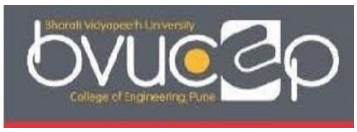
BHARATI VIDYAPEETH DEEMED UNIVERSITY
COLLEGE OF ENGINEERING, PUNE

Lab Manual
Network Security and Cryptography
(Course 2014)



DEPARTMENT OF COMPUTER ENGINEERING

VISION OF THE INSTITUTE

"To be World Class Institute for Social Transformation through Dynamic Education"

MISSION OF THE INSTITUTE

- To provide quality technical education with advanced equipments, qualified faculty members, infrastructure to meet needs of profession and society.
- To provide an environment conductive to innovation, creativity, research and entrepreneurial leadership.
- To practice and promote professional ethics, transparency and accountability for social community, economic and environmental conditions.

VISION OF THE DEPARTMENT

"To pursue and excel in the endeavour for creating globally recognized computer engineers through quality education".

MISSION OF THE DEPARTMENT

- To impart engineering knowledge and skills confirming to a dynamic curriculum.
- To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.
- To produce qualified graduates exhibiting societal and ethical responsibilities in working environment.

PROGRAM EDUCATIONAL OBJECTIVES

Graduates upon completion of B. Tech Computer Engineering Programme will able to:

1. Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.

2. Learn, Practice, and grow as skilled professionals/ entrepreneur/researchers adapting to the evolving computing landscape.
3. Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

PROGRAM OUTCOMES

1. To apply knowledge of computing and mathematics appropriate to the domain.
2. To logically define, analyse and solve real world problems.
3. To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs.
4. To interpret and analyse data for providing solutions to complex engineering problems.
5. To use and practise engineering and IT tools for professional growth.
6. To understand and respond to legal and ethical issues involving the use of technology for societal benefits.
7. To develop societal relevant projects using available resources.
8. To exhibit professional and ethical responsibilities.
9. To work effectively as an individual and a team member within the professional environment.
10. To prepare and present technical documents using effective communication skills.
11. To demonstrate effective leadership skills throughout the project management life cycle.
12. To understand the significance of lifelong learning for professional development

GENRAL INSTUCTIONS:

- Equipment in the lab is meant for the use of students. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care.
- Students are required to carry their reference materials, files and records with completed assignment while entering the lab.
- Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
- All the students should perform the given assignment individually.

- Lab can be used in free time/lunch hours by the students who need to use the systems should take prior permission from the lab in-charge.
- All the Students are instructed to carry their identity cards when entering the lab.
- Lab files need to be submitted on or before date of submission.
- Students are not supposed to use pen drives, compact drives or any other storage devices in the lab.
- For Laboratory related updates and assignments students should refer to the notice board in the Lab.

COURSE NAME: ***NETWORK SECURITY AND CRYPTOGRAPHY***

WEEKLY PLAN:

Week No.	Practical/Assignment Name	Problem Definition
1	Cryptography based Security Tools	Introduction to Cryptography based Security Tools
2	Symmetric Encryption	Write a Program in C/Java to implement symmetric encryption.
3	Asymmetric Encryption	Write a Program in C/Java to implement asymmetric encryption.
4	GnuPG encryption system	Introduction to GnuPG encryption system.
5	Decryption techniques	Implementation of Decryption techniques using secret key in GnuPG.
6	Implementation cryptographic algorithms using HashCalc.	Implementation of various cryptographic algorithms using HashCalc.
7	Firewall	Study of how Firewall works in computing.
8	Antivirus	Study of how Antivirus works according to offline or online mode.
9	Mini project on Antivirus	Implement mini project to develop antivirus application.
10	Case study	Case study on cyber security.

EXAMINATION SCHEME

Practical Exam: 25 Marks
Term Work: 25 Marks
Total: 50 Marks
Minimum Marks required: 20 Marks

PROCEDURE OF EVALUATION

Each practical/assignment shall be assessed continuously on the scale of 25 marks. The distribution of marks as follows.

