Weak Curves In Elliptic Curve Cryptography

Peter Novotney

March 2010

Abstract

Certain choices of elliptic curves and/or underlying fields reduce the security of an elliptical curve cryptosystem by reducing the difficulty of the ECDLP for that curve. In this paper I describe some properties of an elliptical curve that reduce the security in this manner, as well as a discussion of the attacks that cause these weaknesses. Specifically the Pohlig-Hellman attack and Smart's attack against curves with a Trace of Frobenius of 1 . Finally one of the recommended NIST curves is analyzed to see how resistant is would be to these attacks.

1 Elliptic Curves

First a brief refresh on the key points of elliptic curves, for more info see [Han04] [Sil86] [Ste08] . In its more general form, an Elliptic Curve is a curve defined by an equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

over some field F. In this case $E(F)$ defines a set of points that satisfy the elliptic equation in the field F. In notation $E(F) = \{(x, y) \in F^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$.
However, if the characteristic of the field is > 3 [Han04] then we can simplify the curve to

$$y^2 = x^3 + ax + b$$

and the set $E(F)$ becomes $E(F) = \{(x, y) \in F^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$.

1.1 Group Operation

We can create a group operation + over the set $E(F)$ defined as (Taken from [Ste08])

Given $P_1, P_2 \in E(F_p)$ we compute $R = P_1 + P_2$ as:

1. If $P_1 = \mathcal{O}$ then $R = P_2$ if $P_2 = \mathcal{O}$ then $R = P_1$ and terminate. Otherwise $(x_i, y_i) = P_i$

2. If $x_1 = x_2$ and $y_1 = -y_2$ then $R = \mathcal{O}$ and terminate.

3. Set

$$\lambda = \begin{cases} \dfrac{3x_1^2 + a}{2y_1}, & if P_1 = P_2 \\ \dfrac{y_1 - y_2}{x_1 - x_2}, & otherwise \end{cases}$$

4. Then $R = (\lambda^2 - x_1 - x_2, -\lambda x_3 - v)$, where $v = y_1 - \lambda x_1$ and $x_3 = \lambda^2 - x_1 - x_2$

The order or this group is represented by $\#E(F_p)$ and it is an important property of the curve used many times in the attacks described in this paper.

1.2 Choice of Field

When creating a cryptosystem based on elliptic curves one must choose which field the curve will be taken over. The popular choices [NIST] are curves over prime fields

$$E(F_p) = \{(x, y) : y^2 = x^3 + ax + b, x, y, a, b \in F_p\} \cup \{\mathcal{O}\}$$

or curves over binary fields

$$E(F_{2^m}) = \{(x, y) : y^2 + xy = x^3 + ax^2 + b, x, y, a, b \in F_{2^m}\} \cup \{\mathcal{O}\}$$

In this paper we will exclusively consider attack against curves over prime fields, but multiple attacks against curves over binary fields exist as well.

1.3 The Elliptic Curve Discrete Logarithm Problem

The security of Elliptic Curve Cryptosystems relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). The ECDLP is as follows:

For two points in an elliptic curve $Q, P \in E(F_p)$ such that $Q = kP$, compute k.

In some sources this is written as $k = \log_P Q$. The fastest algorithm to compute k currently is a combination of the Pohlig-Hellman attack described below, and the Pollard Rho Algorithm, which I do not describe here, but a detailed description can be found in [Han04].

2 Attacks on Weak Curves

There are multiple classes of elliptic curves or underlying fields that reduce the work needed to solve the ECDLP for a curve in that class. Here I cover two such possibilities of weak curves. One where $E(F_p)$ does not have sufficiently large prime subgroups and is subject to the Pohlig-Hellman attack, and another where $\#E(F_p) = p$ allowing an attack found by Smart in [Sma97].

2.1 Pohlig-Hellman Attack

The Pohlig-Hellman attack was first described in [Poh77], however here I will be using a description that follows from [Han04], as it seems simpler to understand. The Pohlig-Hellman attack simplifies the problem of solving the ECDLP in $E(F_p)$ to solving the ECDLP in the prime subgroups of $\langle P \rangle$, the subgroup generate by P.

