Elliptic Curve Cryptography

Martin Leslie
40575063
Advanced Combinatorics

June 5, 2006

1 Introduction

Elliptic curve cryptography is a public key cryptosystem that relies on the believed difficulty of the elliptic curve discrete logarithm for its security. It was first suggested in 1985 by N. Koblitz and V. Miller and is becoming accepted as an alternative to cryptosystems such as RSA and ElGamal over finite fields.

We will give a quick introduction to public key cryptography, the arithmetic of elliptic curves, and ElGamal over elliptic curves over finite fields. We will then discuss attacks on the discrete log problem and mention digital signature algorithms based on elliptic curves.

It should be stressed that this paper is merely an introduction to the concepts of elliptic curve cryptography. The algorithms and examples given are chosen for their explanatory power, not their security. Anyone wishing to implement elliptic curve cryptography should look at the standards that have emerged over the last few years.

2 Public key cryptography

Cryptography is a way for two people, referred to as Alice (A) and Bob (B), to communicate secretly over an insecure channel without an opponent, Oscar (O), understanding what is being said. We take our definition of a cryptosystem from [3].

Definition 2.1. A cryptosystem is a five–tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where the following conditions are satisfied.

1. $\mathcal{P}$ is a finite set of possible plaintexts;
2. $\mathcal{C}$ is a finite set of possible ciphertexts;
3. $\mathcal{K}$, the keyspace, is a finite set of possible keys;
4. For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext $x \in \mathcal{P}$.

From the time that cryptography was first used thousands of years ago until about 1975 every cryptosystem was a *private key* system. This means that before two parties could communicate they would have to share a secret key through a secure channel. Anyone with this key could both encrypt and decrypt messages. This meant that men in black suits with briefcases full of codes were travelling between major banks and governments to allow secret messages to be sent.

It was to solve this *key distribution problem* that the first public key system was proposed. In a *public key* or *asymmetric* cryptosystem there are two keys. The *public key*, which is published in a directory and allows encryption, and the *private key* which is kept secret and allows decryption.

A good way to see the difference between private and public key cryptosystem in a non mathematical context is to imagine physical padlocks and keys. In a private key system the person wishing to send a message must first send the key securely to the intended recipient then can at a later time put a message in a box then padlock it and send it. The recipient can then unlock the box and read the message.

In a public key system we can imagine making thousands of copies of the padlock (the public "key") and making them freely available. Anyone who wishes to can then put their message in a box, padlock it and send it to us. We can then unlock it with our (private) key.

The first public key cryptosystem was invented by Diffie and Helman in 1976. We describe it over a general finite group.

Algorithm 2.2 (Diffie-Hellman key exchange)**.** This algorithm allows two people, A and B, to generate a shared piece of secret information over an insecure communications channel.

1. (Setup) A and B publicly select a finite group G and an element $\alpha \in G$.
2. A generates a random integer a, computes α^a in G, and transmits α^a to B over a public communications channel.
3. B generates a random integer b, computes α^b in G and transmits α^b to A over the same channel.
4. A receives α^b and computes $(\alpha^b)^a$.
5. B receives α^a and computes $(\alpha^a)^b$.

Now A and B both know the element α^{ab} which can be used as a private key for further communication.

If an opponent O listens in on this process he knows G, α, α^a and α^b. Finding α^{ab} from this information is the *Diffie-Hellman problem.* If O could find a from α and α^a then he could simply compute $(\alpha^b)^a$ and solve the Diffie–Hellman problem.

Definition 2.3 (Discrete Logarithm problem)**.** Given a group G, an $\alpha \in G$ and α^a compute a.

No efficient algorithm is known for computing discrete logarithms. The naive method is to compute α^k for $k = 1, 2, 3, \ldots$ until $\alpha^k = \alpha^a$. Then $a = k$

is a solution. This process is highly inefficient and there are better methods available. We shall discuss these in our section on attacks on elliptic curve cryptosystems. If O could solve the discrete logarithm efficiently then he could solve the Diffie–Hellman problem efficiently. However there is no proof that these two problems are equivalent – there may be a way of solving Diffie-Hellman without solving discrete logarithms.

We now define a public key cryptosystem also based on the discrete logarithm system, ElGamal.

