**GROUP 48**

**Cryptography Presentation**

Hi everyone, my Name is Geordie Wicks, and today I am going to talk about elliptic curve cryptography, ( or ECC ), pseudo random number generators, and the NSA. First I will give you a quick overview of ECC, what it is, how it works, then I will discuss one of the first places it was used, a pseudo random number generator, and discuss whether or not it was back doored by the NSA, and what this means for ECC, standards and the internet in general.

First: what is an elliptic curve?

***Slide 2***

They all look like some version of this curve, the important points that make it elliptic are:

1. It’s mirrored about the y-axis, and
2. Any non-vertical line drawn through it will cross the curve in at most 3 places

ECC is similar to other types of cryptography we have looked at like RSA and Diffie-Hellman, in that it is a public key crypto system designed around a one-way function, in ECC’s case it is known as the Elliptic Curve Discrete Logarithm Problem. The benefit of ECC is that it has much higher security for any given key length. In fact a 224 bit ECC key is equivalent to a 2048 bit RSA key. This means much less information needs to be transmitted to perform key exchanges, which is obviously a huge advantage.

***Slide 3***

It works like this, you randomly choose two points P and Q on the curve. (Now it’s important that they are chosen randomly, as I will discuss later)

You then draw a line through P and Q until the line crosses the curve somewhere at a 3rd point R.

***Slide 4***

You then reflect point R across the x-axis, to get point P + Q

***Slide 5***

And you keep going like this, dot where it crosses curve, reflect across x-axis

***Slide 6***

The amount of times you do this, known as e, is your private key, usually a large prime, until you get a map of all the places you have dotted which looks like this:

***Slide 7***

This forms the basics of the one-way function, given a starting point, it is very easy computationally to perform the elliptic curve additions e number of times until you come to an end point. But going the other way, ie knowing the start and end points (which are both public information), it is very hard to know the path taken to get there.

You can think of it like a pool ball. At the start of the game you know where every ball is. Now if you left the room and came back at the end of the game, you could see which pocket each ball ended up in, but working backwards to find the path the ball took to get there is next to impossible. This is the ECC discrete logarithm problem.

***Slide 8***

I now want to talk a bit about Dual\_EC\_DRBG, or Dual Elliptic Curve Deterministic Random Bit Generator, which was the first NIST approved standard using Elliptic Curves. NIST is the National Institute of Standards and Technology, a US organisation that provides standards for lots of thing, including cryptography, and in order to use any form of Crypto, government departments must use one approved as a standard by NIST, and many corporations follow suit, assuming anything signed off by NIST must be secure.

Now around the middle to late 90’s, many of the existing random number generators where being found by mathematicians and crypto analysts to have flaws, so NIST started a program to come up with new standards.

***Slide 9***

They settled on four new standards for random number generation, one based on hash functions, one on MAC’s, one on block ciphers and the last on elliptic curves. At the time is was publically acknowledged that the NSA, as America’s premier experts on all thing cryptology, were intimately involved in the selection of the standards.

In 10 minutes I don’t have time to go into depth about how Dual\_EC\_DRBG works, but a quick overview is like this:

You randomly select a point on the curve. Obviously because the curve is part of a finite field, that coordinate can’t be truly random, so what you do is discard the most significant 16 bits, and out put the rest as your random number.

***Slide 10***

Now at the time of publication, this implementation that NIST chose caused some level of controversy, mainly because it was quite slow, and by only discarding 16 of the MST bits it left a small amount of bias. In practice what you could do was perform a brute force search on all possible combinations of those missing 16 bits, and you would end up with only a small handful of combinations that would lie on the curve. In practice this was not a big deal, as the bias was only about 0.1%, it was just thought to be a curious choice when for very little addition overhead they could have used competing designs which worked exactly the same way, only they threw away up to 2/3 of the Most significant bits, making a brute force attempt to find the original number impossible.

