**Elliptic Curve Cryptography, Random Number Generators and the NSA**

Research Project

COMP90043 Cryptography and Security

Geordie Wicks - 185828

**Introduction**

Elliptic Curve Cryptography (ECC) is a form of cryptography similar to RSA. It is based around public / private key cryptography and the use of a one-way mathematical function that is easy in one direction, and difficult in another. Functionally it works in a very similar way to RSA, just with a more complex mathematical function, which makes it much harder to brute force than RSA with a similar key size.

In this research project I will briefly explain the origins of ECC, the type of mathematical functions used, and show in simply terms an example implementation.

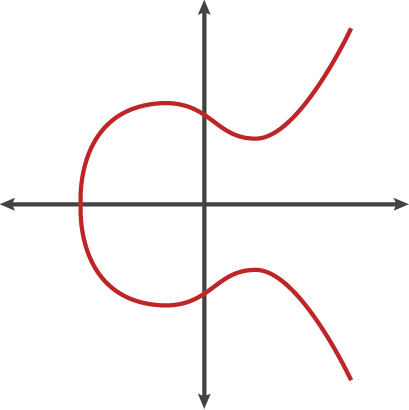
This research project will also investigate the security of a Pseudo Random number generator that uses elliptic curves, what that means for ECC security in general and attempt to draw conclusions about whether the NIST (National Institute of Standards and Technology) implementation of ECC was compromised by the NSA from the very beginning. Given the research findings the report will conclude that using any NIST certified ECC scheme has risks associated with it that outweigh the benefits of elliptic curve cryptography, and that these risks and the lack of trust they inspire are a real risk to both the internet and the economy.

**Background**

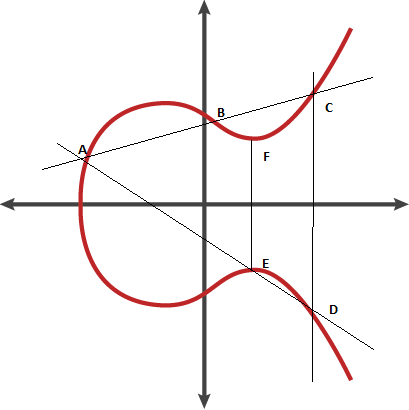
Elliptic Curve Cryptography was first suggested as a form on encryption in 1985. Like all forms of encryption it uses the idea of a one-way function, a mathematical problem easy in one direction and hard in the other. Instead of using the factorisation or modulo of prime numbers, ECC utilises a form of the discrete logarithm function, in that it is assumed that finding the discrete logarithm of a random elliptic curve with respect to a public base point is hard.

The primary reason for using ECC over other form of encryption is due to it’s high level of security for a given key length. For example a 224 bit ECC key is equivalent to a 2048 bit RSA key, which means less information needs to be transmitted and less computation work needs to be done.

ECC works by taking an elliptic curve of the form = + ax + b,

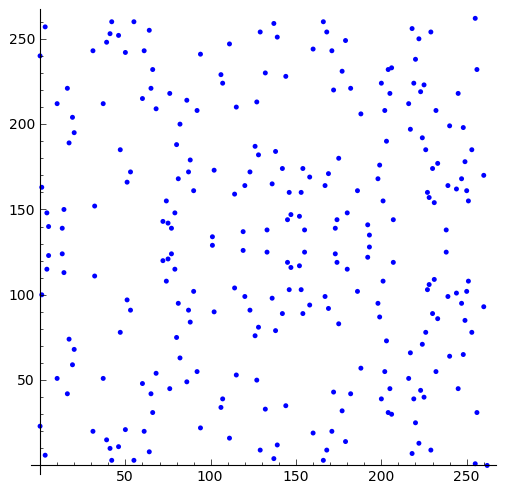


where a and b are randomly chosen, choosing a public starting position (A), then performing point addition (Drawing a line through two points on the line, A & B, until it crosses the curve in the 3rd place, C)



Point C is then reflected across the y-axis to yield point D. By drawing a line between the starting point A and point D, the curve is again crossed in the 3rd place (E). E is then reflected back across the y-axis.

By repeating this process n times, you end up with a map of the curve in dots (places the lines have crossed the curve).



The one-way function works as follows:

Public Information : the curve itself ( i.e. a & b are public )

The start and end positions , (J & K)

A prime modulo P

The private key is a large prime number, n, which signifies how many times the point addition process is carried out.

