Elliptic Curves in Public Key Cryptography:

The Diffie Hellman Key Exchange Protocol and its relationship to the Elliptic Curve Discrete Logarithm Problem

Public Key Cryptography

Public key cryptography is a modern form of cryptography that allows different parties to exchange information securely over an insecure network, without having first to agree upon some secret key. The main use of public key cryptography is to provide information security in computer science, for example to transfer securely email, credit card details or other secret information between sender and recipient via the internet.

There are three steps involved in transferring information securely from person A to person B over an insecure network. These are encryption of the original information, called the plaintext, transfer of the encrypted message, or ciphertext, and decryption of the ciphertext back into plaintext. Since the transfer of the ciphertext is over an insecure network, any spy has access to the ciphertext and thus potentially has access to the original information, provided he is able to decipher the message. Thus, a successful cryptosystem must be able encrypt the original message in such a way that only the intended receiver can decipher the ciphertext. The goal of public key cryptography is to make the problem of deciphering the encrypted message too difficult to do in a reasonable time (by say brute-force) unless certain key facts are known. Ideally, only the intended sender and receiver of a message should know these certain key facts.

Any certain piece of information that is essential in order to decrypt a message is known as a key. A key(s) specifies the particular function that transforms the original message into ciphertext and vice versa. Public key cryptography relies on two keys, a

private key and a public key. The public key is published in a place where anyone has access to it. However, each individual person or computer also chooses a private key that should be known only to that individual. The importance of the key to the encryption algorithm should be so great, that if the key is lost, it should be computationally infeasible to recover the original message from the encrypted data.

In 1976, Diffie and Hellman proposed the use of certain one-way functions called trap-door functions, to make it almost impossible to decipher encrypted data without a key. The idea is that for each key k, we can choose an encryption function $f_k: M \to C$ such that it is almost impossible to compute f_k^{-1} without knowing k, in polynomial time. Here, M is the set of messages and C is the set of ciphertext. Before we illustrate how the cryptosystem works, we make a few observations about these trap-door functions. Firstly, in order to avoid ambiguity in transferring messages, we insist that the encryption function f_k is injective. That is, each message $m \in M$ corresponds to one and only one ciphertext $c \in C$. If we choose M=C, then f_k is a bijection and this implies that each element of M must have an inverse in M. Therefore, it is natural to choose M to be a group. Later, we will see that in elliptic curve cryptography, the group M is the group of rational points on an elliptic curve. Before we delve into public key cryptography using elliptic curves, I will give an example of how public key cryptosystems work in general.

Suppose person A want to send a message to person B. Person A chooses some key, k, and an encryption function f_k as defined above. Person A publishes $f_{k,}$ which is his public key. Therefore, anyone who wishes to send a message to person A must look up his encryption function $f_{k,}$ and encrypt their message using this function. Once the message is encrypted, it is very difficult to decipher. This is why the trap door or secret

key is necessary. We call k, the secret key and only with the knowledge of k, can f_k^{-1} be computed easily and hence the message decrypted. Person A is the only one who should know the value of k and thus, person A is the only one who should be able to decrypt the message.

There is one last point about the description of the trap door function f_k that I have left quite vague. That is, I have not said what it means for it to be "almost impossible" to obtain f_k^{-1} from f_k. In the language of complexity theory, "almost impossible to compute f_k^{-1} from f_k" means "impossible to compute f_k^{-1} from f_k by a deterministic algorithm in polynomial time". Now, due to the unsolved nature of the P=NP problem, it is not known whether there actually exists a trap door function. However, we do have functions that behave like trap door functions in the time frame within which we are trying to solve the problem i.e. "reasonable" time. However, what counts as reasonable time depends on the level of security one desires. For example, for an intelligence agency, "reasonable time" will be much longer than that for someone sending an e-card.

Diffie-Hellman Key Exchange

The cryptosystem we aim to achieve is one where the sender and receiver exchange pieces of information via an insecure network resulting in both parties sharing a common secret whereas anyone else who intercepts the transfer of the message, is unable to discover the shared secret. This shared secret is what is used as a key in conventional cryptosystems. Such a system is a full public key cryptosystem. We illustrate this key exchange protocol with an example.

Alice and Bob aim to exchange information using a public key cryptosystem.

1. They publicly choose a cyclic group G and a generator x of G.
2. Alice and Bob choose private keys a and b respectively, where a and b are random integers.
3. Alice computes x^a, Bob computes x^b and they exchange these values over an insecure network.
4. On receiving the information from each other, both Alice and Bob compute the value x^{ab} using their private keys and the fact that $x^{ab} = (x^a)^b = (x^b)^a$.

Now, both Alice and Bob share a secret, namely, the value x^{ab}. That is, Alice and Bob have exchanged a key, x^{ab}, that can now be used in a conventional cryptosystem to encrypt any messages between Alice and Bob.

If the message was intercepted, the eavesdropper, in order to decipher the message, has to obtain the value x^{ab} from x, x^a and x^b. This problem is called the Diffie-Hellman problem. One way to tackle this problem is to try to compute a from x^a. This is known as the discrete logarithm problem.

