce3c3e 27d2604b} \ \text{(in hexadecimal)} \\
&\approx 4.1058 \times 10^{76} \text{ (in decimal).}
\end{aligned}$$

Also given (in hexadecimal) in the NIST document are both the coordinates of the point used as a subgroup generator, P, and those of the point Q in the subgroup reached by scalar

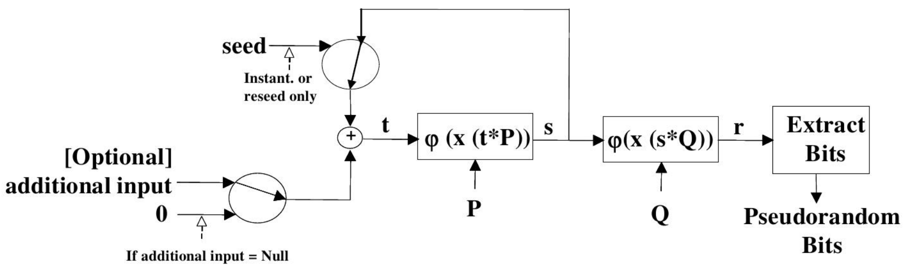


Figure 6: Flow chart in the NIST document showing the function of Dual EC, where φ is Euler's totient function. The "seed" is uniquely chosen by each user, and is the reason why the NSA would (allegedly, of course) need to observe about 32 bytes of output before cracking the encryption.

multiplication of P by the scalar k (note that $k = log_P(Q)$ is the skeleton key allegedly known to the NSA):

$$
\begin{aligned}
P &= (\text{6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296},\\
&\qquad \text{4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5})\\
Q &= (\text{c97445f4 5cdef9f0 d3e05e1e 585fc297 235b82b5 be8ff3ef ca67c598 52018192},\\
&\qquad \text{b28ef557 ba31dfcb dd21ac46 e2a91e3c 304f44cb 87058ada 2cb81515 1e610046}).
\end{aligned}
$$

