

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Belgaum-590018**



**A Mini Project Report  
on**

**“Diffie–Hellman key exchange”**

**Submitted in Partial fulfillment of the Requirements for the VI Semester of the**

**Degree of**

**Bachelor of Engineering  
In**

**Computer Science & Engineering**

**By**

**Suman Kumar  
(1CR15CS193)**

**Subhashis Gupta  
(1CR15CS157)**

**Yash Purohit  
(1CR15CS184)**

**Under the Guidance of**

**Mrs. Swetha K V  
Asst. Professor, Dept. of CSE**

# ABSTRACT

## **Problem Statement:**

Write a Java or C program to implement Diffie–Hellman key exchange Algorithm.

## **Description:**

Diffie-Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication. That's an important distinction: You're not sharing information during the key exchange, you're creating a key together.

# ACKNOWLEDGEMENT

We have immense pleasure in successful completion of this project on "**Diffie-Hellman**" based on the subject "**CRYPTOGRAPHY AND NETWORK SECURITY**". The special environment at CMRIT, Bangalore always supports educational activities that facilitated our work in this project.

I would like to express my special thanks of gratitude to our **Principal Dr. Sanjay Jain** as well as our **HOD Dr. Jhansi Rani** who gave me the golden opportunity to do this wonderful project.

We acknowledge the support, the encouragement extended for this project by our guide **Mrs. Swetha K V.**

We thank our dearest parents, who encouraged us to extend our reach. With their help and support, we have been able to complete the project.

# TABLE OF CONTENTS

<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Introduction to the project.....	1
1.2 Purpose of the project .....	1
<b>CHAPTER 2 : SYSTEM REQUIREMENTS .....</b>	<b>2</b>
2.1 Software Requirements .....	2
2.2 Hardware Requirements.....	2
<b>CHAPTER 3 : DESIGN .....</b>	<b>3</b>
<b>CHAPTER 4 : IMPLEMENTATION .....</b>	<b>7</b>
<b>CHAPTER 5 : DISCUSSION AND SCREENSHOTS .....</b>	<b>19</b>
5.1 Source Code .....	1
5.2 Output .....	1
<b>CHAPTER 6 : CONCLUSIONS AND FUTURE SCOPE .....</b>	<b>27</b>
<b>BIBLIOGRAPHY .....</b>	<b>28</b>

## LIST OF FIGURES AND TABLES

5.1 Decryption Code .....	3
---------------------------	---

## Chapter 1

# INTRODUCTION

## 1.1 INTRODUCTION TO THE PROJECT

Diffie–Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. The following conceptual diagram illustrates the general idea of the key exchange by using colors instead of very large numbers.

The process begins by having the two parties, Alice and Bob, agree on an arbitrary starting color that does not need to be kept secret (but should be different every time[8]); in this example the color is yellow. Each of them selects a secret color that they keep to themselves. In this case, orange and blue-green. The crucial part of the process is that Alice and Bob now mix their secret color together with their mutually shared color, resulting in orange-tan and light-blue mixtures respectively, then publicly exchange the two mixed colors. Finally, each of the two mix together the color they received from the partner with their own private color. The result is a final color mixture yellow-brown that is identical to the partner's color mixture.

If a third party listened to the exchange, it would be computationally difficult for them to determine the secret colors. In fact, when using large numbers rather than colors, this action is computationally expensive for modern supercomputers to do in a reasonable amount of time.

## 1.2 PURPOSE OF THE PROJECT

The Diffie-Hellman protocol is a method for two computer users to generate a shared private key with which they can then exchange information across an insecure channel. Let the users be named Alice and Bob. First, they agree on two prime numbers  $g$  and  $p$ , where  $p$  is large (typically at least 512 bits) and  $g$  is a primitive root modulo  $p$ . (In practice, it is a good idea to choose  $p$  such that  $(p-1)/2$  is also prime.) The numbers  $g$  and  $p$  need not be kept secret from other users. Now Alice chooses a large random number  $a$  as her private key and Bob similarly chooses a large number  $b$ . Alice then computes  $A = g^a \pmod{p}$ , which she sends to Bob, and Bob computes  $B = g^b \pmod{p}$ , which he sends to Alice.

Now both Alice and Bob compute their shared key  $K = g^{(ab)} \pmod{p}$ , which Alice computes as

$$K = B^a \pmod{p} = (g^b)^a \pmod{p}$$

and Bob computes as

$$K = A^b \pmod{p} = (g^a)^b \pmod{p}.$$

Alice and Bob can now use their shared key  $K$  to exchange information without worrying about other users obtaining this information. In order for a potential eavesdropper (Eve) to do so, she would first need to obtain  $K = g^{(ab)} \pmod{p}$  knowing only  $g$ ,  $p$ ,  $A = g^a \pmod{p}$  and  $B = g^b \pmod{p}$ .