Algorithm 2.4 (ElGamal)**.** This algorithm allows two people to communicate messages secretly over an insecure communications channel.

1. (Setup) A finite cyclic group G of order n and generator $\alpha \in G$ are chosen. Each user picks a random integer $l \in \{0, 1, \ldots, n-1\}$ (the private key), and makes public α^l (the public key). We suppose that messages are elements of G and that user A wishes to send a message, m, to user B.
2. A generates a random integer $k \in \{0, 1, \ldots, n-1\}$ and computes α^k.
3. A looks up B's public key α^l and computes $(\alpha^l)^k$ then $m\alpha^{lk}$.
4. A sends to B the pair of group elements $(\alpha^k, m\alpha^{lk})$.
5. B computes $(m\alpha^{lk})((\alpha^k)^l)^{-1} = m\alpha^{lk}(\alpha^{lk})^{-1} = m$ and recovers the message.

An eavesdropper, O, would know α, α^l, α^k and $m\alpha^{lk}$. Being able to solve the discrete logarithm would allow the attacker to find l and then use the same process as B does to decrypt m. ElGamal encryption is known to be very susceptible to chosen ciphertext attacks, see [4]. However the Cramer–Shoup cryptosystem which is based on ElGamal is secure against chosen ciphertext attacks.

3 Finite field cryptography

Finite fields are a well understood mathematical topic and form the basis for many cryptographic protocols. Many of the algorithms used in elliptic curve cryptography are the same as used in finite field cryptography except instead of working over a finite field we work over points over an elliptic curve whose coordinates come from a finite field.

Recall some basic facts about finite fields.

Theorem 3.1. *For every prime p and integer n greater than or equal to 1 there exists a finite field, denoted $\mathbb{F}_{p^n}$, such that $|\mathbb{F}_{p^n}| = p^n$. The only finite fields are of this form. The finite field $\mathbb{F}_{p^n}$ is unique for a given p and n and the group of units $\mathbb{F}_{p^n}^\times$ is a cyclic group. For $n = 1, \mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$.*

Proof. Omitted. See any abstract algebra book. □

In our algorithms we will generally be working over $\mathbb{F}_q$ where either $q = p > 2$ or $q = 2^m$. The first case is simple to represent in a computer as $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$. A characteristic two finite field is also able to be efficiently represented in a computer as binary.

Cryptographic protocols like ElGamal are usually implemented over finite fields.

Example 3.2. We will carry out an ElGamal encryption and decryption with the group $G = \mathbb{F}_{11}^{\times} = \{1, 2, 3, \ldots, 10\} = \langle 2 \rangle$.

1. (Setup) We have $n = 10$ and $\alpha = 2$. User B picks randomly $b = 5$ then $\alpha^b = 2^5 = 10$. Have public key $(n, \alpha, \alpha^b) = (10, 2, 10)$ and private key $b = 5$.
2. User A chooses $k = 7$ and calculates $\alpha^k = 2^7 = 7$.
3. User A looks up $\alpha^b = 10$ and encodes message as $m = 3$ then calculates $m(\alpha^b)^k = 3 \times 10^7 = 8$.
4. User A sends $(\alpha^k, m\alpha^{bk}) = (7, 8)$.
5. User B calculates $m\alpha^{bk}(((\alpha^k)^b)^{-1} = 8(7^5)^{-1} = 3$.

Thus B receives the message, 3, as sent by A.

We can see that this process is relatively efficient given good algorithms for exponentiation and finding inverses in finite fields, which we do have. Obviously an example with such a small group is easily cracked, to achieve real world security n would have to be thousands of digits long.

We will eventually see a similar example with the group being a cyclic subgroup of an elliptic curve over a finite field. We will be able to compare the efficiency and also consider attacks on both cryptosystems.

4 Elliptic curves

An elliptic curve over a field K is a non-singular projective algebraic curve over K with genus 1 together with a given point, $\mathcal{O}$. Having given this high powered definition we will know proceed to ignore it and try to explain elliptic curves in a manner that is understandable (with some suspension of disbelief) by anyone who knows high school algebra and geometry.

We can understand an elliptic curve as all the points on the curve (given by the *Weirstrass equation*)

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

together with $\mathcal{O}$, the "point at infinity". We also require that the curve is non-singular, so has no cusps or self intersections. An example is shown in Figure 1, where we consider the point at infinity to be the point infinitely far to the top and bottom of the graph.

