Elliptic Curve Diffie-Hellman Key Exchange

MTH426: Algebraic Curves

Manvendra Somvanshi (MS20126)

Department of Mathematical Sciences
IISER Mohali

April 28, 2024

Outline

Introduction

- Modern society is highly dependent on encyrption: all banking transactions, personal messages, military secrets, etc. require cryptographic algorithms and encryption.

Introduction

- Modern society is highly dependent on encyrption: all banking transactions, personal messages, military, etc. require cryptographic algorithms and encryption.
- Suppose that Alice and Bob want to communicate remotely but *Big Brother is watching*. They devise an encryption function f which takes a key k and a message M and encrypts it. This is invertible so that the same key can be used to decrypt the message using f^{-1}.

Introduction

- Modern society is highly dependent on encyrption: all banking transactions, personal messages, military, etc. require cryptographic algorithms and encryption.
- Suppose that Alice and Bob want to communicate remotely but *Big Brother is watching*. They devise an encryption function f which takes a key k and a message M and encrypts it. This is invertible so that the same key can be used to decrypt the message using f^{-1}.
- Alice and Bob can safely communicate with each other once they agree on a secret key k, even though the algorithm f is known publicly. But how do they agree on a secret key? This process is known as *Key Sharing*.

Outline

Discrete Log Problem

Discrete Log Problem

Let G be a group and $g, h \in G$ where we know that h is some power of g. Then determine $n \in \mathbb{Z}$ such that $h = g^n$.

Discrete Log Problem

Discrete Log Problem

Let G be a group and $g, h \in G$ where we know that h is some power of g. Then determine $n \in \mathbb{Z}$ such that $h = g^n$.

The Discrete Log problem is a classically hard problem. This means that there is no algrithm on classical computers that can solve it in polynomial time (in the number of digits of $|G|$).

Discrete Log Problem

Discrete Log Problem

Let G be a group and $g, h \in G$ where we know that h is some power of g. Then determine $n \in \mathbb{Z}$ such that $h = g^n$.

The Discrete Log problem is a classically hard problem. This means that there is no algrithm on classical computers that can solve it in polynomial time (in the number of digits of $|G|$).

Note:

This is not post quantum secure! Solving Discrete Log problem comes down to solving prime factorization, which can solved by quantum computers by Shor's algorithm.

Outline

Diffie Hellman Key Exchange

- Let's return to Alice, Bob, and Big brother. The Diffie-Hellman key exchange works as follows:
  1. Alice and Bob publically agree on a group G an element g.
  2. Alice and Bob each choose an integer, say $a, b \in \mathbb{Z}$ respectively, privately. This is not known to anyone.
  3. Alice makes the element $h_A = g^a$ public and Bob makes $h_B = g^b$ public.
  4. Alice calculates $(h_B)^a = (g^b)^a$ and Bob calculates $(h_A)^b = (g^a)^b$. Clearly both are the same and thus they have a shared secret.
  5. This shared secret $k = g^{ab}$ is used are their key for symmetric encryption.
- Here a, b are called private keys of Alice and Bob and h_A, h_B are public keys.

Diffie Hellman Key Exchange

Alice	Public (Big Brother)	Bob
	(G, g)	
a		b
	g_A, g_B	
$k = g^{ab}$		$k = g^{ab}$

Table: Diffie Hellman Key exchange.

- Big Brother only has g_A, g_B and the information that these are powers of g. Thus he has to solve the discrete log problem to determine the encryption key k.
- Thus Alice and Bob have successfully shared a secret which the Big Brother cannot figure out, if the group chosen is large enough.
- Note that DH key exchange is vulnerable to MITM attacks!

Outline

Elliptic Curves

Definition

An elliptic curve E in $\mathbb{P}^2$ is given by $Y^2Z = X^3 + aXZ^2 + bZ^3$ where $a, b, c \in k$.

Elliptic Curves

Definition

An elliptic curve in $\mathbb{P}^2$ is given by $Y^2Z = X^3 + aXZ^2 + bZ^3$ where $a, b, c \in k$.

- The curve $E_* \subset U_3 \cong \mathbb{A}^2$ with $a = -1$ & $b = 1$ is shown in the figure below. Outside U_3 the curve only passes through one point $(0 : 1 : 0)$, which we call the point at infinity. Call this point ∞.

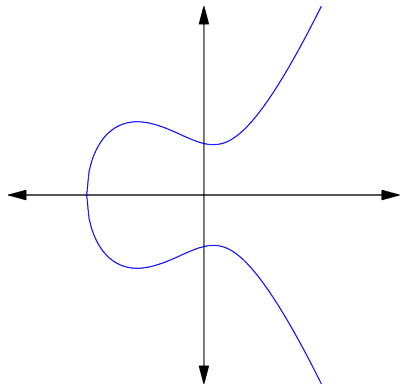

Figure: $V(Y^2Z = X^3 - XZ + Z^3) \cap U_3$.