Sr. No	Evaluation Criteria	Marks for each Criteria	Rubrics
1	**Timely Submission**	**07**	➢ Punctuality reflects the work ethics. Students should reflect that work ethics by completing the lab assignments and reports in a timely manner without being reminded or warned.
2	**Presentation**	**06**	➢ Student are expected to write the technical document (lab report) in their own words. The presentation of the contents in the lab report should be complete, unambiguous, clear, understandable. The report should document approach/algorithm/design and code with proper explanation.
3	**Understanding**	**12**	➢ Correctness and Robustness of the code is expected. The Learners should have an in-depth knowledge of the practical assignment performed.The learner should be able to explain methodology used for designing and developing the program/solution. He/she should clearly understand the purpose of the assignment and its outcome.

LABORATORY USAGE

Students use this lab which houses computers with the configuration:---------- ---------- for executing the lab experiments, document the results and to prepare the technical documents for making the lab reports.

OBJECTIVE

The objective of this lab is to make students aware about various cryptography based security tools and implementation of symmetric and asymmetric encryption. This lab also includes GnuPG so as the students also learn the encryption and decryption system in GnuPG.

PRACTICAL PRE-REQUISITE

- C and JavaProgramming Language Skills
- Linux

SOFTWARE REQUIREMENTS

- C compiler.
- JDK
- GnuPG

COURSE OUTCOMES

1. Understand the basics of network security.
2. Learn different techniques of cryptography.
3. Discuss details of key and certificate management.
4. Learn about system security.
5. Recite Network and Transport Layer security.
6. Apply knowledge of network security and cryptography in real life.

HOW OUTCOMES ARE ASSESSED?

Outcome	Assignment Number	Level	Proficiency evaluated by
Understand the basics of network security.	1,7,8,9	3,3,3,3	Performing Practical and reporting results
Learn different techniques of cryptography.	2,3,4,5,6	3,3,3,3,3	Problem definition&Performing Practical and reporting results
Discuss details of key and certificate management.	4,5,6	2,3,3	Performing experiments and reporting results
Learn about system security.	1,7,8,9	2,3,3,2	Performing experiments and reporting results
Recite Network and Transport Layer security.	2,3	2,2	Performing experiments and reporting results

Apply knowledge of network security and cryptography in real life.	1,8,9,10	2,3,3,3	Performing experiments and reporting results

DESIGN EXPERIENCE GAINED

The students gain moderate design experience by understanding various cryptographic algorithms and convert that algorithm to executable code.

LEARNING EXPERIENCE GAINED

The students learn both soft skills and technical skills while they are undergoing the practical sessions. The soft skills gained in terms of communication, presentation and behaviour. While technical skills they gained in terms of various cryptographic algorithms and their implementation. And also, students will learn about decryption and encryption system in GnuPG.

LIST OF PRACTICAL ASSIGNMENTS:

1. Introduction to Cryptography based Security Tools
2. Write a Program in C/Java to implement symmetric encryption.
3. Write a Program in C/Java to implement asymmetric encryption.
4. Introduction to GnuPG encryption system.
5. Implementation of Decryption techniques using secret key in GnuPG.
6. Implementation of various cryptographic algorithms using HashCalc.
7. Study of how Firewall works in computing.
8. Study of how Antivirus works according to offline or online mode.
9. Implement mini project to develop antivirus application.
10. Case study on cyber security.

Experiment 1

Title: Introduction to Cryptography based Security Tools

Theory:

Cryptography:

Cryptography is the science to encrypt and decrypt data that enables the users to store sensitive information or transmit it across insecure networks so that it can be read only by the intended recipient.

Data which can be read and understood without any special measures is called plaintext, while the method of disguising plaintext in order to hide its substance is called encryption.

Encrypted plaintext is known as cipher text and process of reverting the encrypted data back to plain text is known as decryption.

- The science of analysing and breaking secure communication is known as cryptanalysis. The people who perform the same also known as attackers.
- Cryptography can be either strong or weak and the strength is measured by the time and resources it would require to recover the actual plaintext.
- Hence an appropriate decoding tool is required to decipher the strong encrypted messages.
- There are some cryptographic techniques available with which even a billion computers doing a billion checks a second, it is not possible to decipher the text.

A cryptographic algorithm works in combination with a key (can be a word, number, or phrase) to encrypt the plaintext and the same plaintext encrypts to different cipher text with different keys.

Hence, the encrypted data is completely dependent couple of parameters such as the strength of the cryptographic algorithm and the secrecy of the key.