Note that if the field we are working over has characteristic not equal to two or three then we can make a change of variables (for example could complete the square with y's on the left hand side since could divide by two) to convert our Weirstrass equation to

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

Although this would make what is following simpler we would lose the ability to work over $\mathbb{F}_{2^m}$, which is important in the world of cryptography.

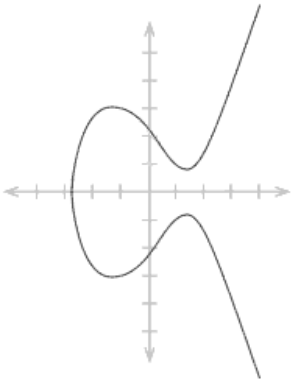

Figure 1: An elliptic curve, drawn over the reals.

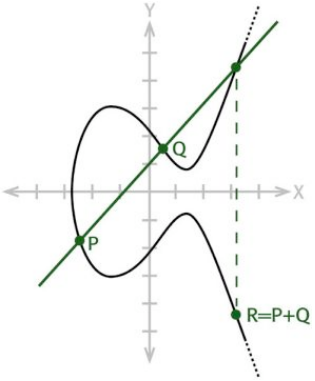

Figure 2: The group law on an elliptic curve.

We have already stated that an elliptic curve forms an abelian group and this isn't at all obvious from our definition. To define the group law consider two points, P and Q, on our elliptic curve and draw the line from P to Q until you hit the curve again. This forms another point on the curve. Now draw a line from the point at infinity, $\mathcal{O}$, through this new point. The point where this line intersects the elliptic curve again is $P + Q$.

This definition is perfectly general but if $P = Q$ or one of P and Q equals $\mathcal{O}$ then it takes some interpreting. Because we only have one point at infinity we can usually get away with considering $\mathcal{O}$ as a special case.

If we are trying to find $2P$, then P and Q are the same point. In this case we take the line between P and Q to be the tangent line at P and proceed in the same manner as above. If the line from P to Q doesn't intersect the curve anywhere on the finite plane (in our figures this means the line is vertical) then we say that it intersects the elliptic curve at $\mathcal{O}$ – this is why we needed to include this point on our curve. Then the line from $\mathcal{O}$ to $\mathcal{O}$, the "line at infinity", intersects the curve at $\mathcal{O}$. Thus $P + Q = \mathcal{O}$. If this all seems a bit quick – don't worry, we will soon provide explicit formulae for addition so you can simply take this as your definition.

Note that this definition is symmetrical, the line between P and Q is the

same as the line between Q and P so $P + Q = Q + P$. Also note that if we want to calculate $P + \mathcal{O}$ then the line between these two points is the same as the line between $\mathcal{O}$ and the point where the first line intersects the curve. This means $P + \mathcal{O} = P$ and $\mathcal{O}$ is the identity of our group.

From this definition of addition we can provide an explicit algorithm to add points on an elliptic curve. For us this is especially important as the geometric ideas don't make sense on the finite fields that we are interested in for cryptography. These formulae come from [1] where they are developed from the above ideas using conceptually simple but somewhat tedious coordinate geometry.

Algorithm 4.1. Let E be an elliptic curve given by

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

(a) If $P_0 = \mathcal{O}$ then $-\mathcal{O} = \mathcal{O}$. Otherwise let $P_0 = (x_0, y_0) \in E$. Then

$$-P_0 = (x_0, -y_0 - a_1x_0 - a_3).$$

(b) If one of P_1 or P_2 equals $\mathcal{O}$ then use $P + \mathcal{O} = \mathcal{O} + P = P$, otherwise let

$$P_1 + P_2 = P_3 \text{ with } P_i = (x_i, y_i) \in E.$$

If $x_1 = x_2$ and $y_1 + y_2 + a_1x_2 + a + 3 = 0$ then $P_1 + P_2 = \mathcal{O}$.
Otherwise, let

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},\ \nu = \frac{y_1x_2 - y_2x_1}{x_2 - x_1} \text{ if } x_1 \neq x_2;$$

$$\lambda = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} \text{ and}$$

$$\nu = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3} \text{ if } x_1 = x_2.$$

(So $y = \lambda x + \nu$ is the line through P_1 and P_2, or tangent to E if P_1=P_2.)
Then $P_3 = (x_3, y_3)$ where

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$$

$$y_3 = -(\lambda + a_1)x_3 - \nu - a_3.$$

With this definition of addition we can (almost) show that the points on an elliptic curve define a abelian group.