***Slide 11***

In this slide, the green arrow represents going from the output back to the original number. They could have made it next to impossible, but nobody was really worried because the thick red arrow in the slide represents a much harder one-way function, the elliptic curve discrete logarithm function.

Not long after the publication however, in 2007 two Microsoft researchers, Dan Shumow and Niels Ferguson held a presentation at CRYPTO 2007 that showed something startling.

The two constants, P and Q, which remember at the start I said should be chosen randomly, could in fact be chosen carefully, in concert with a chosen curve.

***Slide 12***

If the creators of the algorithm chose P, Q, a and b carefully, then they would only need to find one original point of the PRNG output, and they could easily calculate e, the private key. This was because the elliptic curve NIST selected had prime order, that means that there exists an e such that = P.

So if you now e from the backdoor, and you can find Q via the first weakness, it is trivial to find P.

This weakness, when combined with the original weakness, meant if you could brute force the missing 16 Most Significant Bits, you could break the randomness of the number generations and know what numbers were coming next.

In fact Dhumow and Ferguson showed how you would only need 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 there is no way to prove that the implementation has been back doored. If P and Q were chosen at random, it wouldn’t be, but if they were carefully selected it meant it could be, but nobody but the original designers can say for sure.

This is interesting because for the other 3 PRNG’s published by NIST, they all had appendixes which explained where the constants came from, Dual\_EC\_DRBG did not.

***Slide 13***

So that was the state of the matter at the end of 2007, and whilst security researchers like Bruce Schiner maintained Dual\_EC\_DRBG was broken, NIST and the NSA denied it, and most corporations including RSA continued to use it as their standard PRNG.

Then in 2013 something changed.

**Slide 14**

Edward Snowden Released a massive amount of information about the NSA’s spying apparatus, one of which was a program called BULLRUN

***Slide 15***

These are a couple of slide from the leaked documents!

***Slide 16***

This information, whilst not referring directly to Dual\_EC\_DRBG, was compelling enough for the Guardian in the UK, and the New York Times in America to conclusively state that it was back doored.

A few months later more information was leaked which outlined a deal between security company RSA and the NSA, in which RSA was paid $10 million dollars by the NSA to make Dual\_EC\_DRBG their default number generator in their most popular products.

***Slide 17***

So at this point in time, whilst without confirmation by NIST or the NSA it is impossible to say conclusively that Dual\_EC\_DRBG is back doored, the general consensus amounts the crypto community is that is almost certainly is.

OK, so at the start I said I would talk about what this means for Elliptic Curve Cryptography in general, and the issue is this:

* ECC is fast and efficient, both in computation time and transmission volume. With a proper implementation it appears superior to RSA.
* However, just like with Dual\_EC\_DRBG, the standards NIST has published for key exchange using ECC are somewhat suspect
* The problem is that certain elliptic curves are known to be weak and AGAIN, the curves constants are not explained in any way, usual with this kind of thing there is documentation saying why things are the way they are, not so with NIST’s standard ECC implementations.

Cryptographers have speculated that with their massive computing capacity, the NSA could have run billions of curves and found previously unknown weak curves, and chosen their parameters to enable back doors. Without knowing where the constants came from, it is impossible to be sure the standard curves are not back doored.

This kind of behaviour is reprehensible for a couple of reasons.

1. One obviously, the NSA shouldn’t be spying on us all at all times,
2. But more importantly, this kind of behaviour can weaken peoples confidence in cryptography and cryptography standards. The entire internet works on some level of trust, and e-commerce relies on it. If people and organisations start to think that the tools that are supposed to keep them safe have been engineered not be safe, then the possible effects to our modern, online economy could be devastating.

So in conclusion, what I would like you take away from this talk is that in all likelihood the NSA has back doored at least on elliptic curve algorithm, and it is at the very least quite possible they have back doored all of the standard ones. So we should all do what companies like apple have done, and come up with new curves that NIST and the NSA had nothing to do with standardising.

***Slide 18***