Cryptography Techniques:

Symmetric Encryption − Conventional cryptography, also known as conventional encryption, is the technique in which only one key is used for both encryption and decryption. For example, DES, Triple DES algorithms, MARS by IBM, RC2, RC4, RC5, RC6.

Asymmetric Encryption − It is Public key cryptography that uses a pair of keys for encryption: a public key to encrypt data and a private key for decryption. Public key is published to the people while keeping the private key secret. For example, RSA, Digital Signature Algorithm (DSA), Elgamal.

Hashing − Hashing is ONE-WAY encryption, which creates a scrambled output that cannot be reversed or at least cannot be reversed easily. For example, MD5 algorithm. It is used to create

Digital Certificates, Digital signatures, Storage of passwords, Verification of communications, etc.

OpenSSH/PuTTY/SSH

SSH (Secure Shell) is the now ubiquitous program for logging into or executing commands on a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network, replacing the hideously insecure telnet/rlogin/rsh alternatives. Most UNIX users run the open source OpenSSH server and client. Windows users often prefer the free PuTTY client, which is also available for many mobile devices, and WinSCP. Other Windows users prefer the nice terminal-based port of OpenSSH that comes with Cygwin.

GnuPG

PGP is the famous encryption system originally written by Phil Zimmerman which helps secure your data from eavesdroppers and other risks. GnuPG is a very well-regarded open source implementation of the PGP standard (the actual executable is named gpg). While the excellent GnuPG is always free, PGP is now owned by Symantec and costs a lot of money.

OpenSSL

The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general-purpose cryptography library. Apart from being a component of many crypto programs, OpenSSL comes with a lot of command-line tools for encryption, hashing, certificate handling, and more

Tor

Tor is a network of virtual tunnels designed to improve privacy and security on the Internet by routing your requests through a series of intermediate machines. It uses a normal proxy server interface so that ordinary Internet applications like web browsers and chat programs can be configured to use it. In addition to helping preserve users' anonymity, Tor can help evade firewall restrictions. Tor's hidden services allow users publish web sites and other services without revealing their identity or location. For a free cross-platform GUI, users recommend Vidalia. Remember that Tor exit nodes are sometimes run by malicious parties and can sniff your traffic, so avoid authenticating using insecure network protocols (such as non-SSL web sites and mail servers).

OpenVPN

OpenVPN is an open-source SSL VPN package which can accommodate a wide range of configurations, including remote access, site-to-site VPNs, WiFi security, and enterprise-scale remote access solutions with load balancing, failover, and fine-grained access-controls. OpenVPN implements OSI layer 2 or 3 secure network extension using the industry standard SSL/TLS protocol, supports flexible client authentication methods based on certificates, smart cards, and/or 2-factor authentication, and allows user or group-specific access control policies using firewall rules applied to the VPN virtual interface. OpenVPN uses OpenSSL as its primary cryptographic library.

Questions:

1. What are the various security principles?
2. Define cryptography.
3. State some cryptography-based security tools.

Experiment 2

Title: Write a Program in C/Java to implement symmetric encryption.

Theory:

Symmetric key cryptography is any cryptographic algorithm that is based on a shared key that is used to encrypt or decrypt text/cyphertext, in contract to asymmetric key cryptography, where the encryption and decryption keys are different.

Establishing the shared key is difficult using only symmetric encryption algorithms, so in many cases, an asymmetric encryption is used to establish the shared key between two parties.