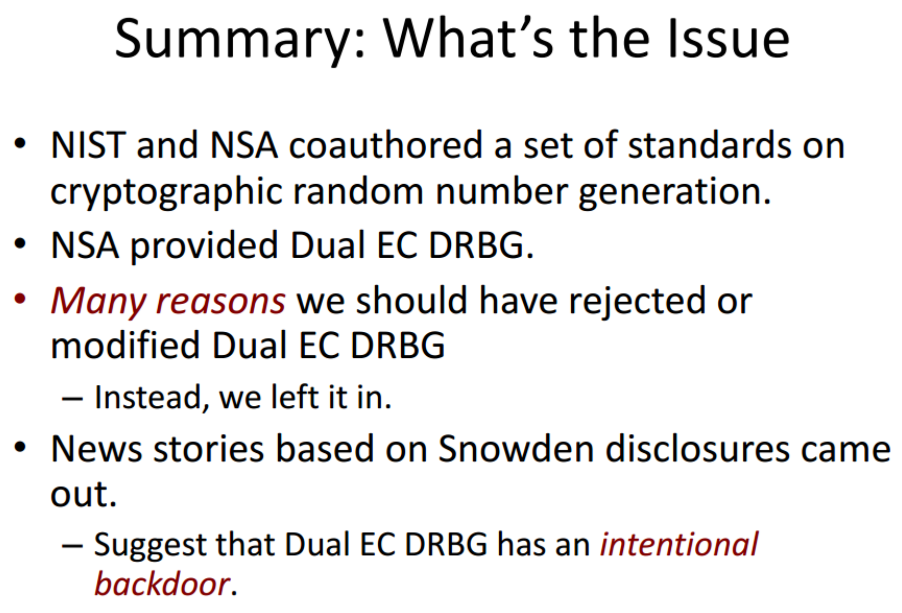
Examining the points P and Q here, it is obvious why cryptographers were suspicous of the Dual EC: as put by Thomas Hales in a notice from the American Mathematical Society (AMS): once the scalar k is known, it is a "simple matter to determine the secret internal state s of the pseudo-random bit generator" [6], by observing as few as *32 bytes of output* [16]. When Don Johnson, a contractor who worked under the NSA in the development of Dual EC, was asked by one of the coauthors of the NIST document—John Kelsey—where Q came from, the result was the following email exhange [2, p. 8-9]:

```
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
```

```
264 D.J. Bernstein et al.
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 simplest way that Q could have been generated in the system is by use of the "random private key" mentioned by Johnson. Curiously enough, however, this private key is exactly the information one would need to install a backdoor in Dual EC. The original NIST document does not specify how P and Q were chosen, but the NSA eventually responded to criticisms by saying that it had "generated (P, Q) in a secure, classified way" [2, p. 9]. It would have been a simple matter for the NIST to generate Q to be "like a(nother) canonical G", as Johnson put it, however this would have made it much more difficult for the designers of Dual EC to know the private key. Given this fact, it is difficult to explain why the NSA did not allow Johnson to "publically discuss it [the more secure implementation idea]", unless it was attempting to conceal the existence of a skeleton key. As Bernstein et al. put it, "[i]n hindsight it is quite amazing how blindly NIST trusted NSA" [2, p. 10].

Essentially from its initial conception, Dual EC was criticized for this obvious weakness. For this reason, the NIST added an appendix to the SP 800-90A document which would allow users to generate their own points on the curve; unfortunately, it also specifically recommended that clients not do this [1, p. 79]. In fact, the NIST created a strong incentive for use of the original points, only providing FIPS (Federal Information Processing Standard, needed for a company's software to be used by federal agencies) validation to clients that use the given P and Q. Better yet, since professional cryptographers were justifiably suspicous of the system presented by the NIST (at the NSA's behest), reports in December 2013 indicated that the NSA actually paid the company RSA Security "$10 million in a deal that set [Dual EC] as the preferred, or default, method for number generation in the BSafe software," RSA's widely used cryptographic library [2, p. 2]. Such an explanation seems plausible considering that the Dual EC PRNG algorithm—being many orders of magnitude slower than alternatives and also essentially guaranteed to yield predictable numbers to the algorithm's designers—was nonetheless implemented by cryptographic experts who should have known better.


Summary: What's the Issue

- NIST and NSA coauthored a set of standards on cryptographic random number generation.
- NSA provided Dual EC DRBG.
- *Many reasons* we should have rejected or modified Dual EC DRBG
  – Instead, we left it in.
- News stories based on Snowden disclosures came out.
  – Suggest that Dual EC DRBG has an *intentional backdoor*.

Figure 7: Slide three from Kelsey's NIST presentation.

Conclusion

The situation almost seems like a typical conspiracy theory, if we put aside all the glaring and salient evidence, that is. Even coauthor of the standard John Kelsey has had no choice but to recognize the NIST's naivete (see figure 7, a slide from an official presentation by Kelsey). This same presentation shows why it is so concerning that the NSA behaved as they did: Kelsey says that the Dual EC debacle resulted from the NIST's reliance "on NSA for expertise" which the NIST lacked on elliptic curve cryptography. He further states that NIST officials "[t]rusted NSA input where [they] would have been much more skeptical of anyone else" [7]. There is no compelling reason to believe the NIST conspired with the NSA. Instead, one part of the government genuinely sought to protect data on behalf of American citizens and companies, while another part—which was entrusted with this same aim—sought to deliberately undermine it, obstensibly for its own self-interest. This leads quite directly to mistrust in intelligence services like the NSA, weakening the legitimacy of an organization charged with the responsibility of protecting Americans from the kind of secretive, dark forces that one hopes to never encounter. Furthermore, it shows that a collective of some of the most clever cryptographers in the world is being put to work compromising the security of data, rather than protecting it. Data does not discriminate by user; weakening encryption in an attempt to prevent terrorism, for example, makes everbody's data vulnerable, not just the data of the intended targets. One can only hope that—from the perspective of an NSA insider—the agency's actions were far more justified than it appears they were from afar.