First let n be the order of the subgroup generated by P, so $n = \#\langle P \rangle$. Now take the prime factorization of $n = \mathrm{p}_1^{\mathrm{e}_1} * \mathrm{p}_2^{\mathrm{e}_2} * \ldots * \mathrm{p}_{\mathrm{r}}^{\mathrm{e}_{\mathrm{r}}}$. We want to find $k_i \equiv k \ (mod\ p_i^{e_i})$ for each prime in the factorization and we do this by representing $k_i$as a base $p_i$number such that $k_i = z_0 + z_1 p_i + z_2 p_i^2 + \ldots + z_{e-1} p_i^{e-1}$ and then compute each z_j in sequence.

To do this let $P_0 = \frac{n}{p_i} P$ and $Q_0 = \frac{n}{p_i} Q$. P_0has order p_i since $p_i P_0 = \frac{p_i n}{p_i} P = nP$. By massaging the equation a bit we get

$$Q_0 = \frac{n}{p_i} Q = \frac{n}{p_i}(lP) = l\left(\frac{n}{p_i} P\right) = lP_0$$

and since the order of $P_0$is p_i and z_0 is the first digit of a base p_i number we have $lP_0 = z_0 P_0 = Q_0$. Thus finding z_0 requires computing the ECDLP in $\langle P_0 \rangle$. By expanding this same argument (see [Han04]) we are able to get each z_j by solving $Q_j = z_j P_0$ in $\langle P_0 \rangle$ where

$$Q_j = \frac{n}{p_i^{j+1}}\left(Q - z_0 P - z_1 p_i P - z_2 p_i^2 P - \cdots - z_{j-1} p_i^{j-1} P\right)$$

This leaves us with a system of equations

$$k \equiv k_1 \ (mod\ p_1^{e_1})$$

$$k \equiv k_2 \ (mod\ p_2^{e_2})$$

$$\ldots$$

$$k \equiv k_r \ (mod\ p_r^{r})$$

We know we can solve this system via the Chinese Remainder Theorem since all of the prime factors are certainly co-prime to each other, and thus we retrieve k, the solution to the discrete logarithm problem.

The following SAGE code [sage] computes the discrete logarithm of two points P,Q as described above (The discrete_log function in SAGE already uses Pohlig-Hellman, but this code is for illustrative purposes).

```
sage: def PolligHellman(P,Q):
sage:     zList = list()
sage:     conjList = list()
sage:     rootList = list()
sage:     n = P.order()
sage:     factorList = n.factor()
sage:     for facTuple in factorList:
sage:        P0 = (ZZ(n/facTuple[0]))*P
sage:        conjList.append(0)
sage:        rootList.append(facTuple[0]^facTuple[1])
sage:        for i in range(facTuple[1]):
sage:            Qpart = Q
sage:            for j in range(1,i+1):
sage:                Qpart = Qpart - (zList[j-1]*(facTuple[0]^(j-1))*P)
sage:            Qi = (ZZ(n/(facTuple[0]^(i+1))))*Qpart
sage:        zList.insert(i,discrete_log(Qi,P0,operation='+'))
sage:        conjList[-1] = conjList[-1] + zList[i]*(facTuple[0]^i)
sage:     return crt(conjList,rootList)

sage:     E = EllipticCurve(GF(7919), [234,75])
sage:     P = E.gens()[0]
sage:     Q = 2341*P
sage:     PolligHellman(P,Q)

2341
```