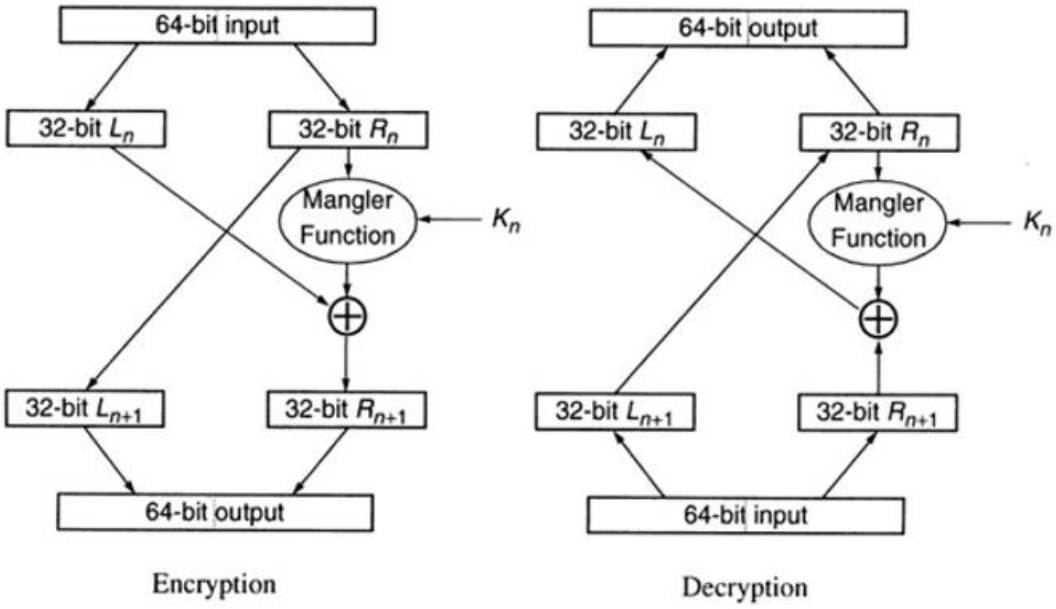
Examples for symmetric key cryptography include AES, DES, and 3DES. Key exchange protocols used to establish a shared encryption key include Diffie-Hellman (DH), elliptic curve (EC) and RSA.

Implementation of symmetric encryption algorithm

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are used for parity checks. The key therefore has a "useful" length of 56 bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on 64 bits and made of 16 blocks of 4 bits, generally denoted k1 to k16. Given that "only" 56 bits are actually used for encrypting, there can be 256 different keys.

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit blocks
- Initial permutation of blocks
- Breakdown of the blocks into two parts: left and right, named L and R
- Permutation and substitution steps repeated 16 times
- Re-joining of the left and right parts then inverse initial permutation



Algorithm:

STEP-1: Read the 64-bit plain text.

STEP-2: Split it into two 32-bit blocks and store it in two different arrays.

STEP-3: Perform XOR operation between these two arrays.

STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

```
import des.algo.DesAlgorithm;
import des.algo.DesAlgorithmImpl;
import des.constants.DesConstants;
import des.enums.Algorithm;
import des.enums.ChipperMode;
```

```
import des.enums.DesPaddingMode;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

import javax.crypto.*;

import javax.xml.bind.DatatypeConverter;

import java.security.*;

import java.util.Scanner;

public class DesApp {

DesAlgorithmdesAlgorithm = new DesAlgorithmImpl();

    public String encrypt(String dataToEncrypt, SecretKey key, String algo) throws IllegalBlockSizeException,
InvalidKeyException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException,
InvalidAlgorithmParameterException {

        return desAlgorithm.encrypt(dataToEncrypt, key, algo);

    }

    public String decrypt(String dataToDecrypt, SecretKey key, String algo) throws NoSuchAlgorithmException,
BadPaddingException, NoSuchPaddingException, IllegalBlockSizeException, InvalidKeyException,
InvalidAlgorithmParameterException {

        return desAlgorithm.decrypt(dataToDecrypt, key, algo);

    }

    public String encrypt(String dataToEncrypt, byte[] key, String algo) throws IllegalBlockSizeException,
InvalidKeyException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException,
InvalidAlgorithmParameterException {

        return desAlgorithm.encrypt(dataToEncrypt, key, algo);

    }

    public String decrypt(String dataToDecrypt, byte[] key, String algo) throws NoSuchAlgorithmException,
BadPaddingException, NoSuchPaddingException, IllegalBlockSizeException, InvalidKeyException,
InvalidAlgorithmParameterException {

        return desAlgorithm.decrypt(dataToDecrypt, key, algo);

    }

    public static void main(String[] args) throws NoSuchAlgorithmException, InvalidKeyException,
BadPaddingException, NoSuchPaddingException, IllegalBlockSizeException,
InvalidAlgorithmParameterException {

Security.addProvider(new BouncyCastleProvider());

SecretKeysecretKey = getSecretKey(Algorithm.DES.getValue());

System.out.println("Secret Key : " + DatatypeConverter.printHexBinary(secretKey.getEncoded()));

DesAppdesApp = new DesApp();

System.out.println("Enter your data to encrypt : ");

        Scanner scanner = new Scanner(System.in);

        String dataToEncrypt = scanner.next();
```

```
System.out.println("Original data : " + DatatypeConverter.printHexBinary(dataToEncrypt.getBytes()));
        String encryptedData = desApp.encrypt(dataToEncrypt, DesConstants.baseKey,
getAlgo(Algorithm.DES.getValue(),ChipperMode.CBC.getValue(),DesPaddingMode.PKCS5_PADDING.getValue
()));
System.out.println("Encrypted Data : " + encryptedData);
        String decryptedData = desApp.decrypt(encryptedData, DesConstants.baseKey,
getAlgo(Algorithm.DES.getValue(),ChipperMode.CBC.getValue(),DesPaddingMode.PKCS5_PADDING.getValue
()));
System.out.println("DecryptedData : " + decryptedData);
        System.out.println("################################################################################");
System.out.println("Original data : " + DatatypeConverter.printHexBinary(dataToEncrypt.getBytes()));
encryptedData = desApp.encrypt(dataToEncrypt, secretKey,
getAlgo(Algorithm.DES.getValue(),ChipperMode.CBC.getValue(),DesPaddingMode.PKCS5_PADDING.getValue
()));
System.out.println("Encrypted Data : " + encryptedData);
decryptedData = desApp.decrypt(encryptedData, secretKey,
getAlgo(Algorithm.DES.getValue(),ChipperMode.CBC.getValue(),DesPaddingMode.PKCS5_PADDING.getValue
()));
System.out.println("DecryptedData : " + decryptedData);
    }
    private static String getAlgo(String algo,StringcipherMode, String paddingMode) {
        return algo + "/" + cipherMode + "/" + paddingMode ;
    }
    private static SecretKeygetSecretKey(String algo) throws NoSuchAlgorithmException {
KeyGeneratorkeyGenerator = KeyGenerator.getInstance(algo);
SecureRandomsecureRandom = new SecureRandom();
        int keyBitSize = 56;
keyGenerator.init(keyBitSize, secureRandom);
        return keyGenerator.generateKey();
    }
}
```

Questions:

1. What is the symmetric key cryptography?
2. Discuss some symmetric key cryptography algorithms.
3. What is the size of key used for encryption in DES algorithm?

Experiment 3

Title: Write a Program in C/Java to implement asymmetric encryption.

Theory:

Asymmetric encryption is also referred to as public key encryption. In asymmetric encryption, both the encrypting and decrypting systems have a set of keys. One is called the public key, and

another is called the private key. If the message is encrypted with one key in the pair, the message can be decrypted only with the other key in the pair.

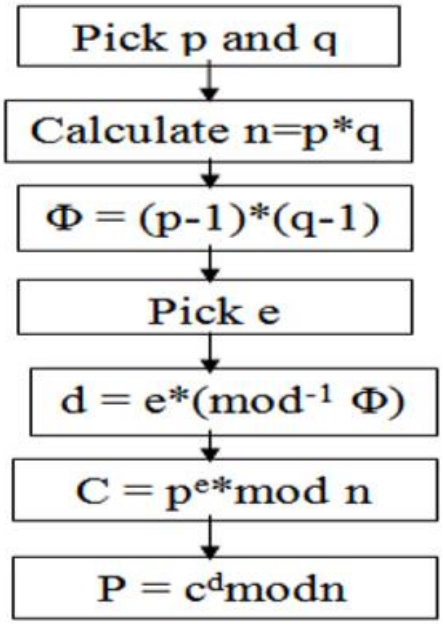
Asymmetric key algorithms are not quite as fast as symmetric key algorithms. This is partially due to the fact that asymmetric key algorithms are generally more complex, using a more sophisticated set of functions.

Diffie-Hellman key exchange: The Diffie-Hellman algorithm was one of the earliest known asymmetric key implementations. The Diffie-Hellman algorithm is mostly used for key exchange. Although symmetric key algorithms are fast and secure, key exchange is always a problem. You have to figure out a way to get the private key to all systems. The Diffie-Hellman algorithm helps with this. The Diffie-Hellman algorithm will be used to establish a secure communication channel. This channel is used by the systems to exchange a private key. This private key is then used to do symmetric encryption between the two systems.

