Elliptic Curve Cryptography Summer Project

Ben Ward

September 6, 2015

1 The Discrete Log Problem

A discrete logarithm is an integer x solving $g^x = h$, where g, h are elements of some finite group (note that x is only defined modulo the order of the group). The process of computing these logarithms is known as the discrete log problem (DLP).

The security of the RSA cryptosystem revolves around the presumed difficulty of integer factorisation. The algorithms used in ECC rely instead on the difficulty of the DLP, which depends heavily on the choice of group. For example, in the group of the integers mod p under addition, solving the DLP is simply solving a linear congruence. However, in an elliptic curve group there are no known techniques for solving the DLP beyond some simple ones described below (this is known as the ECDLP).

1.1 Some Techniques to Solve the DLP

We want techniques for finding x which require fewer than $N = o(g)$ operations (an obvious upper bound).

1.1.1 Babystep-Giantstep

A collision algorithm. Let $n = 1 + \lfloor N \rfloor$. Compute and store the list e, g, g^2, ..., g^n. Then compute the list h, hg^{-n}, hg^{-2n}, ... term by term until an element present in the first list is found. We have $g^i = hg^{-jn}$ for some i, j, so $g^{i+jn} = h$.

It is easy to show that a collision will always be found before n elements of the second list have been computed, so this algorithm is $O(\sqrt{N})$. Note that it is independent of the choice of group (in particular, it is tricky to get better than this for the ECDLP).

1.1.2 Pohlig-Hellman

This algorithm takes advantage of the fact that discrete logarithms are defined modulo N to use the chinese remainder theorem to simplify the problem. Deter-

mine the prime factorisation $N = p_1^{k_1}...p_n^{k_n}$. Then solve the discrete logarithms

$$\left(g^{N/p_1^{k_1}}\right)^{x_1} = h^{N/p_1^{k_1}}$$

$$\vdots$$

$$\left(g^{N/p_n^{k_n}}\right)^{x_n} = h^{N/p_n^{k_n}}$$

which can be done efficiently for smooth N since in each case the element has order $p_i^{k_i}$ - using a refinement of the Babystep-Giantstep algorithm detailed above this can be done in $O(k_i\sqrt{p_i})$ operations.

Finally, use the chinese remainder theorem to solve the congruences $x \equiv x_1(p_1^{k_1})$, ..., $x \equiv x_n(p_n^{k_n})$. It can be shown that the resultant x satisfies $g^x = h$.

1.1.3 Index Calculus

The above methods are 'generic' in the sense that they can be used for any group. The index calculus is an example of a method for solving the DLP in a particular group - $\mathbb{F}_p$. In contrast to the previous methods however, it is a subexponential algorithm.

To solve $g^x = h \pmod p$, we first choose a value B and solve the DLP for all primes $l \leq B$. This can be done efficiently by choosing a random j and computing $g_j = g^j \pmod p$ - if g_j is B-smooth, then it can be factored as

$$g_j = \prod_{l \leq B} l^{u_l(j)}$$

yielding the relation

$$j = \sum_{l \leq B} u_l(j) \log_g l$$

Given $\pi(B)$ distinct relations, we can use Gaussian elimination along with the CRT to solve for the logarithms of the primes.

Finally, seek a value of j such that $hg^{-k} \pmod p$ is B-smooth. Then

$$\log_g h = k + \sum_{l \leq B} e_l \log_g l$$

where e_l is the exponent of l in the factorisation of hg^{-k}.

Although somewhat complex, this algorithm can be implemented very efficiently (for example, the $\pi(B)$ relations required can be found simultaneously with parallel programming). It is of particular interest that elliptic curve groups do not have 'prime' elements, so the method cannot be used there.

1.2 ElGamal Encryption

The ElGamal encryption system is a form of public key cryptography which uses the DLP for security. It can be used over any group - in later sections we

will use elliptic curve groups.

1.2.1 Key generation

Pick $g \in G$ of order q, and $0 < a < q - 1$. Let $A = g^x$. Publish A along with g, q and G as the public key, retain a as the private key.