By choosing two large Elliptic groups, one with an order that has large prime factors, and another which doesn't, we can compare the performance of Pohlig-Hellman under these situations.

```
sage: #Group Order Compare SLOW
sage: m = 21345332
sage: p = 4516284508517
sage: E = EllipticCurve(GF(p), [7,1])
sage: Q = E.gens()[0]
sage: mQ = m*Q;
sage: print E.order().factor()
sage: time mRec = PolligHellman(Q,mQ)
sage: print mRec
11 * 13 * 31582419389
Time: CPU 49.14 s, Wall: 49.14 s
21345332

sage: #Group Order Compare FAST
sage: m = 21345332
sage: p = 4516284508517
sage: E = EllipticCurve(GF(p), [7,1])
sage: Q = E.gens()[0]
sage: mQ = m*Q;
sage: print E.order().factor()
sage: time mRec = PolligHellman(Q,mQ)
sage: print mRec
2^3 * 19 * 23 * 67 * 2089 * 18913
Time: CPU 0.16 s, Wall: 0.16 s
21345332
```

2.2 Smart's Attack where #E(Fp) = p

Smart in [Sma97] describes a linear time method of computing the ECDLP in curves over a field F_p such that $\#E(F_p) = p$, or in other words such that the trace of Frobenius is one, $t = p + 1 + \#E(F_p) = 1$. However describing this attack first requires some additional background.

2.2.1 Lifts and Hensel's Lemma

Suppose we have a polynomial $f(X) \in \mathbb{Z}[X]$ and we know a x such that $f(x) \equiv 0 \ (mod\ p)$ and we want to find an x' such that $f(x') \equiv 0 \ (mod\ p^2)$ and $x' \equiv x \ (mod\ p)$. We can achieve this by using what's known as Hensel's Lemma, which given a root of f modulo p^s computes a root of f modulo p^{s+1}. Hensel's Lemma is as follows (see [Bak10][Theorem 1.33] for a proof):

> *For $f(X) \in \mathbb{Z}[X]$ let x be a root of f modulo p^s and let $f'(x)$ be invertible modulo p and let that inverse be u such that $uf'(x) \equiv 1 \ (mod\ p)$. Let x' be*
>
> $$x' = x - uf'(x)$$
>
> *then $x' \equiv x \ (mod\ p^s)$ and $f(x') \equiv 0 \ (mod\ p^{s+1})$.*

Here x' is referred to as a lift of x modulo p^{s+1}. We will be using this theorem to lift element from F_p to the field of p-adic numbers Q_p described next.

2.2.2 P-adic Numbers

A p-adic number can be represented by an infinite series with the form

$$c_{-n}p^{-n} + \ \ldots + \ c_0 + c_1 p + \cdots + \ c_m p^m + \ \ldots \text{ [Lep04]}$$

The field of p-adic numbers are represented by Q_p, and the number that have no negative powers of p (i.e $c_i = 0 \ for \ i < 0$) are known as the p-adic integers and are represented as $\mathbb{Z}_p$. We can define elliptic curves over the field of p-adic numbers and we use the lifts described above to lift our points of interest into the elliptic curve over Q_p. This allows us to reduce the ECDLP to the group $p\mathbb{Z}_p$ where it is easily computable as we'll see in 2.2.4.

2.2.3 Curve Reduction Modulo P

Another important component of this attack is the reduction of an elliptic curve modulo p. This basically amounts to taking the coefficients and points of a curve and taking its congruency module some prime p.

Let $E(Q_p)$ be an elliptic curve over the p-adic field, we create a curve over F_p by reducing the coefficients of the curve $E(Q_p) : y^2 = x^3 + ax + b$ modulo p, such that $E(F_p) : y^2 = x^3 + \tilde{a}x + \tilde{b}$. To be complete we should check ensure that this new curve isn't singular by checking that the discriminate is non-zero, but for our purposes here I'm just going assume it is non-singular.

Now we follow a similar process to map a point $P = (x, y) \in E(Q_p)$ to another point $\tilde{P} = (\tilde{x}, \tilde{y}) \in E(F_p)$ where $\tilde{x} = x \ (mod\ p)$ and $\tilde{y} = y \ (mod\ p)$.

This mapping is a group homomorphism from $E(Q_p)$ to $E(F_p)$ [Sil86], and let $E_1(Q_p)$ be defined as in [Sil86] as the kernel of this homomorphism, so $E_1(Q_p)$ contains all the points in $E(Q_p)$ that reduce to the point at infinity in $E(F_p)$. Chapter VII in [Sil86] contains further discussion of the Reduction Modulo P.

