Elliptic Curve Cryptography

Phillip Marcus Harker Wilt

January 2014

Contents

1 Cryptography Primer

Cryptography has become an essential part of everyday life. It encompasses a large range of fields from secure communication, banking, and data protection to new applications in Bitcoin and Tor. With this data traveling across public channels, secure communication schemes have become a necessity.

For this reason, the field of Elliptic Curve Cryptography was an appealing alternative. Elliptic Curve Cryptography was introduced seperately in the mid-1980s by Neal Koblitz and Victor S. Miller [4, p. 135]. This used a then obscure mathematical concept that would use smaller key sizes and be computationally easy for parties involved but hard to break. However, there are weaknesses that can be introduced to the system.

In the wake of the Edward Snowden leaks of the United States' National Security Administration (NSA) documents, it appears there exists an inherent backdoor in one such implementation. Whether this was put in on purpose or

neglagence is speculation. Regardless, there is a serious security threat to be aware of in this particular cryptography technology. This paper will develop a brief history of crypotgraphy technology and then explore the famed Dual Elliptic Curve Pseudorandom Number Generator vunerability.

2 Diffie-Hellman Key Exchange

A solution to the problem over secure communication over public channels was proposed by Diffie-Hellman in 1976. The computation used a set of function suited for cryptography are that trapdoor, or one-way, functions. These are functions that are computationally easy to encode, but difficult to decode (meaning the inverse of the function is difficult to compute).

The exchange has two pieces: a public key and private key aspect for each party. The public keys are sent over public channels, whereas the private keys, as their name suggests, are private. Then there is a "trap-door" function that will allow each party to compute a shared number, known between both parties, but no one who witnessed the exchange. Diffie & Helmman theorized they could encode keys using properties of prime numbers and modular arthimatic.

In order for a third party to reak the Diffie-Hellman key exchange one would need to compute the private keys of each party. This algorithm involves efficiently solving the discrete log problem , which is a non-trivial task [1]. This key exchange works well if the goal is secure two-way communication.

3 RSA Crytography

The next major advance in cryptography technology came in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman. The RSA Cryptography Algorithm (initials for , the inventors of the algorithm and founders of RSA Security) solves the problem

Its security relies on factoring the public keys and obtaining a secret key. This is equivilent to solving the discrete log problem which is claimed to be very hard.

3.1 Weaknesses of RSA

One weakness of RSA is that the keys generated need to be very large, as computational speed increases and algorithms are developed efficiently to compute prime factorization. Originally, RSA offered prizes to factor specific length numbers, but eventually they were being done with such frequency that prizes for factoring were stopped.

Stemming from the problem of having to use such large numbers is the fact that it increases computational complexity, memory, and bandwidth requirements. With encryption being used on mobile phones, tablets, and other small devices, this takes up time and resources. In steps Elliptic Curve Cryptography to solve this problem.

4 Elliptic Curve Cryptography

The research into using Elliptic Curves for Cryptography (ECC) systems was done by Neil Koblitz and Victor Miller in the 1980s [4, p. 135]. It solves the same problem that is solved by RSA cryptography, but does so with numbers smaller in length. It also uses an excellent trapdoor function, making it hard for would-be intercepters to gain access to illicitly gathered information.

4.1 Elliptic Curves.

Now to take a look into the world of Elliptic Curves. The elliptic curve equation is in the form of Equation (1) with a generic plot shown in Figure 1.

$$y^2 = x^3 + ax + b \tag{1}$$

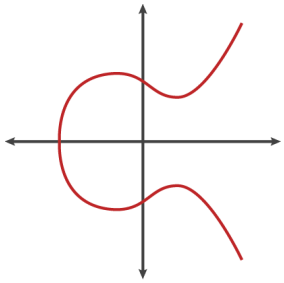

Figure 1: Generic Elliptic Curve (Image Courtesy [5])

The advantage of working with these types of curves is there is a unique group operation which is hard to invert. It is possible to take a line through two points on the curve and reflect the line across the x-axis to obtain a third point. This operation is precisely the trapdoor function needed to make Elliptic Curve Cryptography secure. For a more in depth discussion with proofs [4] is an excellent resource.

4.2 Dual Elliptic Curve Deterministic Random Bit Generator

For this paper we are going to concern ourselves with the application of Elliptic Curve Cryptography to random number generation and, in particular, to the Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG). This was specified in NIST 800-90A, which is the particular implementation implicated in the Snowden documents.