1.2.2 Encryption

Pick a random $0 < y < q - 1$. Given a message m, encrypt m as the pair (g^y, $A^y m$).

1.2.3 Decryption

Given the ciphertext (c_1, c_2), compute the quantity $(c_1^a)^{-1} c_2$.

2 Elliptic Curves

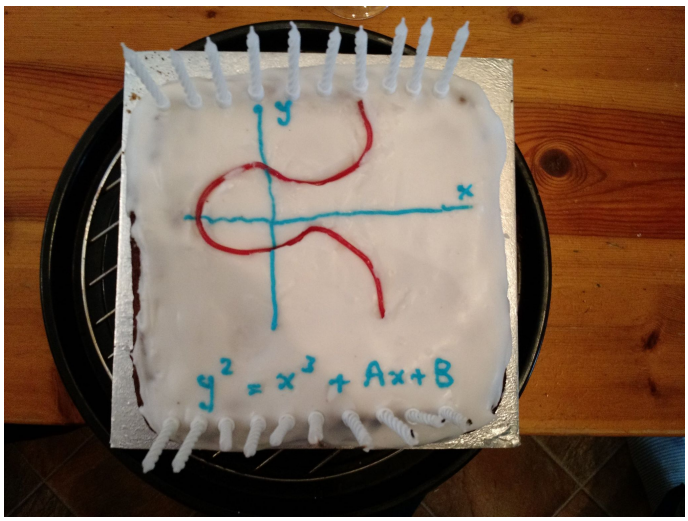

Figure 1: An elliptic curve

Informally, an elliptic curve is the set of solutions to an equation of the form $y^2 = x^3 + Ax + B$ (note that this form is not general over fields of characteristic 2 or 3) over some field, where $4A^3 + 27B^2 \neq 0$. The points on a curve can be made into a group (with the addition of a point at infinity) by a geometric group law.

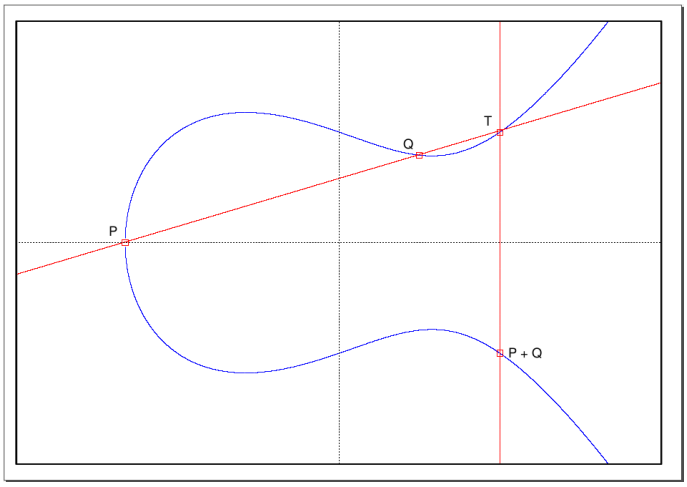

Figure 2: The group law

2.1 Elliptic Curves over Finite Fields

Over finite fields, elliptic curve groups are also finite so become suitable for cryptographic use - e.g. with ElGamal as descibed above. Finding the order of these groups is difficult - an important theorem of Hasse tells us that

$$|E(\mathbb{F}_p)| = p + 1 - t_p$$

with t_p satisfying $|t_p| \leq 2\sqrt{p}$. Further, let α, β be the roots of the polynomial

$$z^2 - t_p z + p$$

Then

$$|E(\mathbb{F}_{p^k})| = p^k + 1 - \alpha^k - \beta^k$$

(this will be useful later).

3 Some Attacks on the ECDLP

3.1 Curves of Bad Reduction

We specified above that curves should have discriminant $\neq 0$. Given a curve with $4A^3 + 27B^2 = 0$, there are two possibilities: if $A = 0$ then the curve has a cusp and the map

$$E_{ns} \to \mathbb{F}_q^+, \quad (x, y) \to \frac{x}{y}$$

is a group isomorphism. If $A \neq 0$ then the curve has a node and the map

$$E_{ns} \to \mathbb{F}_q^*, \quad (x, y) \to \frac{y - \alpha_1 x - \beta_1}{y - \alpha_2 x - \beta_2}$$

where

$$y = \alpha_1 x + \beta_1 \text{ and } y = \alpha_2 x + \beta_2$$

are the distinct tangents to E at the node is a group isomorphism. We can use this fact to reduce the ECDLP to the DLP in much easier groups - recall that the first case is particularly bad as the DLP in the additive group of a field of prime order is simply solving a linear congruence!