Theorem 4.2. *The addition law on an elliptic curve, E, as given in Algorithm 4.1, has the following properties:*

1. $P + \mathcal{O} = P$ *for all* $P \in E$.
2. $P + Q = Q + P$ *for all* $P, Q \in E$.
3. *Let* $P \in E$. *There is a point of* E, *denoted* $-P$, *so that*

$$P + (-P) = \mathcal{O}.$$

4. *Let $P, Q, R \in E$. Then*

$$(P + Q) + R = P + (Q + R).$$

In other words, the addition law makes E into an abelian group with identity $\mathcal{O}$. Furthermore:

5. *Suppose E is defined over K, that is, given by*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \text{ with } a_i \in K.$$

Then the points on the curve with coordinates in K

$$E(K) = \{(x, y) \in K^2 \colon y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$$

form a subgroup of E.

Proof. Associativity is the only hard part of this proof.

1. By definition.
2. The addition formulae given above are symmetrical. If you swap P_1 and P_2 then λ and ν remain the same.
3. $-P$ is defined above and has this property.
4. The three options are to: convince yourself geometrically, take three points on E and construct their sum in the two different ways. Alternatively you could take three general points and add them in the two different ways using the explicit addition formulae. The problem with this is you have to consider all the different cases depending on whether any of the points P, Q and R are equal.

   Probably the most enlightening proof is by the Riemann–Roch Theorem, as in [1]. In this proof it is obvious that the object we define is a group but it is not so obvious that it is an elliptic curve. However this would take us too far afield so we leave this as just a suggestion of a proof.
5. If $P_1, P_2 \in E(K)$ then we can see that $P_1 + P_2 = (x_3, y_3)$ where the coordinates are given by rational functions (quotients of polynomials) in the variables x_i, y_i, a_i, all of which are in the field K. This means that $x_3, y_3 \in K$, that is $P_1 + P_2 \in E(K)$. Therefore $E(K)$ is a subgroup of E.

□

Elliptic curves are fascinating objects but we only need to know enough to be able to understand their use in cryptography so we leave their general study here.

5 Elliptic curves over finite fields

We can consider an elliptic curve defined over a finite field, which will turn out to have a finite number of points, which we can in fact bound. Our first example comes from [2].

Example 5.1. Let E be the curve $y^2 = x^3 + x + 1$ over $\mathbb{F}_5$. To find solutions of this equation in $\mathbb{F}_5$, just consider $x = 0, 1, 2, 3, 4$ and take square roots to find the corresponding y's. We get

$$E = \{(0,1), (1,4), (2,1), (2,4), (3,1), (3,4), (4,2), (4,3), \mathcal{O}\}.$$

To do some additions we can use our algorithm. For example to find $(2,1) + (3,4)$, we note that $a_1 = a_3 = a_2 = 0$ and $a_4 = a_6 = 1$. Then

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-1}{3-2} = 3 \text{ and}$$

$$\nu = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1} = \frac{1 \times 3 - 4 \times 2}{3-2} = 3 - 8 = 0.$$

So $x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 = 2^2 - 2 - 3 = 4$ and $y_3 = -(\lambda + a_1)x_3 - \nu - a_3 = -3 \times 4 - 0 = 3$. So, finally $(2,1) + (3,4) = (4,3)$ which is inside our group E. More calculations, which are perhaps more appropriately done using a system such as PARI/GP [5], show that in fact E is cyclic, generated by $(0,1)$.

Probably the most useful result about elliptic curves over finite fields is

Theorem 5.2 (Hasse's theorem)**.** *Let E be an elliptic curve over the finite field $\mathbb{F}_q$. If $E(\mathbb{F}_q)$ has order N we have*

$$|N - (q+1)| \leq 2\sqrt{q}.$$

Proof. Omitted. See [2]. □

Hasse's theorem gives us a fairly tight bound on the number of points on an elliptic curve over a finite field. However we will see later that it is sometimes important to know exactly how many points there are. The most efficient current algorithm to find this is Schoof's point counting algorithm, invented in 1985 by Schoof. It can compute the number of points on an elliptic curve over a finite field $\mathbb{F}_q$ in at most a constant times $\log^8 q$ operations. This algorithm is fairly complicated – for a nice exposition see [2] – but allows us to calculate the number of points on $E(\mathbb{F}_q)$ for q with several hundred decimal digits.