An useful analogy is the game of pool. At the start of the game you know the positions of all the balls on the table. If you left the room and came back when the game was over, you could check the pockets and see the end position of each ball. Knowing the start and end position of a certain ball however, does not make it easy to know the path the ball in question took to end up in that pocket.

In practice the equation is also mod with p.

PUBLIC: = + ax + b (mod p), (j, k)

PRIVATE: n

With this basic principle, ECC can be used for key exchange, for encryption, for random number generation and for Digital Signatures. It is important to note that the curve used (i.e. the a and b values), can have a large impact on the security of ECC, and thus should be chosen carefully.

**Elliptic Curve Cryptography and the NSA**

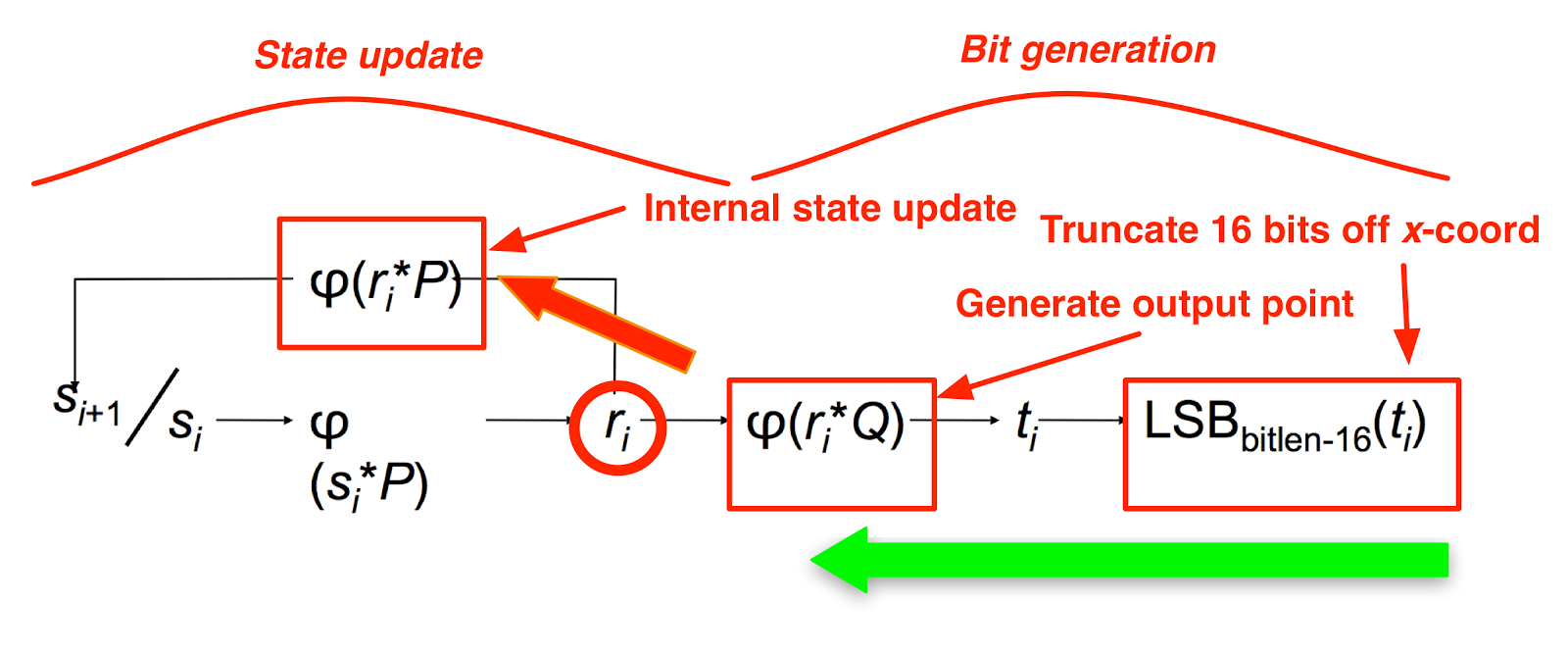
Random numbers are crucial for cryptography, and one of the first places ECC was used was in the generation of pseudo random numbers [PRNG]. In general the generation of random numbers is very difficult, as there is no mathematical way to prove how random a sequence of number is. In the early and mid 90’s, researchers were finding flaws in multiple different PRNG’s, and cryptologists and mathematicians starting hunting for a new, superior algorithms for generating random numbers.  
In 2006 the American National Institute of Standards and Technology (NIST) released new official standards for random number generators, approving four different techniques for their creation. One was based on hash functions, one on HMAC, one on block ciphers, and the last on elliptic curves. Because the NSA (National Security Agency) is the USA’s premier expert in all thing cryptology, they were intimately involved in the selection of generators.