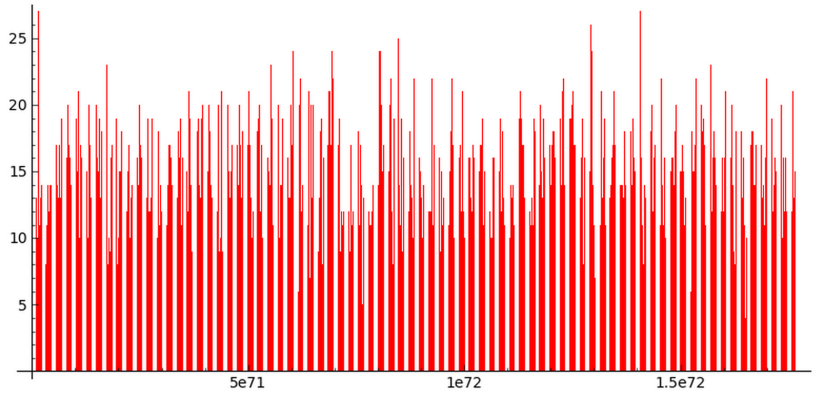

Figure 2: Histogram of 10,000 Iterations of Dual_EC_DRBG (Scaled)

4.3 Generating Secrets

Applications of random number generation are used to compute the keys that create a secure link between two parties. If someone were to know these numbers, they could eavesdrop on the conversation and possible forge false messages.

The random number generator works in two pieces, generating output and reseeding itself.

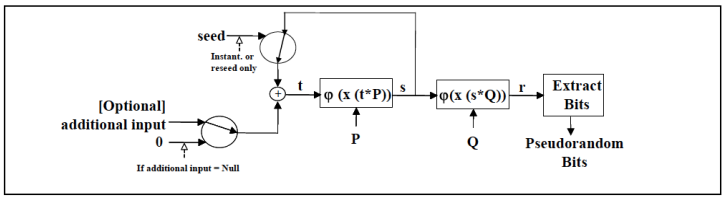

Figure 3: NIST 800-90A Spec

4.4 Dual_EC_DRBG and its Achilles' Heel

In 2007, Dan Shumow and Niels Ferguson of Microsoft Research proposed a backdoor into Dual_EC_DRBG [2].

Proof of Dual_EC_DRBG (Figure 3):

1. First we need elliptic curve functions, which are specified in NIST-800-90A (P-256 in proof of concept).

2. Pick points P and Q different and seemingly random on the curve.

3. Add P to itself t times, where $t \in \mathbb{Z}$ and produce a new point with x-coordinate s.

4. Multiply s with Q to output a random x-coordinate r

5. Extract 30-bytes (Least Significant) of r to output a random number.

6. Update next random seed with $t = s$.

If P and Q have no relation then this produces correct random numbers. However, if they have a relation a backdoor exists. Now suppose they have the relation given in Equation 2.

$$Q = d * P \tag{2}$$

If d is known then let a be the additive order of point P. Then it is easy to compute

$$e = d^{-1} \pmod{a}, e * d = 1 \pmod{a} \tag{3}$$

This will give the relation in equation 4, which will allow predicition of the state of the Dual_EC_DRBG.

$$e * Q = P \tag{4}$$

Given a 30-byte block of output A, it is easy to brute force guess candidates for the missing 2-bytes. Iterating over these candidates and checking

$$e * A = e * (s * Q) = s * (e * Q) = s * (P) = t * P \tag{5}$$

The term $t*P$ is the potential next state of the Dual_EC_DRBG. If two more bytes of the next output are known then they can be checked against this candidate and the internal state is comprimised. Thus breaking the Dual_EC_DRBG only requires using 32-bytes of output from the victims machine.

4.5 NSA and RSA

The fact that this backdoor exists does not prove that the NSA intentionally put it there. Even the Snowden documents do not specifically name this implmentation as being compromised. However there are a few pieces of evidence which sway the case. One, there is no proof in NIST 800-90A given for how the two starting points are derived (P, Q). Two, the NSA did pay the RSA company to use Dual_EC_DRBG as the default cryptographic algorithm in their toolkit [3]. Finally, a Canadian company filed a patent for an Elliptic Curve cryptographic scheme which had special keys for law enforcement to bypass privacy [1]. Therefore, there is strong conjecture that the NSA did intend to use this to comprimise privacy. Regardless, given the fact that this exploit is known, the NSA would be remiss to not take advantage of the exploit.

5 Conclusion

Private communication and secure transactions are essential in today's life. In the USA, citizen's privacy has come into question in the revelation of the Snowden documents. The Dual_EC_DRBG is a good example of the kind of technology out there currently effecting privacy. There is now strong evidence that government agencies are possibly weakening privacy within the United States. Hopefully, in light of recent revelations, these citizens will be more aware of potential violations of civil liberties.