References

[1] Barker, Elaine & Kelsey, John. *NIST Special Publication 800-90A.* National Institue of Standards and Technology (Last revised: January 2012). http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

[2] Bernstein, Daniel J., et al. *Dual EC: A Standardized Back Door.* European Commision (2015). https://projectbullrun.org/dual-ec/documents/dual-ec-20150731.pdf

[3] Brown, Ezra. *Adding Points on Elliptic Curves.* http://www.math.vt.edu/people/brown/class_homepages/elliptic_curve_addition.pdf

[4] Corbellini, Andrea. *Elliptic Curve Cryptography.* (Series of four posts, starting with) http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

[5] Freeman, David M. *Facts about Finite Fields.* Stanford (2011). http://theory.stanford.edu/~dfreeman/cs259c-f11/lectures/finitefields.pdf

[6] Hales, T.C. *The NSA Back Door to NIST.* AMS 61(2), 190192 (2013). http://www.ams.org/notices/201402/rnoti-p190.pdf

[7] Kelsey, John. *Dual EC in X9.82 and SP 800-90.* NIST (2014). http://csrc.nist.gov/groups/ST/crypto-review/documents/dualec_in_X982_and_sp800-90.pdf

[8] Morris, Steve. *Everything You Need to Know About modular Arithmetic...* Cornell (2006). http://www.math.cornell.edu/~morris/135/\protect\unhbox\voidb@x\hbox{mod}.pdf

[9] Nie, Jiawang. *Lecture 18: Groups, Rings, Fields and Ideals.* University of California San Diego (2010). http://www.math.ucsd.edu/~njw/Teaching/Math271C/Lecture_18.pdf

[10] Perlroth, Nicole & Larson, Jeff & Shane, Scott. *N.S.A. Able to Foil Basic Safeguards of Privacy on Web. New York Times* (2013). http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?_r=0

[11] Roeder, Tom. *Asymmetric-Key Cryptography.* Cornell (2013). http://www.cs.cornell.edu/courses/cs5430/2013sp/TL04.asymmetric.html#sec-17

[12] Roman, Steven. *Fundamentals of Group Theory.* Birkhauser/Springer (2012). DOI 10.1007/978-0-8176-8301-6.

[13] Rowland, Todd. *Elliptic Discriminant.* From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/EllipticDiscriminant.html

[14] Selyukh, Alina & Shahani, Aarti. *Apple-Justice Department Standoff Over iPhone Access Goes On, In New York.* NPR (2016). http://www.npr.org/sections/thetwo-way/2016/04/08/473423254/apple-justice-department-standoff-over-iphone-access-goes-on-in-new-yorks.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?_r=0

[15] Schneier, Bruce. *Did NSA Put a Secret Backdoor in New Encryption Standard?* Wired (2007). http://www.wired.com/2007/11/securitymatters-1115/

[16] Shumow, Dan & Ferguson, Niels. *On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng.* Microsoft (2007). http://rump2007.cr.yp.to/15-shumow.pdf

[17] Silverman, Joseph H. *An Introduction to the Theory of Elliptic Curves.* Brown University (2006). http://www.math.brown.edu/~jhs/Presentations/WyomingEllipticCurve.pdf

[18] Sutherland, Andrew. *18.783 Elliptic Curves Lecture 1.* MIT (2015). http://ocw.mit.edu/courses/mathematics/18-783-elliptic-curves-spring-2015/lecture-notes/MIT18_783S15_lec1.pdf

[19] Zetter, Kim. *How a Crypto 'Backdoor' Pitted the Tech World Against the NSA.* Wired (2013). http://www.wired.com/2013/09/nsa-backdoor/

Image Credits

Figure 1(a,b): Generated on http://www.wolframalpha.com/

Figure 2(a): Retrieved from https://upload.wikimedia.org/wikipedia/commons/thumb/d/d0/ECClines-3.svg/2000px-ECClines-3.svg.png

Figure 2(b): Screenshotted from http://www.csuchico.edu/~tmattman/althesis.pdf

Figure 3: Retrieved from http://mathworld.wolfram.com/EllipticDiscriminant.html

Figure 4: Retrieved from http://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields

Figure 5(a): Retrieved from http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-i

Figure 5(b): Custom image by the author

Figure 5(c): Retrieved from http://cs.ucsb.edu/~koc/cren/docs/w03/09-ecc.pdf

Figure 6: Retrieved from http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

Figure 7: Retrieved from http://csrc.nist.gov/groups/ST/crypto-review/documents/dualec_inX982_and_sp800-90.pdf (slide 3)