The one based on elliptic curves was called Dual\_EC\_DRBG ( Dual elliptic curve deterministic random bit generator). At the time of it’s inclusion it caused some controversy, as not only was it several orders of magnitude slower than competing generators, but the random numbers it generated could be shown to have a small bias.~

The PRNG would generate random numbers by generating a series of (x,y) co-ordinates on the elliptic curve. These coordinates are elements of a finite field, specifically x and y are numbers in the range 0 to p-1 where p is a large prime number, and (x,y) satisfy the elliptic curve equation = + ax + b mod p.

These points are obviously not random, as they are numbers less than some prime number, and because they satisfy the elliptic curve equation. Thus the second phase of the generator is to extract some ( but not all ) of the bits from the (x,y) points. There are many different methods of doing this, Dual\_EC\_DRBG achieves it by dropping the most significant 16 bits, and outputting the rest of the number, unlike other EC PRG’s which drop up to 2/3 of the most significant bits.

This crucial difference is what leads Dual\_EC\_DRBG to have a small bias. Due to some mathematical properties of the field operations, an attacker now a small, but non-trivial, chance of predicting the next bits of output, calculated at around 0.1%. This is not enough to make it unusable, but a curious choice when faster and more random options were available. It was a weakness that simply did not need to be there, since dropping more bits would have made the generators significantly more secure, with little computational overhead. Whilst in theory it was a vulnerability, in practice it was not a big deal, as to take advantage of it an attacker would need to brute force the missing 16 bits and solve the elliptic curve equation for y, doable for one point, but not every point on the curve, which would be needed to predict the next bits outputted by the generator. It was only when another weakness was discovered however, which works in concert with the biased output, that serious questions about Dual\_EC\_DRBG and the possibility of a backdoor were raised.



**Figure 1.4**

The generator is reliant on two values, P, which is the generator, and Q, which lies on the curve and should be chosen randomly. The green arrow in figure 1.4 represents the weakness discussed above, whilst not easy to go from the output to the (x,y) values before the bits were removed, it is possible via a brute force attack. The reason it is not a deal breaker for the algorithm is because, with properly chosen P and Q values, it should be impossible to go from r\_i\*Q to r\_i\*P, as the \* operator represents complex elliptic curve multiplication which is hard to reverse. ( Thick Red Arrow)

However not long after the standards were finalised a presentation was given by Dan Shumow and Niels Ferguson at the CRYPTO 2007 conference, showing how mathematically, if the P and Q values were not chosen at random by the implementers, but chosen careful in concert with the elliptic curve itself, then it was possible that there could be a back door in the algorithm. This is because the elliptic curves NIST selected have prime order. This means there exists an *e* such that = P. Knowing *e* means both steps in the algorithm that are supposed to be hard to do backwards, are in fact relatively easy to do. In fact Shumow and Ferguson showed how an attacker would need only 32 bytes of output to be able to uniquely identify the internal state of the PRNG.

One of the interesting things about this is that nobody except the algorithms designers could prove there was a back-door, as it depended on how they had chosen the constants P and Q. Essentially, whoever designed the algorithm and chose the constants could have generated a private key for themselves.

For years after this controversy there was little evidence pointing one way or the other conclusively. Security researchers like Bruce Schneier were convinced the back door existed, but the NSA and RSA have always denied it.

Some of the arguments for a backdoor included:

* DUAL\_EC\_DRBG was known to have security issues prior to standards being set
* Nobody from either NIST or the NSA has said where P and Q come from
* NIST published a method of generating P and Q randomly in the same standards, but made the procedure to do so optional
* The NSA had been pushing ECC for a while, and publically championed Dual\_EC\_DRBG to be included in the standard, despite knowing of its problems.