A Dual_EC_DRBG Proof of Concept

The following code is a proof of concept of the attack on Dual_EC_DRBG proposed by Dan Shumow and written for SageMathCloud™. The most updated version is available on github at `https://github.com/phillwilt/Dual_EC_DRBG`. tlatex

```
from multiprocessing import Process;

#bitmask
mask = (2**(30*8) - 1)

def gen_candidates(output):
    """
    Given a 30-byte block of output generate a list of potential 32-byte numbers
    """
    print "Generating Candidates";
    S = [];
    max_i = 2**16;

    #brute force find missing 2 bytes
    for i in range(0,max_i):
        sh = i << (30*8);
        x = sh | output;
        z = Mod(x^3-3*x + b, prime256);
        if(z.is_square()):
            y = z.sqrt();
            S.append([x,y]);
    return S;


def is_on_curve(point, curve):
    """
    Checks to see if the point is on the curve.
    """
    try:
        p = curve(point);
        return true;
    except:
        return false;


def test_match(e,Q,curve,pt_list,next_output):
    """
    Given next 2 bytes of output from a Dual_EC_PRNG instance, find the internal
```

```
    Params:
        e - d^1 (mod p_ord)
        Q - e*Q = P
        curve - the curve
        pt_list - list of potential points
        output - next 2 bytes of output from a Dual_EC_PRNG instance
    """
    print "Testing matches: " + str(len(pt_list)) + '\n';

    for p in pt_list:
        if(is_on_curve(p, E)):
            A = curve(p); # A = t = r*Q

            s1 = (e*A)[0].lift(); # x-coord: e*A = e*(r0*Q) = e*(r0*d*P) = r0*P
            r1 = (s1*P)[0].lift(); # x-coord: s1 * P = r1
            t1 = (r1*Q)[0].lift(); #x-coord: r1 * Q

            pred = t1 & mask; #throw away 16 Most Sig Bits
            test_pred = pred >> (28*8); #get 2 Most Sig

            if(test_pred == next_output):
                print "Match: ",pred;
                print "A: ", A

def predict_next(output,next_output, curve):
    """
    Given output, finds internal state of a Dual_EC_PRNG

    output - 30-byte output block
    next_output - next 30-byte output block
    curve - the curve
    """
    # we only need 2 bytes of Dual_EC_PRNG output to determine state
    output_test = (next_output >> (28*8))

    #Brute Force Generate Missing Point Data
    print "Brute Force Generation";
    pt_list = gen_candidates(output);
    print "Total Potential Matches: ", len(pt_list);

    procs = [];
    jump = 1000;

    #Test for matches
    for i in range(0, len(pt_list), jump):
        proc = Process(target=test_match, args=(e,Q,curve,pt_list[i:i + jump],ou
```

```
        procs.append(proc);
        proc.start();


    for proc in procs:
        if(proc.is_alive()):
            proc.join()



def dual_ec_gen(curve, P, Q, length, seed=None):
    """
    Dual_EC_DRNG instance

    Params:
        curve - the curve
        P - point P
        Q - point Q
        length - quantity of numbers to generate
        seed - initial seed (could be an internal state of a comprimised instanc
    """
    rand_list = [];
    # random initial seed
    if(seed == None):
        rand = floor((2**16-1)*random());
        s = int(rand);
    else:
        s = seed;

    for i in range(length):
            r = (s*P)[0].lift();
            s = (r*P)[0].lift();
            t = (r*Q)[0].lift();

            rand = t & mask;
            rand_list.append(rand);

    return rand_list;


#Curve P-256 from NIST SP800-90
prime256 = 115792089210356248762697446949407573530086143415290314195533631308867
b = 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b;
E = EllipticCurve(GF(prime256), [0,0,0,-3,b]); #NIST y^2= x^3- 3x + b (mod p)

Px = 0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296;
```

```
Py = 0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5;
P = E(Px,Py);

#Original Q
#Qx = 0xc97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c59852018192;
#Qy = 0xb28ef557ba31dfcbdd21ac46e2a91e3c304f44cb87058ada2cb815151e610046;

#Backdoor Q
d = 13;
Q = d*P;

p_ord = P.additive_order();

e = inverse_mod(d, p_ord); # find e = d^-1 (mod p_ord)
#print "e*d: ", Mod(e*d,p_ord); # 1

#print "P: ", P
#print "e*Q: ", e*Q

# Spin up a generator
rands = dual_ec_gen(E,P,Q,4);
print rands;

# Drop the hammer
%time q = predict_next(rands[1], rands[2], E);
```

References

[1] Omar El Akkad. The strange connection between the nsa and an ontario tech firm. http://www.theglobeandmail.com/technology/business-technology/the-strange-connection-between-the-nsa-and-an-ontario-tech-firm/article16402341. Accessed: 2014-02-12.

[2] Niels Ferguson Dan Shumow. On the possibility of a back door in the nist sp800-90 dual ec prng. http://rump2007.cr.yp.to/15-shumow.pdf. Accessed: 2014-03-10.

[3] Joseph Menn. Exclusive: Secret contract tied nsa and security industry pioneer. http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220. Accessed: 2014-03-10.

[4] William Stein. *Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach*. Springer, 2009.

[5] Nick Sullivan. A (relatively easy to understand) primer on elliptic curve cryptography. http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography. Accessed: 2014-02-12.