Diffie Hellman

Diffie–Hellman key exchange

- One of the earliest examples of key exchange (1976).
- The Diffie–Hellman key exchange allows two parties to jointly establish a shared secret key over an insecure communications channel.
- This key can then be used for symmetric key encryption.

Diffie–Hellman Parameters

- Three parameters are used.
  - prime p
  - base g,
  - the length in bits of the private value, l
- These are shared but it doesn't matter if they are known by other parties.

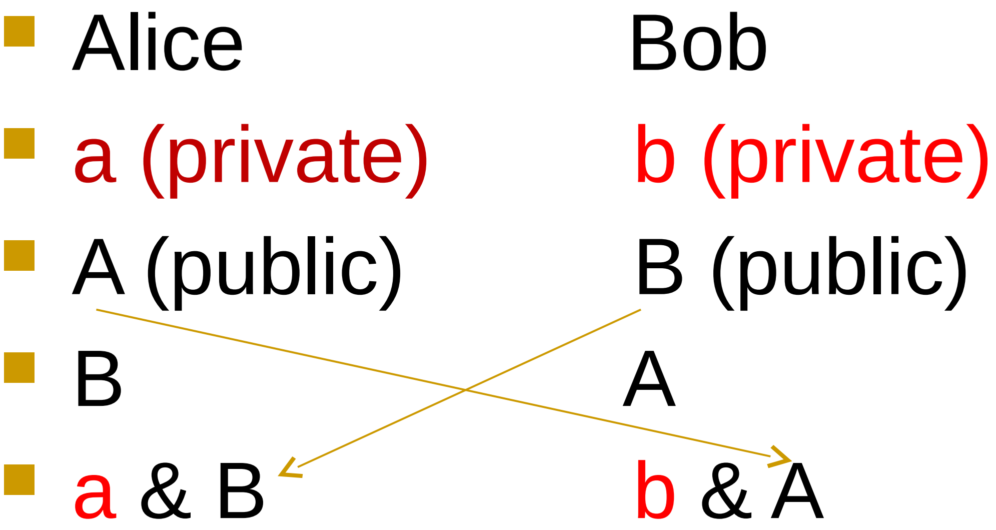
Diffie-Hellman Key Agreement

- Both parties obtain the DH parameters
- (In practise, one party can generate them and send to the other.)
- Both parties generate a Diffie-Hellman public-key/private-key pair.
- Both parties send the public key to the other party.

Diffie-Hellman Key Agreement

- Both parties use their own private key and the others public key to generate a symmetric key.
- The values are the same.

Diffie/Hellman



Alice	Bob
a (private)	b (private)
A (public)	B (public)
B	A
a & B	b & A

- symmetric key == symmetric key

Diffie/Hellman

- Alice – generates a
- Calculates g^a mod p = A (public)
- Bob – generates b
- Calculates g^b mod p = B (public)

Alice

- Has a and B (g^b mod p)
- Calculates
  - B^a mod p = (g^b mod p)^a mod p

    = g^{ab} mod p

Bob

- Has b and A (g^a mod p)
- Calculates
  - A^b mod p = (g^a mod p)^b mod p

    = g^{ab} mod p

Example

- $p = 23$, $g = 5$
- $a = 6$, $A = 5^{6} \bmod 23 = 15{,}625 \bmod 23 = 8$
- $b = 15$, $B = 5^{15} \bmod 23$
  $= 30{,}517{,}578{,}125 \bmod 23 = 19$

- Alice $s = 19^{6} \bmod 23 = 2$
- Bob $s = 8^{15} \bmod 23 = 2$

Cryptanalysis

- With p = 23 there are only 23 possible values for the public keys (n mod 23).
- Better
  - p is a prime of ate least 300 digits.
  - a, b at least 100 digits long
- To solve is the "discrete logarithm problem" and is not possible for these types of values.

Three programs

- Generate DH parameters and save them to a file
- Generate DH public key/private key pair and save then to a file (Execute for Alice and Bob)
- Read in both pairs of keys,
  - generate an AES key from AlicePrivate and BobPublic
  - generate an AES key from BobPrivate and AlicePublic
  - Show they are the same.

Generate DH Parameters

```
AlgorithmParameterGenerator paramGen = AlgorithmParameterGenerator
    .getInstance("DH");
paramGen.init(1024);

// Generate the parameters
AlgorithmParameters params = paramGen.generateParameters();
DHParameterSpec dhSpec = params
    .getParameterSpec(DHParameterSpec.class);

String s = dhSpec.getP() + "," + dhSpec.getG() + "," + dhSpec.getL();
System.out.println(s);
writeToFile("data/dhParams", s) ;
```

GenerateAndSaveKeys

```
String PARTY = args[0];

// get DH parameters
String valuesInStr = (String) readFromFile("data/dhParams");
String[] values = valuesInStr.split(",");
BigInteger p = new BigInteger(values[0]);
BigInteger g = new BigInteger(values[1]);
int l = Integer.parseInt(values[2]);
DHParameterSpec dhSpec = new DHParameterSpec(p, g, l);

// Use the values to generate a key pair
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DH");
keyGen.initialize(dhSpec);
KeyPair keypair = keyGen.generateKeyPair();
```

GenerateAndSaveKeys

```
// Save the private key
PrivateKey privateKey = keypair.getPrivate();
writeToFile("data/" + PARTY + "Private", privateKey) ;

// Save the public key
PublicKey publicKey = keypair.getPublic();
writeToFile("data/" + PARTY + "Public", publicKey) ;
```

Two Run Configurations

- This has to be run twice, one for each party.
  - RC - Run As – Run Configurations
  - New – java Application
  - Name –
  - Arguments – Alice
- Create a second Run Configuration
  - Bob

Two Run Configurations

- Run twice to generate files
  - AlicePrivate
  - AlicePublic
  - BobPrivate
  - BobPublic

GenerateSymKeyTwiceAndCheck

```
// read both keypairs
PrivateKey privateKey1 = (PrivateKey) readFromFile("data/AlicePrivate");
PrivateKey privateKey2 = (PrivateKey) readFromFile("data/BobPrivate");
PublicKey publicKey1 = (PublicKey) readFromFile("data/AlicePublic");
PublicKey publicKey2 = (PublicKey) readFromFile("data/BobPublic");

// AlicePrivate and BobPublic
KeyAgreement ka = KeyAgreement.getInstance("DH");
ka.init(privateKey1);
ka.doPhase(publicKey2, true);
byte[] rawValue = ka.generateSecret();
SecretKey secretKey1 = new SecretKeySpec(rawValue, 0, 16, "AES");

String encodedKey = Base64.getEncoder().
                      encodeToString(secretKey1.getEncoded());
System.out.println("Base64 encoded secret key 1 " + encodedKey);
```

GenerateSymKeyTwiceAndCheck

```
// AlicePublic and BobPrivate
ka.init(privateKey2);
ka.doPhase(publicKey1, true);
byte[] rawValue2 = ka.generateSecret();
SecretKey secretKey2 = new SecretKeySpec(rawValue2, 0, 16, "AES");

String encodedKey2 = Base64.getEncoder().
                                   encodeToString(secretKey2.getEncoded());
System.out.println("Base64 encoded secret key 2 " + encodedKey2);
```