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all integer m:

$$(m^e)^d = m \pmod n$$

The public key is represented by the integers n and e; and, the private key, by theinteger d. m represents the message. RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.



Algorithm:

STEP-1: Select two co-prime numbers as p and q.
STEP-2: Compute n as the product of p and q.
STEP-3: Compute (p-1)*(q-1) and store it in z.
STEP-4: Select a random prime number e that is less than that of z.
STEP-5: Compute the private key, d as e * mod^{-1}(z).
STEP-6: The cipher text is computed as $message^{e}$ * mod n.
STEP-7: Decryption is done as $cipher^{d}$mod n.

Questions:

1. How asymmetric key cryptography is different from symmetric key cryptography?
2. State basic steps in RSA algorithm.

Experiment 4

Title: Introduction to GnuPG encryption system.

Theory:

GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as PGP). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. GnuPG also provides support for S/MIME and Secure Shell (ssh).

Since its introduction in 1997, GnuPG is Free Software (meaning that it respects your freedom). It can be freely used, modified and distributed under the terms of the GNU General Public License .

GnuPG is an implementation of PGP (Pretty Good Privacy), which is a form of public key/private key encryption. The strength of the encryption comes from the fact that a file can be encrypted for a given recipient using only the public key, yet both keys are needed in order to decrypt the file. So the idea is to give your public key to your friends and colleagues, but keep the private key closely guarded. With GnuPG the keys are associated with an ID consisting of a name, comment and e-mail address. When specifying recipients you may use either the name or the e-mail address but due to the complexity of dealing with the spaces in the names we will be using the e-mail address.
The private key is additionally locked with a passphrase which is required to access it. This adds an additional level of security to prevent someone using your private key if they gain physical access to both your computer and account.

Installation

GnuPG can be downloaded from the GnuPG homepage and you should follow the instructions relating to your own system. This page assumes you are using a Mac OS X or Linux based environment and Windows users would have to adjust accordingly (in fact this page would be more appropriate reading).
For installation on a Mac the easiest method is to install Fink and then issue the following command in your terminal:

```
yoda:~ ian$ sudo apt-get install gnupg
```

If you get a message about not being in the sudoers file make sure to add your username into /etc/sudoers using the root account. Many programs can be installed in this way with Fink and I highly recommend it for anyone who enjoys the UNIX aspect of OS X.

Verify your installation by typing the following command and checking that a path is returned as follows:

```
yoda:~ ian$ which gpg
/sw/bin/gpg
```

Questions:

1. What is GnuPG?
2. Explain the significance of Pretty Good Privacy.

Experiment 5

Title: Implementation of Decryption techniques using secret key in GnuPG.

Theory:

Generating Your Keys

Assuming you have successfully installed gpg you can go ahead and create a new key pair. This is accomplished by running gpg with the --gen-key switch as follows:

```
yoda:~ian$ gpg --gen-key
gpg (GnuPG) 1.2.4; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: /Users/ian/.gnupg: directory created
gpg: new configuration file `/Users/ian/.gnupg/gpg.conf' created
gpg: WARNING: options in `/Users/ian/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/Users/ian/.gnupg/secring.gpg' created
gpg: keyring `/Users/ian/.gnupg/pubring.gpg' created
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
              minimum keysizeis  768 bits
              default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
         0 = key does not expire
<n>  = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs the user id
from Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Ian Atkinson
Email address: ian@atkinson.co.uk
Comment: home
You selected this USER-ID:
    "Ian Atkinson (home) <ian@atkinson.co.uk>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
```

```
Enter passphrase: my passphrase
Enter passphrase: my passphrase

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
++++++++++.++++++++++++++++++++..++++++++++++
gpg: /Users/ian/.gnupg/trustdb.gpg: trustdb created
public and secret key created and signed.
key marked as ultimately trusted.

pub  1024D/0EB6F689 2006-04-06 Ian Atkinson (home) <ian@atkinson.co.uk>
     Key fingerprint = 908E 6C41 8689 7981 F6B4  CEDF A70C 0200 0EB6 F689
sub  1024g/71445CC9 2006-04-06
```

Throughout this process you can just press enter at any time to accept the default value in brackets.