For a while it looked the issue would remain unresolved, with most of the world continuing to use Dual\_EC\_DRBG without worry, however in September of 2013, included in files released by Edward Snowden, was information relating to a NSA program known as Bullrun. It is described in the documents as existing “*to covertly introduce weaknesses into the encryption standards followed by hardware and software developers around the world”*

The New York Times states the “the NSA had inserted a back door into a 2006 standard called the Dual EC DRBG standard”, but does not disclose the documents it saw.

In December of the same year Reuters reports that the NSA paid RSA $10 million dollars to set Dual\_EC\_DRBG as the default random number generator in some of their products.

**Analysis**

Whilst no proof is possible and a public admission by NIST or the NSA is extremely unlikely, it is now generally accepted that Dual\_EC\_DRBG is compromised. Due to the controversy in September 2013 NIST published a guideline that “*strongly recommends that, pending the resolution of the security….Dual\_EC\_DRBG…no longer be used*”. [wiki]

This controversy has also cast suspicion on all of the elliptic curves selected by NIST got be included in its standards. Similar to how P and Q were chosen arbitrarily in Dual\_EC\_DRBG, with no explanation of how they came about, the a and b values in all of the NIST elliptic curves ( constants in the curve = + ax + b ), likewise were chosen with no explanation or documentation provided as to how they came about. This leads to the possibility that they were chosen because they belong to a publically unknown class of weak curves, and also have back doors built in for the NSA. Given how rarely ECC is used for random number generation as opposed to how commonly it is used for encryption / key exchange, it is fair to ask: Would the NSA go to this much trouble to compromise a PRNG, and not try to compromise the general ECC curves as well? And given that, if you needed encryption for something sensitive, would you trust ECC? Are the speed and size benefits greater than the risk of using a compromised implementation?

**Conclusion**

When NIST first published the standards for approved Pseudo Random Number Generators, it was obvious there was something strange about the slow, insecure version using elliptic curve cryptography. It was simply hard to believe some of the smartest crypto minds in the world would implement something so flawed, and have no explanation for where key parameters came from.

In 2007 when Shumow and Ferguson showed at CRYPTO 2007 that carefully chosen parameters to the generator could enable a back door, security researchers were convinced the algorithm was suspect.

In 2013 when Edward Snowden’s leak started becoming public, and the operation code named BULLRUN outlined the NSA’s efforts to break cryptography standards, the idea that Dual\_EC\_DRBG was back doored moved from fringe conspiracy theory to fact in all but name.

This situation must leave any security professional, software developer, or citizen concerned with their security and privacy necessarily sceptical of the security of all elliptic curve cryptography. Whilst ECC offers tangible benefits over RSA, lower overhead and higher speed do not make up for the possibility of a

back door. Indeed, the whole sordid tale of the efforts the NSA went to to convince NIST and RSA to implement their suspect algorithm as the standard, leads to a lack of trust in all NIST standards.

Cryptography is essential for the modern internet, and indeed our modern lives. Consumers need to be able to trust that their bank details won’t be stolen just like businesses need to be able to trust that their communications and transactions are private.

By deliberately weakening a cryptographic standard for their own benefit, the NSA and NIST have done irrevocable harm to that trust. It leads one to wonder what else they have broken, all Elliptic Curve Standards? Other standards as well?

It took a long time for people to start to trust they could conduct business on the internet, and it has revolutionised how we do commerce. If that trust gets weakened to the point where people or organisations start to seriously question their online security, the possible ramifications, both to the internet and the economy, could be catastrophic.

**References**

<https://en.wikipedia.org/wiki/Dual_EC_DRBG#Security>

<http://rump2007.cr.yp.to/15-shumow.pdf>

<https://www.schneier.com/essays/archives/2007/11/did_nsa_put_a_secret.html>

<http://andrea.corbellini.name/2015/06/08/elliptic-curve-cryptography-breaking-security-and-a-comparison-with-rsa/>

<http://www.secg.org/sec1-v2.pdf>

<http://www.wired.com/2013/09/nsa-backdoor/>

<http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?pagewanted=all&_r=0>

<http://blog.cryptographyengineering.com/2015/01/hopefully-last-post-ill-ever-write-on.html>

<http://blog.cryptographyengineering.com/2013/09/the-many-flaws-of-dualecdrbg.html>

<https://www.schneier.com/paper-prngs.pdf>

<https://www.schneier.com/blog/archives/2007/11/the_strange_sto.html>

<https://wiki.openssl.org/index.php/Elliptic_Curve_Cryptography>