EXPERIMENT 6
ELLIPTIC CURVE CRYPTOGRAPHY

6.1 Aim

To write a program for implementing the Elliptical curve cryptography algorithm.

6.2 Theory

Node.js has an in-built crypto module which can be used to run code using Javascript.
This Experiment implements the Elliptic Curve Diffie Hellman (ECDH) key exchange method.
We define y^2=x^3+ax+b over Fp, and the parameters as T=(p;a;b;G;n;h).
256-bit Elliptic curve: secp256k1. This has the strength of a 128-bit symmetric key and 3072-bit RSA.
Elliptic Curve Diffie Hellman (ECDH) is used to create a shared key. In this experiment secp256k1 (as used in Bitcoin) is used to generate points on the curve.
Its format is: y^2=x^3+7
with a prime number (p) of 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
and which is 2^256−2^32−2^9−2^8−2^7−2^6−2^4−1
All the operations will be (mod p)

Bob will generate a public key and a private key by taking a point on the curve. The private key is a random number (dB) and the Bob's public key (QB) will be:
QB=dB×G

Alice will do the same and generate her public key (QA) from her private key (dA):
QA=dA×G

They then exchange their public keys. Alice will then use Bob's public key and her private key to calculate:
SharekeyAlice=dA×QB
This will be the same as:
SharekeyAlice=dA×dB×G

Bob will then use Alice's public key and his private key to determine:
SharekeyBob =dB×QA
This will be the same as:
SharekeyBob=dB×dA×G

And the keys will thus match.

6.3 Algorithm

1. START
2. Create a Node.js project to work .
3. Create the ecc.js file.
4. Import the internal crypto module of Node.js .
5. Create an Elliptic Curve Diffie-Hellman (ECDH) key exchange object using a predefined curve, secp256k1 using createECDH function of the crypto module for Alice
6. Generate private and public EC Diffie-Hellman key values, and return the public key.
7. Perform the same step 5 and 6 to generate keys for Bob.
8. Compute the shared secret using otherPublicKey as the other party's public key and return the computed shared secret using the computeSecret function of the crypto module.
9. Log the Secret of Alice and Bob in the console.
10. Convert the Secret of Alice and Bob into Hex.
11. Log the converted Hex string in console

6.4 Program

```
// import crypto module
const { createECDH } = require('crypto');

// Generate Alice's keys
const alice = createECDH('secp256k1');
const aliceKey = alice.generateKeys();

// Generate Bob's keys
const bob = createECDH('secp256k1');
const bobKey = bob.generateKeys();

// Exchange and generate the secret
const aliceSecret = alice.computeSecret(bobKey);
console.log("aliceSecret : ", aliceSecret);

const bobSecret = bob.computeSecret(aliceKey);
console.log("bobSecret : ", bobSecret);

// convert secret into hex
let aliceHexSecret = aliceSecret.toString("hex");
```

```
console.log("aliceHexSecret : ", aliceHexSecret);

let bobHexSecret = bobSecret.toString("hex");
console.log("bobHexSecret : ", bobHexSecret);
```

6.5 Output

```
C:\Users\cinoy\OneDrive\Desktop\sc lab\Exp6>node ecc.js
aliceSecret :  <Buffer 2a aa 23 e8 b3 5f ab 51 ee 21 0f da c8 d4 b2 eb 66 4e 7b cb e3 bd 40 30 c4 e4 75 96 63 fe 26 17>
bobSecret :  <Buffer 2a aa 23 e8 b3 5f ab 51 ee 21 0f da c8 d4 b2 eb 66 4e 7b cb e3 bd 40 30 c4 e4 75 96 63 fe 26 17>
aliceHexSecret :  2aaa23e8b35fab51ee210fdac8d4b2eb664e7bcbe3bd4030c4e4759663fe2617
bobHexSecret :  2aaa23e8b35fab51ee210fdac8d4b2eb664e7bcbe3bd4030c4e4759663fe2617
```

3.5 Result

The Elliptical curve cryptography algorithm was implemented successfully.