2.2.3 P-adic Elliptic Logarithm

The P-adic Elliptic Logarithm ψ_p provides a isomorphism from $E_1(Q_p)$ to $p\mathbb{Z}_p$. For a full discussion of the derivation of this isomorphism see Chapters IV and VII in [Sil86]. The derivation is rather long, so for our purposes I will simply show how to compute ψ_p by the method described in [Lep04].

For a point $S \in E_1(Q_p)$ we compute

$$\psi_p(S) = -\frac{x(S)}{y(S)}$$

By reducing the problem of finding k where $Q = kP$to $p\mathbb{Z}_p$we can directly compute k as we see in the description of the attack itself.

2.2.4 The Attack

We will now go over the attack as presented in [Sma97] using the ideas developed above. At the end of this section there is SAGE code [sage] that runs this attack.

First recall that we are trying to find k such that $Q = kP$ where $Q, P \in E(F_p)$ and $\#E(F_p) = p$. Our first step is to lift these points to $E(Q_p)$ to get two new points P', Q'. We do this by setting the x component of P' equal to the x component of P. We then use Hensel's Lemma described above to compute y in Q_p by using the curve equation with x fixed. We know that $Q = kP$ in $E(F_p)$ so thus the value $Q' - kP'$ in $E(Q_p)$ goes to the point at infinity by the Reduction Modulo P map and is thus in the kernel of that homomorphism.

$$Q' - kP' \in E_1(Q_p)$$

Now we rely on the fact that the order of $E(F_p)$ is p, which ensures that multiplying any element in $E(Q_p)$ by p maps the element into $E_1(Q_p)$ since for any point $R \in E(Q_p)$ the point pR will map via Reduction Modulo P to $\mathcal{O}$ in $E(F_p)$. So multiply through by p and we get

$$pQ' - k(pP') \in E_2(Q_p)$$

with $pQ' \in E_1(Q_p)$ and $pP' \in E_1(Q_p)$. We can now apply the P-Adic Elliptic Log to get

$$\psi_p(pQ') - k\psi_p(pP') \in p\mathbb{Z}_p$$

and thus

$$k = \frac{\psi_p(pQ')}{\psi_p(pP')}$$

and then reduce k modulo p to return to F_p solving the ECDLP.

The SAGE code for the Hensel Lift on a point P is:

```
sage: def HenselLift(P,p,prec):
sage:     E = P.curve()
sage:     Eq = E.change_ring(QQ)
sage:     Ep = Eq.change_ring(Qp(p,prec))
sage:     x_P,y_P = P.xy()
sage:     x_lift = ZZ(x_P)
sage:     y_lift = ZZ(y_P)
sage:     x, y, a1, a2, a3, a4, a6 = var('x,y,a1,a2,a3,a4,a6')
sage:     f(a1,a2,a3,a4,a6,x,y) = y^2 + a1*x*y + a3*y - x^3 - a2*x^2 - a4*x - a6
sage:     g(y) = f(ZZ(Eq.a1()),ZZ(Eq.a2()),ZZ(Eq.a3()),ZZ(Eq.a4()),ZZ(Eq.a6()),ZZ(x_P),y)
sage:     gDiff = g.diff()
sage:     for i in range(1,prec):
sage:         uInv = ZZ(gDiff(y=y_lift))
sage:         u = uInv.inverse_mod(p^i)
sage:         y_lift = y_lift - u*g(y_lift)
sage:         y_lift = ZZ(Mod(y_lift,p^(i+1)))
sage:     y_lift = y_lift+O(p^prec)
sage:     return Ep([x_lift,y_lift])
```