This can be done by computing  $a$  from  $A = g^a \pmod{p}$  or  $b$  from  $B = g^b \pmod{p}$ . This is the discrete logarithm

## **Chapter 2**

# **SYSTEM REQUIREMENTS**

## **2.1 SOFTWARE REQUIREMENTS**

1. Text editor
2. Python and JSON

## **2.2 HARDWARE REQUIREMENTS**

1. RAM : 512 Mb
2. HARDDISK SPACE : 5GB
3. SYSTEM TYPE : 32-BIT OR 64-BIT PROCESSOR

## Chapter 3

# DESIGN

### DIFFIE HELLMAN ALGORITHM:

Diffie-Hellman key exchange, also called exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming.

To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers  $p$  and  $q$ , such that  $p$  is a prime number and  $q$  is a generator of  $p$ . The generator  $q$  is a number that, when raised to positive whole-number powers less than  $p$ , never produces the same result for any two such whole numbers. The value of  $p$  may be large but the value of  $q$  is usually small.

Language Used: Python



## Chapter 4

### IMPLEMENTATION

Alice Side code:

```
prime = 1
while True:
    prime = random.randint(100,1000)
    if isPrime(prime) == True:
        break
    else :
        prime = prime
while True:
    generator = random.randint(2,prime-2)
    if isPrime(generator) == True:
        break
    else :
        generator = generator
print "prime =",prime    #prime
print "generator =",generator    #generator
a = random.randint(1,35535)
tcp = socket(AF_INET,SOCK_STREAM)
tcp.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
tcp.bind(('localhost',2222))
tcp.listen(5)
TEMP = { }
FOO = "
BAR = "
TEMP = {'p':prime,'base':generator}
conn, addr = tcp.accept()
while 1:
```

```
msg = conn.recv(1024)
if "NEGOCIATION" in msg:
    conn.send(json.dumps(TEMP))
    print "Sending base and prime number..."
    break
A = (generator ** a)% prime
print "the shared secret of alice is: %d " % (A)
msg = conn.recv(1024)
FOO = json.loads(msg)
B = FOO['B']
s = (B ** a)% prime
TEMP = {'A':A}
conn.send(json.dumps(TEMP))
print("wait for calculations to be completed !!!<<<<<<>>>>>>")
time.sleep(3)
print "The secret is %d " % (s)
msg =conn.recv(1024)
BAR =json.loads(msg)
secret2= BAR['S']
if s == secret2 :
    print " keys are verified successfully ! "
```