6 Elliptic curve cryptography

Using elliptic curves as the underlying group for public key cryptography was first suggested in 1985 by N. Koblitz and V. Miller because the discrete logarithm problem was believed to be harder for elliptic curves than for finite fields.

The algorithm we present is the ElGamal cryptosystem. This has never been standardised for use with elliptic curves and as we have already noted is vulnerable to chosen ciphertext attack. For cryptographic use we recommend Elliptic Curve Integrated Encryption Scheme (ECIES), see [6].

Algorithm 6.1 (ElGamal for Elliptic Curves)**.** We present our ElGamal algorithm again for elliptic curves. Note that is basically the same as before – the biggest difference is we are now writing our group additively.

1. (Setup) An elliptic curve, E, over a finite field, $\mathbb{F}_q$, is chosen, together with a point P on the curve. The point P generates a cyclic subgroup $G = \{\mathcal{O}, P, 2P, \ldots, (n-1)P\}$ of order n. Each user picks a random integer $l \in \{0, 1, \ldots, n-1\}$ (the private key), and makes public lP (the public key). We suppose that messages are elements of G and that user A wishes to send a message, m, to user B.

2. A generates a random integer $k \in \{0, 1, \ldots, n-1\}$ and computes kP.

3. A looks up B's public key lP and computes $k(lP)$ then $m + klP$.

4. A sends to B the pair of group elements $(kP, m + klP)$.

5. B computes $(m + klP) - l(kP) = m$ and recovers the message.

Once again being able to find l from P and lP (solving the *elliptic curve discrete logarithm problem*) would allow us to crack this cryptosystem.

Studying the processes needed to carry out encryption and decryption we see that we need to do addition and subtraction on the elliptic curve. Although this is computationally harder than the corresponding operations on a finite field the smaller key sizes used for elliptic curve cryptography more than make up for this difference. Quoting from the National Security Association (NSA) in [7]:

> To use RSA or Diffie-Hellman to protect 128-bit AES keys one should use 3072-bit parameters: three times the size in use throughout the Internet today. The equivalent key size for elliptic curves is only 256 bits. One can see that as symmetric key sizes increase the required key sizes for RSA and Diffie-Hellman increase at a much faster rate than the required key sizes for elliptic curve cryptosystems. Hence, elliptic curve systems offer more security per bit increase in key size than either RSA or Diffie-Hellman public key systems.
>
> Security is not the only attractive feature of elliptic curve cryptography. Elliptic curve cryptosystems also are more computationally efficient than the first generation public key systems, RSA and Diffie-Hellman. Although elliptic curve arithmetic is slightly more complex per bit than either RSA or DH arithmetic, the added strength per bit more than makes up for any extra compute time.

We will talk no more about implementation, leaving this to more computationally minded sources such as [4]. However there are issues with the selection of elliptic curve, finite field and point P that we will discuss when we talk about attacks on elliptic curve cryptosystems.

7 Attacks

All the cryptosystems that we have seen can be cracked if we can find discrete logarithms efficiently. This has led many people to study this problem and many different attacks are known. The first algorithm we look at is the index calculus. This is an attack that is the best known when our group is the cyclic group of

a finite field. We will present it only for $\mathbb{F}_p^\times$ because the attack on $\mathbb{F}_{p^m}^\times$ requires more number theory than we have.

First we recall that $G = \mathbb{F}_p^\times$ is cyclic so $G = \langle g \rangle$ for some g. Then any $h \in G$ can be written as $h = g^k$ where k is unique mod $(p-1)$. Then let $L(h) = k$ denote the *discrete logarithm* of h with respect to g and p. Then $g^{L(h)} = h \mod p$, so if we have $h_1, h_2 \in G$ then $g^{L(h_1 h_2)} = h_1 h_2 = g^{L(h_1)} g^{L(h_2)} = g^{L(h_1)+L(h_2)}$. This means that $L(h_1 h_2) = L(h_1) + L(h_2) \mod (p-1)$. We see that the discrete log takes multiplication to addition just like the standard log.