and the code for the complete attack is:

```
sage: def SmartAttack(P,Q,p,prec):
sage:     E = P.curve()
sage:     Eqq = E.change_ring(QQ)
sage:     Eqp = Eqq.change_ring(Qp(p,prec))
sage:
sage:     P_Qp = HenselLift(P,p,prec)
sage:     Q_Qp = HenselLift(Q,p,prec)
sage:
sage:     p_times_P = p*P_Qp
sage:     p_times_Q=p*Q_Qp
sage:
sage:     x_P,y_P = p_times_P.xy()
sage:     x_Q,y_Q = p_times_Q.xy()
sage:
sage:     phi_P = -(x_P/y_P)
sage:     phi_Q = -(x_Q/y_Q)

sage:     k = phi_Q/phi_P
sage:     k = Mod(k,p)
sage:     return k
sage:
sage: E = EllipticCurve(GF(43), [0,-4,0,-128,-432])
sage: print E.order()
sage: P=E([0,16])
sage: Q=39*P
sage: SmartAttack(P,Q,43,8)

43 (Order of E)
39 ( k )
```

3 NIST Recommended Curve

The US government via the National Institute of Standards and Technology recommends elliptic curves for use in its own cryptographic systems [NIST]. Let's check if one of those curves is resistant to the attack described above.

The curves over prime fields in the recommendations are all of the form $y^2 = x^3 - 3x + b$ where b is some large integer. Taking the defined curve for the 192-bit case we have

$$p = 6277101735386680763835789423207666416083908700390324961279$$

$$b = 2455155546008943817740293915197451784769108058161191238065$$

In SAGE we have:

```
sage: E = EllipticCurve(GF(6277101735386680763835789423207666416083908700390324961279), \
[-3,2455155546008943817740293915197451784769108058161191238065])
sage: E.order()
6277101735386680763835789423176059013767194773182842284081

sage: E.order().factor()
6277101735386680763835789423176059013767194773182842284081
```

The order of the group does not match p so we cannot apply Smart's Attack, and the order is itself prime thus preventing the Pohlig-Hellman attack from being efficient. This shouldn't be too surprising that NIST chose a curve and field preventing some of the most well know attacks poorly chosen curves.

4 Conclusion

In this paper we have covered two attacks against improperly chosen elliptic curves and their underlying fields, but this by no means an extensive list. There are multiple other attacks against curve over prime fields as well as attacks against curves over binary fields of the form $E(F_{2^m})$, which we didn't touch on at in this paper. Suffice to say, anyone implementing an elliptic curve cryptosystem needs to be aware of these potentially harmful curve choices and correctly mitigate them in their system.

5 References

[Bak10] A. Baker, *An Introduction to p-adic Numbers and p-adic Analysis*, October 2010. http://www.maths.gla.ac.uk/~ajb/dvi-ps/padicnotes.pdf

[Blu] Antonia W. Bluher, *A Leisurely Introduction to Formal Groups and Elliptic Curves*, http://www.math.uiuc.edu/Algebraic-Number-Theory/0076/FmGp.ps.gz

[Han04] D. Hankerson, S. Vanstone, A. Menezes, *Guide to elliptic curve cryptography*, Springer-Verlag, 2004.

[Lep04] F. Leprevost, J. Monnerat, S. Varrette, S. Vaudenay, *Generating anomalous elliptic curves*, 2004. http://lasecwww.epfl.ch/pub/lasec/doc/LMVV05.pdf

[Poh77] S. Pohlig, M. Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance,* 1977. http://www.ee.stanford.edu/~hellman/publications/28.pdf

[NIST] National Institute of Standards and Technology, *Recommended Elliptic Curves for Federal Government Use*, July 1999. http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.doc

[Sma97] N. P. Smart, *The discrete logarithm problem on elliptic curves of trace one*, October 1997. http://www.hpl.hp.com/techreports/97/HPL-97-128.html

[Sil86] J. H. Silverman, *The arithmetic of elliptic curves*, Springer-Verlag, 1986.

[Sage] William A. Stein et al., *Sage Mathematics Software (Version 4.3).* The Sage Development Team, 2009. http://www.sagemath.org.

[Ste08] William A. Stein, *Elementry Number Theory: Primes, Congruences, and Secrets.* Springer-Verlag, 2008.