We want to consider the elliptic curves which are non-singular. The curve is singular at a point p iff $(F_X, F_Y, F_Z, F) = 0$ when evaluated at p. Let $p = (x : y : z)$, then we get the system of equations:

$$3x^2 + az^2 = 0,\ 2yz = 0,\ y^2 = 2axz + 3bz^2,\ y^2z = x^3 + axz^2 + bz^3$$

If $z = 0$ then we know that the point p is ∞ which does not solve the third equation. Thus we can assume that $z = 1$ and work in $\mathbb{A}^2$. We get the equations:

$$3x^2 + a = 0,\ y = 0,\ y^2 = 2ax + 3b,\ y^2 = x^3 + ax + b.$$

this can be solved to get that p is $(\pm\sqrt{-a/3} : 0 : 1)$. Substituting this in the other two equations, we get the same equation $4a^3 - 27b^2 = 0$.

The previous slide can be summarized as:

Proposition.

An elliptic curve E is non-singular if and only if $4a^3 - 27b^2 \neq 0$.

Addition on Elliptic Curves

Addition on E

Let $o \in E$ be a finxed point and p, q be arbitrary points on E and let L be the unique line passing through p, q. Then we have $L \cap C = p + q + r$ (p, q, r may not be distinct). Define $\phi : C \times C \to C$ to be the function $(p, q) \mapsto r$. Define $p \oplus q = \phi(o, \phi(p, q))$.

Theorem

(E, o) with the operation $\oplus$ is an abelian group.

Proof.

The identity is o since if $\phi(o, p) = r$ then $\phi(o, r) = p$. The inverse of p is just $r = \phi(o, p)$. The abelian part is also easy to see since $\phi(p, q) = \phi(q, p)$. Associativity is non-trivial. Consider three points p_1, p_2, p_3 and let L_1 be the line joining p_1, p_2. □

Addition on Elliptic Curves

proof (cont.)

Now let

$$L_1 \cap C = p + q + s', \ M_1 \cap C = o + s' + s,$$
$$L_2 \cap C = r + s + t', \ M_2 \cap C = q + r + u',$$
$$L_3 \cap C = o + u' + u, \ M_3 \cap C = p + u + t'',$$

where L_i and M_i are lines. Then define $C_1 = L_1L_2L_3$ and $C_2 = M_1M_2M_3$. Then

$$C_1 \cap C = p + q + s' + r + s + t' + o + u + u'$$
$$C_2 \cap C = o + s' + s + q + r + u' + p + u + t''$$

Since all the points are same except t' and t'' we have that $t' = t''$. Since $p \oplus (q \oplus r) = t''$ and $(p \oplus q) \oplus r = t'$, the associativity follows. □

Outline

Elliptic Curve Diffie Hellman

- Let E be an elliptic curve over $\mathbb{F}_p$ and fix the identity element to be ∞. Let p_0 be some fixed point. The elliptic curve itself is determined by constants $a, b \in k$. Let $np_0 = \infty$. Then define $h = |E|/n$ (which is an integer by Lagrange's theorem).
- The tuple (p, a, b, p_0, n, h) is called the domain of E.
- In ECDH, the domain of a curve E is made public. Alice and Bob choose integers d_A and d_B between 1 and $n - 1$.
- Alice shares $Q_A = d_A p_0$ publically and Bob shares $Q_B = d_B p_0$ publically.
- Alice calculates $d_A Q_B$ and Bob calculates $d_B Q_A$. Both of these are the same point on the curve (x, y). Then x is chosen as the encyption key for symmetric encyption.

Elliptic Curve Diffie Hellman

- For ECDH the choice of the curve matters a lot. For some special curves, the ECDLP has been cracked.
- Calculating the parameters n and h require quite heavy computation. This is why there are a lot of standard curves which are prescribed by NIST.
- For the purposes of efficiency NIST recommends using psuedo-Mersenne primes p (of the form $2^{n_1} - 2^{n_2} - \cdots - 1$), when choosing the field $\mathbb{F}_p$. They also recommend $a = -3$, as it makes addition faster.

Implementation of ECDH

- Below is a simple implimentation of ECDH using a python script. I am using the `tinyec` package, and use the curve `brainpool256r1` which has a 256 bit key.

Implementation of ECDH

```python
from tinyec import registry
import secrets

def compress(pubKey):
    return hex(pubKey.x) + hex(pubKey.y % 2)[2:]

curve = registry.get_curve('brainpoolP256r1')

# gen Alice's private and public keys
alicePrivKey = secrets.randbelow(curve.field.n)
alicePubKey = alicePrivKey * curve.g
print("Alice public key:", compress(alicePubKey))

# gen Bob's private and public key
bobPrivKey = secrets.randbelow(curve.field.n)
bobPubKey = bobPrivKey * curve.g
print("Bob public key:", compress(bobPubKey))

print("Now exchange the public keys (e.g. through Internet)")

aliceSharedKey = alicePrivKey * bobPubKey
print("Alice shared key:", compress(aliceSharedKey))

bobSharedKey = bobPrivKey * alicePubKey
print("Bob shared key:", compress(bobSharedKey))

print("Equal shared keys:", aliceSharedKey == bobSharedKey)
```

Thank You!