With the basics of public key cryptography in hand, we are now in a position to apply elliptic curves to public key cryptography in order to generate public and private keys.

Elliptic Curve Cryptography

In 1985, Neal Koblitz and Victor Miller independently suggested the use of elliptic curves in public key cryptography. Supporters of elliptic curve cryptography

(ECC) claim that ECC requires much smaller keys than those used in conventional public key cryptosystems, while maintaining an equal level of security. The use of elliptic curves therefore allows faster encryption and decryption.

We now recall a few facts about elliptic curves before illustrating the application to public key cryptography.

Given an elliptic curve E and a field F_q, we consider the rational points $E(F_q)$ of the form (x,y) where both x and y belong to F_q. We choose the point at infinity to be σ.

1. Define the operation "+" on the set of rational points of E as follows. If P and Q are two rational points on E, then P+Q is given by the following rule:

   Draw the line joining P and Q, take the third point of intersection of this line with the curve as R. Draw the line through σ and R, and take the third point of intersection of this line with E. This point is the point P+Q. Note the operation "+" is commutative.

   In particular we have, σ+ σ= σ and P+(-P)= σ.

2. Define the operation "*" as follows * : $Z \times E(F_q) \rightarrow E(F_q)$ and if P is some point in $E(F_q)$, then we define n*P as P+P+P+…..+P, n times. Note that for integers j and k, j*(k*P) = (j*k)*P = k*(j*P).

3. The set of rational points on E form an abelian group under the operation "+" with identity σ.

Definition: The elliptic curve discrete logarithm problem (ECDLP) is to determine the integer k, given rational points P and Q on E, and given that k*P=Q.

Elliptic curve public key cryptography is based on the premise that the elliptic curve discrete logarithm problem is very difficult; in fact, much more so than the discrete logarithm function for a multiplicative group over a finite field. As mentioned before a group is normally used in public key cryptography as the domain on which we define our encryption function. This is because we need every element of our domain to have an inverse and vice versa. In elliptic curve cryptography, the group used is the group of rational points on a given elliptic curve.

This is how elliptic curve public key cryptography works. For Alice and Bob to communicate securely over an insecure network they can exchange a private key over this network in the following way:

1. A particular rational base point P is published in a public domain for use with a particular elliptic curve $E(F_q)$ also published in a public domain.
2. Alice and Bob choose random integers k_A and k_B respectively, which they use as private keys.
3. Alice computes k_A*P, Bob computes k_B*P and they exchange these values over an insecure network.
4. Using the information they received from eachother and their private keys, both Alice and Bob compute $(k_A*k_B)*P = k_A*(k_B*P) = k_B*(k_A*P)$. This value is then the shared secret that only Alice and Bob

possess. Note that the difficulty of the ECDLP ensures that the private keys k_A and k_B and the shared secrety $(k_A*k_B)*P$ are difficult to compute given k_A*P and k_B*P. Thus, Alice and Bob do not compromise their private keys or their shared secret in the exchange.

Now that Alice and Bob share this secret that is almost impossible for a third party to discover, they can use this shared secret in a classical cryptosystems to communicate securely over the network.

The Elliptic Curve Discrete Logarithm Problem

As stated before, the ECDLP is the problem of determining the integer k, given a rational point P on the elliptic curve E and the value of k*P. Elliptic curve cryptosystems rely on the difficulty of solving the ECDLP. If an eavesdropper is able to solve the ECDLP then the eavesdropper will be able to break the system. Therefore, it is of great importance to understand the methods of tackling the ECDLP. For, we can use the success of these methods as a measure of the security of the system.

Many proponents of the use of elliptic curves in public key cryptography support their view based on a belief that the ECDLP is much more intractable that the DLP in finite fields. The strongest techniques normally used to solve the problem in finite fields are Shank's baby-step giant-step algorithm, Pollard's ρ-method, the Pohlig-Hellman method and the index calculus method. None of these methods works for the elliptic curve problem. Until 1990, the only discrete log algorithms that worked for elliptic curves were exponential time algorithms. These algorithms are general in that they

worked for any group irrespective of group structure. In addition, they only work if the group order is divisible by some large prime. Therefore, it seemed like elliptic curves provided excellent security in public key cryptosystems due to the difficulty of solving the ECDLP. In 1993, the problem was reduced by Menezes, Okamoto and Vanstone from the ECDLP to the DLP on $F^*_{q^k}$. However, this method only works for so-called supersingular curves such as curves of the form $y^2 = x^3 + ax$ when the characteristic p of F_q is $\equiv$-1(mod 4), and curves of the form $y^2 = x^3 + b$ when p $\equiv$-1(mod3); i.e. curves for which k is small. Their method is known as the MOV method. This method is only useful for a small class of elliptic curves because most elliptic curves are not supersingular. I will discuss here one of the methods for solving the DLP over a finite field, the index calculus method, and show how it breaks down in the case of the ECDLP.