Using this property we see that if, for some j, we can factorise $g^j h$ into a product of primes that we have already solved the discrete log problem for then we can find $L(h)$. We give the following algorithm in a slightly informal manner – there is a lot of choice in the description – but rest assured that it can be dealt with formally (for example see [9]).

Algorithm 7.1 (Index calculus for $\mathbb{F}_p^\times$)**.** This algorithm allows us to find $L(h)$ for any $h \in G = \mathbb{F}_p^\times = \langle g \rangle$.

1. Choose factor base made up of 'small' primes $B = \{-1, 2, 3, 5, 7, 11, \ldots, l\}$.
2. Find relations g^k = product of elements in factor base. These lead to linear equations after taking the discrete log.
3. Once you have enough relations, solve the linear system to find $L(2), L(3), \ldots, L(p)$. Calculate $L(-1) = \frac{p-1}{2}$.
4. Choose random j until $g^j h$ is some product of elements in the factor base. Then taking discrete logs can find $L(h)$.

An example should help to explain, the following is based on an example in [2].

Example 7.2. Take $p = 1217, g = 3$. We want to find $L(37)$. We choose our factor base $B = \{-1, 2, 3, 5, 7, 11, 13\}$. Then calculate $3^1 = 3$, $3^{24} = -2^2 \times 7 \times 13$, $3^{25} = 5^3$, $3^{30} = -2 \times 5^2$, $3^{34} = -5 \times 11$ and $3^{87} = 13$.

Taking discrete logs and recalling $L(-1) = \frac{1217-1}{2} = 608$ we get the system of equations

$$\begin{aligned} 1 &= L(3) \\ 24 &= 608 + 2L(2) + L(7) + L(13) \\ 25 &= 3L(5) \\ 30 &= 608 + L(2) + 2L(5) \\ 34 &= 608 + L(5) + L(11) \\ 87 &= L(13) \end{aligned}$$

Now we can solve this linear system to find $L(2) = 216$, $L(3) = 1$, $L(5) = 819$, $L(7) = 113$, $L(11) = 1059$ and $L(13) = 87$. The next step is try random values of j – we find $j = 16$ leads to $3^{16} \times 37 = 2^3 \times 7 \times 11 \mod 1217$. This tells us that

$$16 + L(37) = 3L(2) + L(7) + L(11)$$

and then we can easily find that $L(37) = 588$.

The above example show us the main features of this algorithm. We can see that the choice of the size of B is important. If we choose too large a factor base then we have lots of equations to solve which can become unwieldy but if we choose too small a factor base then finding a j that works might take too long. There are many methods to find the relations g^k equals a product of elements of the factor base – the current most efficient method is the number field sieve.

The expected running time of the index calculus is approximately a constant times $\exp(\sqrt{2\log p \log\log p})$ as is shown in [9]. This means that the index calculus is a *subexponential algorithm* compared to the algorithms that we will study next, which work for a general group but are *exponential algorithms*.

The index calculus relies on the factorisation of integers into primes. There is no obvious set of points corresponding to the primes on elliptic curves. There has been some suggestions of applying index calculus like techniques (such as the 'xedni calculus') but there are problems that seem insurmountable and the current belief is that such a technique will not work.

There are a number of algorithms that run in around $\sqrt{n}$ time that can be used to find discrete logarithms in general groups of order n. We study only one in detail: Baby Step, Giant Step which we present for elliptic curves.

Algorithm 7.3 (Baby Step, Giant Step for elliptic curves)**.** Given a subgroup of an elliptic curve over $\mathbb{F}_q$, $G = \{\mathcal{O}, P, 2P, \ldots, (n-1)P\}$. This algorithm allows us, for $Q \in G$, to find $k \in \{0, 1, \ldots, n-1\}$ such that $Q = kP$.

1. Fix $m > \sqrt{q+1+2\sqrt{q}}$ (so then $m > \sqrt{n}$ by Hasse's theorem). Compute and store $\{\mathcal{O}, P, 2P, \ldots, (m-1)P\}$.
2. Compute $Q - jmP$ for $j = 0, 1, \ldots, m-1$ until $Q - jmP = iP$ where iP is a point you calculated earlier.
3. Deduce that $Q = kP$ where $k = i + jm \mod n$.