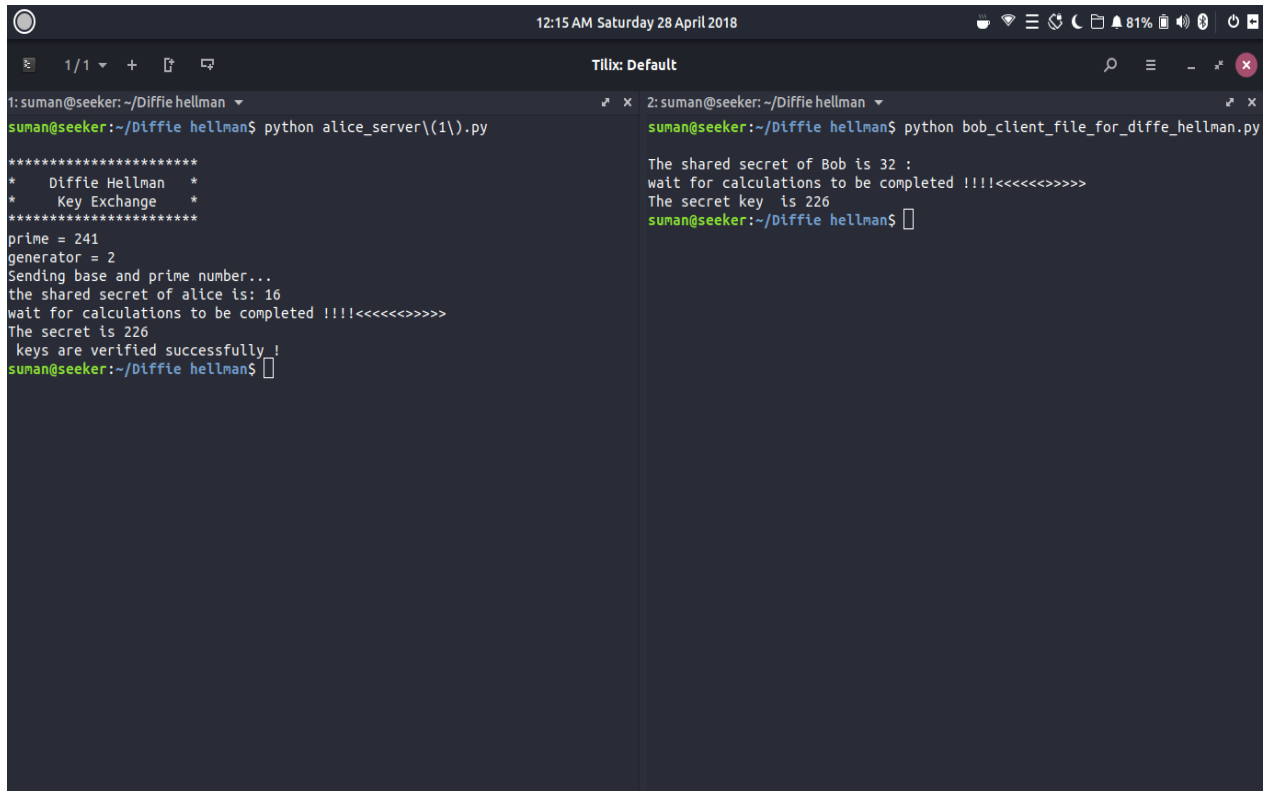
### Bob Side code:

```
a = random.randint(1,35535)
p = TEMP['p']
g = TEMP['base']
B = (int(g)**a)%int(p)
print "The shared secret of Bob is %d : " % (B)
DICT = {'B':B}
tcp.send(json.dumps(DICT))
msg = tcp.recv(1024)
TEMP = json.loads(msg)
A = TEMP['A']
s = (A**a)%p
print("wait for calculations to be completed !!!!<<<<<<>>>>>>")
time.sleep(3)
print "The secret key is %d " % (s)
SECT = {'S':s}
tcp.send(json.dumps(SECT))
```

## Chapter 5

# DISCUSSIONS AND SCREENSHOTS

### 5.1 Demo of differ-hellman



```
12:15 AM Saturday 28 April 2018
Tilix: Default
1: suman@seeker: ~/Diffie hellman
suman@seeker:~/Diffie hellman$ python alice_server\1\).py
*****
*   Diffie Hellman   *
*   Key Exchange     *
*****
prime = 241
generator = 2
Sending base and prime number...
the shared secret of alice is: 16
wait for calculations to be completed !!!!<<<<<<<<<<>>>>
The secret is 226
keys are verified successfully !
suman@seeker:~/Diffie hellman$

2: suman@seeker: ~/Diffie hellman
suman@seeker:~/Diffie hellman$ python bob_client_file_for_diffe_hellman.py
The shared secret of Bob is 32 :
wait for calculations to be completed !!!!<<<<<<<<<<>>>>
The secret key is 226
suman@seeker:~/Diffie hellman$
```

Fig 5.1 Bob and Alice key exchange

## Chapter 6

# CONCLUSION

Ideally, Diffie-Hellman should be used in conjunction with a recognized authentication method such as digital signatures to verify the identities of the users over the public communications medium. Diffie-Hellman is well suited for use in data communication but is less often used for data stored or archived over long periods of time.

---

## FUTURE SCOPE

### Encryption

Public key encryption schemes based on the Diffie–Hellman key exchange have been proposed. The first such scheme is the ElGamal encryption. A more modern variant is the Integrated Encryption Scheme.

### Forward secrecy

Protocols that achieve forward secrecy generate new key pairs for each session and discard them at the end of the session. The Diffie–Hellman key exchange is a frequent choice for such protocols, because of its fast key generation.

### Password-authenticated key agreement

When Alice and Bob share a password, they may use a password-authenticated key agreement (PK) form of Diffie–Hellman to prevent man-in-the-middle attacks. One simple scheme is to compare the hash of  $s$  concatenated with the password calculated independently on both ends of channel. A feature of these schemes is that an attacker can only test one specific password on each iteration with the other party, and so the system provides good security with relatively weak passwords. This approach is described in ITU-T Recommendation X.1035, which is used by the G.hn home networking standard.

---

## BIBLIOGRAPHY

- [1] CRYPTOGRAPHY, NETWORK SECURITY AND CYBER LAWS by  
BERNARD L.MENZES and RAVINDER KUMAR
- [2] <https://en.wikipedia.org/wiki/Diffe-hellman>