The Index Calculus Method

The index calculus method provides a probabilistic subexponential algorithm that is adapted to the multiplicative group of a finite field. There are several index calculus methods but they all consist of two phases. Let the discrete logarithm $l = \log_\alpha \beta$.

Definition: A non-deterministic algorithm with input size log n, is subexponential if there exists constants c>0 and $\alpha \in [0,1)$ such that the expected running time of the algorithm is in

$$L[\alpha,c] = O\left(e^{(c+o(1))(\log n)^{\alpha}(\log\log n)^{1-\alpha}}\right)$$

For $\alpha = 0$, a subexponential algorithm becomes polynomial and for $\alpha = 1$, a subexponential algorithm becomes fully exponential.

Phase 1: Collecting linear equations

Choose a "factor base" $\Gamma = \{\gamma_1, \gamma_2, \ldots \gamma_t\}$ of elements of a group G, with order n. In phase 1, we will try to determine the discrete logarithms of the γ_i's. To do this, we repeatedly choose random integers $s \in \{0, 1, \ldots.n-1\}$ and compute α^s. We then try to factor α^s in Γ. If we do find an equation of the form

$$\alpha^s = \prod_{i=1}^{t} \gamma_i^{\nu_i}$$

then, we have the linear equation in Z_n

$$s = \sum_{i=1}^{t} \nu_i \log_\alpha \gamma_i \ .$$

Sufficiently many of these equations will allow us to solve for the $\log_\alpha \gamma_i$'s.

Phase 2: Calculating individual logarithms.

We select random integers s and try to factor $\beta\alpha^{-s}$ in Γ. If we succeed, we have the equation

$$\beta\alpha^{-s} = \prod_{i=1}^{t} \gamma_i^{\nu_i}$$

and so

$$\log_\alpha\beta = s + \sum_{i=1}^{t} \nu_i \log_\alpha \gamma_i$$

where all the numbers on the right hand side are known.

This procedure allows one to calculate the discrete logarithm l as described above. We note that the procedure entails balancing the time allotted to each phase. A large

factor group slows down phase 1 but speed up phase 2. Therefore, depending on how many discrete logarithms, in the same field, need to be calculated, we can choose an appropriate size for the factor base.

In order to apply this procedure to finding discrete logarithms, we need to be able to choose a factor base, which can efficiently construct the equations above. Suitable factor bases are known for finite fields and for class groups of imaginary quadratic number fields. The index calculus algorithms with the best-proven running times are those due to Pomerance for F_p and F_{2m}, which run in $L\left[1/2, \sqrt{2}\right]$.

Elliptic Curve Logarithms

While the index calculus method solves the DLP for certain fields, it does not work on the ECDLP. In fact, the main advantage of using a cryptosystem based on elliptic curves is that no subexponential algorithms are known, except for some rare classes of curves. Miller argued and Silverman and Suzuki confirmed that an elliptic curve analogue of the index calculus method is unlikely to exist.

For an elliptic curve S defined over the finite prime field F_p, the coefficients can be lifted to integers. Consider a possible lifting to E_Q. If we restrict our curve to $E_{Z(p)}$, where $Z_{(p)} = \{a/b \in Q : p \text{ does not divide } b\}$ is the localised ring of Z at p, then the coordinates of a point on $E_{Z(p)}$ can be reduced modulo p to obtain a point on the curve E_{Fp}. We would then apply the index calculus method in the following way. Fix a factor base containing some points of E_{Fp} which are lifted to points on $E_{Z(p)}$. Choose several random elements of E_{Fp} as described above and factor them lifting to point on $E_{Z(p)}$ and

expressing them as a linear combination of the lifted factor base. The result is that the reduced points satisfy the same linear relation on E_{Fp}.

The problem with this procedure is that no simple method is known for lifting points on E_{Fp} to points on $E_{Z(p)}$. Moreover, in order to make the algorithm efficient, we need a factor base made up of only points of small height, where the logarithmic height of a point $\left(x/d, y/d\right)$ with x, y, d integers and gcd(x,y,d) = 1, is given by log max{|x|, |y|, |d|}. But the logarithmic heights of multiples nP of some point P say, grow quadratically in n. However, by the Mordell-Weil theorem, the maximum number of linearly independent points on a curve E_Q, is finite, and is quite small in general. Thus, we conclude that there are probably not enough points of small height to construct a sufficiently large factor base. Thus, in general, the ECDLP appears to be unsolvable in polynomial time using the index calculus method. This is good news for public key cryptography based on elliptic curves, as we see that it is very difficult for an eavesdropper to obtain the shared secret between Alice and Bob if he intercepts the message being transferred. That is, due to the difficulty of the ECDLP, Diffie-Hellman key exchange is secure for public key cryptography based on elliptic curves.

References

Enge, A. *Elliptic Curves and their Applications to Cryptography; an Introduction* (1999).

Koblitz, N. *A Course in Number Theory and Cryptography* (1994).

Koblitz, N. *Algebraic Aspects of Cryptography* (1998).

Koblitz, N. *Towards a Quarter-Century of Public Key Cryptography* (2000).

Silverman, J. and Tate, J. *Rational Points on Elliptic Curves* (2000).