The most important part of your key generation is choosing your passphrase. Your phrase should be something you won't forget (i.e. don't need to write down) and should contain both upper and lower case characters, numbers and punctuation. It should be of a sufficient length but not so long that it's going to take you a lot of attempts to type it in (remember you won't be able to see anything on the screen as you enter it). Something like eight to twelve words is probably about right for most people.

Now that this process is complete you have both a public and a private keyring. A keyring is simply a collection of keys as you might expect!

Your public keyring should just contain your own public key for now, you can view the keyring with the --list-keys option:

```
yoda:~ian$ gpg --list-keys
/Users/ian/.gnupg/pubring.gpg
------------------------------
pub  1024D/0EB6F689 2006-04-06 Ian Atkinson (home) <ian@atkinson.co.uk>
sub  1024g/71445CC9 2006-04-06
```

If anyone else were to give you their own public keys so that you could send them files, they would appear in here.

You can also check that your private keying contains your own private key with the --list-secret-keys option:

```
yoda:~ian$ gpg --list-secret-keys
/Users/ian/.gnupg/secring.gpg
------------------------------
sec  1024D/0EB6F689 2006-04-06 Ian Atkinson (home) <ian@atkinson.co.uk>
ssb  1024g/71445CC9 2006-04-06
```

Encrypting a File

Now that we have generated our keys we can go ahead and encrypt a file. For this example we will make a gpgtest directory in our home directory for our test file:

```
yoda:~ian$ mkdir ~/gpgtest
yoda:~ian$ cd ~/gpgtest
yoda:~/gpgtestian$ echo 'this is a secret!!!' >secret.txt
yoda:~/gpgtestian$ cat secret.txt
this is a secret!!!
```

Here we first create our directory and then change to it, we then create a simple text file containing line 'this is a secret!!!' and then test the contents of our file using cat.

For now you can see that the secret file is the only one we have in our directory:

```
yoda:~/gpgtestian$ ls
secret.txt
```

We will now go ahead and encrypt this file, specifying the recipient as ourselves (remember we can do this because we have our own address in our public keyring).

We will use the -e (encrypt) flag of gpg, along with the -r (recipient) flag and our recipient and finally the name of the file we wish to encrypt:

```
yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk secret.txt
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
yoda:~/gpgtestian$ ls
secret.txt
secret.txt.gpg
```

As you can see, using the -e option gpg has named the encypted file by simply appended .gpg to our clear text file. In many cases this will be suitable as it's easier to keep track of a file and it's encrypted equivalent if they have the same root file name. Of course, depending on the level of security you want to obtain, changing the file name and removing the .gpg extension is likely to draw less attention.

If we wanted we could specify the name of the output file by using the -o (output) option, for example we could save our encrypted file as encrypted.txt rather than secret.txt.gpg as follows:

```
yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk -o encrypted.txt secret.txt
```

This file is now encrypted, if you try using the cat command again you should just see a lot of random characters (this may make your terminal go weird though so either take my word for it or be prepared to launch a new one!).

It is worth noting that whilst we have used a plain text file for our example, we can encrypt other types of data of necessary, such as an image.

The encryption process will also hide the file type from the operating system since the headers become encrypted, further helping the privacy of our document. Notice in the following example how, when encypted, the PNG image file is no longer recognised as an image by the system and it simply 'data':

```
yoda:~/gpgtestian$ file picture.png
picture.png: PNG image data, 1319 x 968, 8-bit/color RGB, non-interlaced
yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk -o encrypted_picture.png picture.png
yoda:~/gpgtestian$ file encrypted_picture.png
encrypted_picture.png: data
```

Decrypting a File

Now that we have encrypted our file, we can also decrypt it. You may wish to decrypt a file for editing or of couse just to view the contents. We decrypt using the -d flag as follows:

```
yoda:~/gpgtestian$ gpg -d secret.txt.gpg

You need a passphrase to unlock the secret key for
user: "Ian Atkinson (home) <ian@atkinson.co.uk>"
1024-bit ELG-E key, ID 71445CC9, created 2006-04-06 (main key ID 0EB6F689)

Enter passphrase: my passphrase

gpg: encrypted with 1024-bit ELG-E key, ID 71445CC9, created 2006-04-06
      "Ian Atkinson (home) <ian@atkinson.co.uk>"
this is a secret!!!
```

