Lab 5: Key Exchange

Objective: Key exchange allows us to pass a shared secret key between Bob and Alice. The main methods for doing this are either encrypting with the public key, the Diffie Hellman Method and the Elliptic Curve Diffie Hellman method. This lab investigates these methods.

🕮 **Web link (Weekly activities):** https://asecuritysite.com/esecurity/unit05

Demo: https://youtu.be/3n2TMpHqE18

A Diffie-Hellman

No	Description	Result

A.1	Bob and Alice have agreed on the values:
	g=2879, N= 9929
	Bob Select x=6, Alice selects y=9
		Now calculate (using the Windows calculator):
		Bob's A value (g^x mod N):
		Alice's B value (g^Y mod N):

A.2	Now they exchange the values. Next calculate the shared key:	Bob's value (B^x mod N):
		Alice's value (A^Y mod N):
		Do they match? [Yes] [No]

A.3	If you are in the lab, select someone to share a value with. Next agree on two numbers (g and N). You should generate a random number, and so should they. Do not tell them what your random number is. Next calculate your A value, and get them to do the same. Next exchange values.	Numbers for g and N: Your x value: Your A value: The B value you received: Shared key: Do they match: [Yes] [No]

B OpenSSL (Diffie-Hellman)

No	Description	Result
A.1	Generate 768-bit Diffie-Hellman parameters:	What is the value of g:

	`openssl dhparam -out dhparams.pem 768 -text`	How many bits does the prime number have? How long does it take to produce the parameters for 1,024 bits (Group 2)? How long does it take to produce the parameters for 1536 bits (Group 5)? How would we change the g value?

B Discrete Logarithms

B.1 ElGamal and Diffie Hellman use discrete logarithms. This involves a generator value (*g*) and a prime number. A basic operation is g^x (mod p). If p=11, and g=2, determine the results (the first two have already been completed):

x	**g^x (mod p)**
1	2
2	8
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Note: In Python you can implement this as:

```
g=2
p=11
x=3
print g**x % p
```

What happens to the values once we go past 10?

What happens to this sequence if we use *g*=3?

B.2 We can determine the values of g which will work for a given prime number with the following:

```
import sys
import random

p=11

def getG(p):

  for x in range (1,p):
        rand = x
        exp=1
        next = rand % p

        while (next <> 1 ):
                next = (next*rand) % p
                exp = exp+1


        if (exp==p-1):
                print rand

print getG(p)
```

Run the program and determine the possible g values for these prime numbers:

p=11:

p=41:

B.3 We can write a Python program to implement this key exchange. Enter and run the following program:

```
import random
import base64
import hashlib
import sys

g=9
p=1001

a=random.randint(5, 10)

b=random.randint(10,20)

A = (g**a) % p
B = (g**b) % p

print 'g: ',g,' (a shared value), n: ',p, ' (a prime number)'

print '\nAlice calculates:'
print 'a (Alice random): ',a
print 'Alice value (A): ',A,' (g^a) mod p'
print '\nBob calculates:'
print 'b (Bob random): ',b
print 'Bob value (B): ',B,' (g^b) mod p'

print '\nAlice calculates:'
keyA=(B**a) % p
print 'Key: ',keyA,' (B^a) mod p'
print 'Key: ',hashlib.sha256(str(keyA)).hexdigest()

print '\nBob calculates:'
keyB=(A**b) % p
print 'Key: ',keyB,' (A^b) mod p'
print 'Key: ',hashlib.sha256(str(keyB)).hexdigest()
```

Pick three different values for *g* and *p*, and make sure that the Diffie Hellman key exchange works:

g =   p=

g=   p=

g=   p=

C Elliptic Curve Diffie-Hellman (ECDH)

ECDH is now one of the most used key exchange methods, and uses the Diffie Hellman method, but adds in elliptic curve methods. With this Alice generates (a) and Bob generates (b). We select a point on a curve (G), and Alice generates aG, and Bob generates bG. They pass the values to each other, and then Alice received bG, and Bob receives aG. Alice multiplies by a, to get abG, and Bob will multiply by b, and also get abG. This will be their shared key.

C.1 Copy and paste the code from:

🕮 **Web link (ECDH):** https://asecuritysite.com/encryption/ecdh2

and confirm that Bob and Alice will always get the same shared key.

D What I should have learnt from this lab?

The key things learnt:

- The basics of the Diffie Hellman method.
- The basic method used with ECDH.

Notes

To setup your Python to run Python 2.7:

```
sudo update-alternatives --set python /usr/bin/python2.7
```

To install a Python library use:

```
easy_install libname
```

or:

```
pip install libname
```