To prove that this algorithm works we assume that $kP = Q$ and show that there exist i and j such that $Q - jmP = iP$. Firstly, since $m^2 > n$ we can assume that $0 \leq k < m^2$. Write $k = k_0 + mk_1$ where $k_0 = k \mod m$ so $0 \leq k_0 < m$, then we have $k_1 = (k - k_0)/m$. Then we can see that $0 \leq k_1 < m$. Now when $i = k_0$ and $j = k_1$ we have

$$Q - jmP = Q - k_1 mp = Q - (k - k_0)P = Q - kP + k_0 P = k_0 P = iP.$$

This shows that we have a match and our algorithm will work.

We now give an example, again from [2].

Example 7.4. Let E be the curve $y^2 = x^3 + 2x + 1$ over $\mathbb{F}_{41}$ and let $P = (0, 1)$ generate G. Let $Q = (30, 40)$. Calculate $\sqrt{q+1+2\sqrt{q}} = \sqrt{41+1+2\sqrt{41}} = 7.40\ldots$ so take $m = 8$. We calculate the points iP for $i = 0, 1, 2, \ldots, 7$ as

$$(0,1), (1,39), (8,23), (38,38), (23,23), (20,28), (26,9).$$

Now we calculate $Q - jmP$ for $j = 0, 1, 2$ and obtain

$$(30,40), (9,25), (26,9)$$

and notice this third point is $7P$. Now we find $k = i + jm = 7 + 2 \times 8 = 23$ and conclude that $(30, 40) = 23(0, 1)$.

Baby Step, Giant Step has the disadvantage that it requires a lot of storage space. Pollard's ρ–method and Pollard's λ–method overcome this disadvantage but are probabilistic algorithms.

The rest of the attack methods we will discuss in a informal way: don't worry if you don't know all the words – just recognise the existence of these attacks and their ramifications. For details see [2].

The *Pohlig–Hellman* method works by factorising the order of G into prime factors, solving the discrete log for each factor and putting it all back together with the Chinese Remainder theorem. To protect against this we make the order of G a large prime.

The MOV (Menezes, Okamoto and Vanstone) attack uses the Weil pairing (which we haven't discussed) to convert a discrete log problem in $E(\mathbb{F}_q)$ to one in $\mathbb{F}_{q^m}$. We have seen that the index calculus allows us to solve such a problem fairly efficiently. For general elliptic curves the value of m may be too large for this attack to work but for *supersingular* curves we can take $m = 2$.

A supersingular elliptic curve over $\mathbb{F}_q$ where $q = p^m$ is one that has no points of order p. It can be proven (see [2]) that a curve is supersingular if and only if the number of points on it is equal to 1 modulo p. This is why we mentioned Schoof's point counting algorithm earlier.

To avoid the MOV attack it was suggested that we should pick E and q such that the order of $E(\mathbb{F}_q)$ is q. Such a curve is called *anomalous*. Unfortunately there is a separate attack on anomalous curves that in fact runs in polynomial time.

All these different attacks mean that we have to be careful picking our elliptic curve and finite field. Generally, instead of figuring out the number of points and checking all the different conditions, a curve is chosen from a list such as the American National Institute for Standards and Technology (NIST) document: "Recommended elliptic curves for federal government use", see [10].

8 Digital Signatures

Another use of public key cryptography is in 'digital signature' algorithms. On a physical document a signature is (ideally) proof that you wrote the document. A *digital signature* is a similar proof that you are who you claim to be that can be sent electronically.

We present a simplified version of the ElGamal digital signature algorithm from [2].

Algorithm 8.1 (ElGamal Elliptic Curve Digital Signatures)**.** This algorithm allows a user A to send a message m, represented as an integer, to B such that B can be sure that A sent it. We choose an elliptic curve, E, over $\mathbb{F}_p$ and a point $P \in E(\mathbb{F}_p)$ such that the discrete logarithm problem is hard for $E(\mathbb{F}_p)$. Let $G = \langle P \rangle = \{\mathcal{O}, P, 2P, \ldots, (n-1)P\}$.