3.1.1 Example: MysteryTwister Challenge

In the Mystery Twister challenge 'Not-so-secret message from Malawi - Part II' we are given the curve

$$\begin{aligned} y^2 = x^3 &+ 963218343336110113016174981681596148974425738450x^2 \\ &+ 646590556709322492479302728449481955110774251117x \\ &+ 137346957280116216304262434997226868149786911218 \end{aligned}$$

over the finite field $\mathbb{F}_p$, where

$$p = 193779545873623981383940726165651897912330459621 7$$

At first glance nothing looks wrong here. However, reducing the curve to Weierstrass form by translating

$$x \to x + 321072781112036704338724993893865382991475246150$$

shows us that the curve is isomorphic to

$$y^2 = x^3 - 387559091747247962767881452331303795824660919243 4x$$

which has a cusp at $(0, 0)$ allowing us to easily solve the challenge.

3.2 Supersingular Curves and the Weil Pairing

The Weil pairing, denoted e_m, is a bilinear map from the m-torsion subgroup of an elliptic curve to the m-th roots of unity. Crucially, it is efficient to compute.

The MOV algorithm uses the Weil pairing to reduce the ECDLP for $E(\mathbb{F}_p)$ to the DLP for $\mathbb{F}_{p^k}$, where k is the embedding degree of the curve (the embedding degree is essentially the least k for which this is possible). Supersingular elliptic curves, where $|E(\mathbb{F}_p)| = p+1$, are a class of curve with k usually ≤ 2 and always ≤ 6, so are especially susceptible to this technique.

To solve the ECDLP $P^n = Q$, where P, Q have order l, in $E(\mathbb{F}_p)$ with the MOV algorithm:

- Find $N = |E(\mathbb{F}_{p^k})|$ (see section 3.1)

- Choose a random point $T \in E(\mathbb{F}_{p^k}) \setminus E(\mathbb{F}_p)$ such that $T' = (N/l)T \neq \mathcal{O}$: this point then has order l
- Compute the Weil pairings

$$\alpha = e_l(P, T') \qquad \text{and} \qquad \beta = e_l(Q, T')$$

  note that these are elements of $\mathbb{F}_{p^k}$.
- Solve the DLP $\alpha^n = \beta$. Then $P^n = Q$

A Some Code Samples

Listing 1: Solving the DLP in $\mathbb{F}_{11251}$

```
>>> from discrete_log import pohlig_hellman
>>> from groups.intmodp import IntModP
>>> g = IntModP(5448, 11251)
>>> h = IntModP(6909, 11251)
>>> pohlig_hellman(g, h)
3636
>>> pow(5448, 3636, 11251)
6909
```

Listing 2: Arithmetic in an elliptic curve group

```
>>> f = FiniteField(13,1)
>>> c = EllipticCurve((3, 8), f)
>>> c.order()
9
>>> 3*c.point(1,5)
(9, 7)
>>> c.point(9,7).order()
3
>>> c.point(1, 5) + c.point(9, 7)
(12, 2)
```

Listing 3: Computing the Weil pairing

```
>>> f = FiniteField(5,2) # Finite field of order 5^2
>>> c = EllipticCurve((0, 1), f) # Elliptic curve y^2 = x^3 + 1 over f
>>> p = c.point(2, 2) # Point (2, 2)
>>> q = c.point([0,1], [2,2]) # Point (x, 2+2x)
>>> c.weil_pairing(p, q)
2+4x
>>> c.weil_pairing(p, q)**6
1
```

References

[1] Jeffrey Hoffstein, *An Introduction to Mathematical Cryptography*, Springer

[2] Joseph Silverman, *The Arithmetic of Elliptic Curves*, Springer