1. (Setup). A chooses $l \in \{0, 1, \ldots, n-1\}$ and calculates lP. A's public Key is $E, \mathbb{F}_p, lP, P$ and A's private key is l.

2. A chooses integer k with $\gcd(k, N) = 1$ and computes kP. Also A computes $s = k^{-1}(m - lx)$ where $kP = (x, y)$.

3. A sends signed message (m, kP, s). We can think of m as the message and kP, s as the signature.

4. B receives (m, kP, s) and calculates $V_1 = x(lP) + s(kP)$ and $V_2 = mP$.

5. If $V_1 = V_2$ B accepts the message as coming from A, otherwise he rejects it.

First we check that this algorithm will accept messages coming from A as valid. If it is a valid signature then $V_1 = x(lP)+s(kP) = xlP+k^{-1}(m-lx)kP = xlP + mP - lxP = mP = V_2$ as the algorithm checks for. Now we note that being able to crack the elliptic curve discrete log problem would allow you to impersonate A since you could find l from P and lP and then send any message m signed with A's signature (since could calculate the correct s).

Once again it is not proven that this is the only way to forge A's signature. Consider the problem: you have your own m that you wish to send and you can choose kP and s to send as your signature. If you pick k or s, fix the condition that $V_1 = V_2$, then try to find the other you have a discrete log problem. However there may be some way of picking k and s simultaneously (without solving a discrete log) such that $V_1 = V_2$. No one has been able to do this and it is generally believed that forging elliptic curve signatures is as difficult as cracking the discrete log problem.

An implementation consideration is that representing a long message as an integer is inefficient. For this reason a 'hash' of the message is often used as m instead. A *cryptographic hash function* is a function that takes input of arbitrary length and returns values of fixed length. Of course this means that different messages will hash to the same value. To reduce the possibility for attack and to making hashing efficient we need the function, H, to have the following properties:

1. Given a message m, the value $H(m)$ can be calculated efficiently.

2. Given y, it is computationally infeasible to find m with $H(m) = y$. (This condition is that H is *pre-image resistant.*)

3. It is computationally infeasible to find distinct messages m_1 and m_2 with $H(m_1) = H(m_2)$. (This says that H is *strongly collision free.*)

Using such a hashing function (which are available and efficient) we can implement elliptic curve digital signature algorithms in the background to provide assurance in many transactions over the internet.

9 Conclusion

Elliptic curve cryptography is implemented in the NSA's Suite B, "intended to protect both classified and unclassified national security systems and information" as described in [8], where it used for both digital signatures and key exchange. This shows that elliptic curve crypto is ready for real world use and is to be preferred in many cases over other cryptosystems.

We have shown some of the reasons for this in this paper and also seen some of the ways that elliptic curve crypto can be attacked and factors that you need to be careful of in implementation. Although there is no real mathematical proof

that elliptic curve crypto is more secure than cryptosystems based on discrete logs over finite fields or integer factorisation, elliptic curve crypto seems to be the most efficient and secure public key cryptosystem available today.

References

[1] Joseph H. Silverman, The Arithmetic of Elliptic Curves., *Graduate Texts of Mathematics Vol. 106 Springer-Verlag* 1986.

[2] Lawrence C. Washington, Elliptic Curves – Number Theory and Cryptography, *Chapman and Hall*, 2003.

[3] Peter Adams, *MATH3302 Notes*, Department of Mathematics, University of Queensland, 2005.

[4] Alfred J. Menezes, Elliptic Curve Public Key Cryptosystems, *Kluwer Academic Publishers*, 1993

[5] PARI/GP, version `2.1.7`, Bordeaux, 2005, `http://pari.math.u-bordeaux.fr/`.

[6] Victor Shoup, A Proposal for an ISO Standard for Public Key Encryption, `http://www.shoup.net/papers/iso-2_1.pdf`, 2001

[7] National Security Agency Information Assurance, *The Case for Elliptic Curve Cryptography*, `http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm`.

[8] National Security Agency Information Assurance, *Fact Sheet NSA Suite B Cryptography*, `http://www.nsa.gov/ia/industry/crypto_suite_b.cfm?MenuID=10.2.7`.

[9] A. J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997.

[10] National Institute of Standards and Technology, *Recommended elliptic curves for federal government use*, `http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf`, 1999.