Gpg will ask us for our passphrase, which you chose when generating your keys. If you type the phrase in successfully then the content of the file will be printed to stdout (the screen).
This isn't usually very useful, and certainly isn't for non text files, so what is more common is to output to a file in the same way as we did when encrypting using the -o flag. We can save the decrypted file as notasecret.txt as follows:

```
yoda:~/gpgtestian$ gpg -d -o notasecret.txt secret.txt.gpg
```

This is all the information you need in order to encrypt and decrypt files for yourself. You can encrypt any of your own sensitive data in this way and keep it safe without learning any further commands.
However, if you wish to encrypt files for other people, or have people encrypt files for you, then you must learn about importing and exporting public keys.

Sharing Keys

The first thing we need to do is export our public key into a text file that we can give to other people to add to their keyrings, this will let them encrypt files before sending them to us. Here we will use the --export-keys option along with the -a (armor) option.
The armor option will create a plain text file for us, this will be more useful shortly when we come to backing up our keys but for now it will let us see the contents of the keys we're exporting if we wish:

```
yoda:~ian$ gpg -a -o publickey.txt --export ian@atkinson.co.uk
```

This has now saved our public key into the file publickey.txt. Let us now see how to import this key onto another machine so that user may encrypt files for us (we could use the same process to add other people's keys to our keyring and encyrpt files for them). First we will see what happens if we try to encrypt the file without first having the correct key on our keyring:

```
ian@chewy:~>echo 'a secret from chewy' >chewysecret.txt
ian@chewy:~>cat chewysecret.txt
a secret from chewy
ian@chewy:~>gpg -e -r ian@atkinson.co.uk chewysecret.txt
gpg: ian@atkinson.co.uk: skipped: public key not found
gpg: chewysecret.txt: encryption failed: public key not found
```

Here we have created our new secret file on another computer, but when we have tried to encrypt the file gpg has told us that we can't as we don't have the key. What we need to do is import the key that we made earlier and try again:

```
ian@chewy:~>gpg --import publickey.txt
gpg: /home/ian/.gnupg/trustdb.gpg: trustdb created
gpg: key 34BEE591: public key "Ian Atkinson <ian@atkinson.co.uk>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

The next thing we need to do is set up the trust for the key, this tells gpg how much we trust the key owner

```
ian@chewy:~>gpg --edit-key ian@atkinson.co.uk
gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

pub  1024D/34BEE591  created: 2007-01-01  expires: never       usage: CS
                     trust: unknown       validity: unknown
sub  2048g/F6F6EA8F  created: 2007-01-01  expires: never       usage: E
[ unknown] (1). Ian Atkinson <ian@atkinson.co.uk>

Command>trust
pub  1024D/34BEE591  created: 2007-01-01  expires: never       usage: CS
                     trust: unknown       validity: unknown
```

```
sub  2048g/F6F6EA8F  created: 2007-01-01  expires: never       usage: E
[ unknown] (1). Ian Atkinson <ian@atkinson.co.uk>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 5
Do you really want to set this key to ultimate trust? (y/N) y

pub  1024D/34BEE591  created: 2007-01-01  expires: never       usage: CS
                trust: ultimate      validity: unknown
sub  2048g/F6F6EA8F  created: 2007-01-01  expires: never       usage: E
[ unknown] (1). Ian Atkinson <ian@atkinson.co.uk>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

Command>quit
```

Now that we have imported the key and set the trust level we can go ahead and encrypt the file without error:

```
ian@chewy:~>gpg -e -r ian@atkinson.co.uk chewysecret.txt
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
ian@chewy:~>ls chewysecret*
chewysecret.txt  chewysecret.txt.gpg
```

Finally, for completeness, let's go ahead and decrypt this file on the other computer to check everything worked OK (of course, we can't decrypt this file on the machine that we encrypted it on as demonstrated first because we don't have the private key there, only the public key):

```
ian@chewy:~>gpg -d chewysecret.txt.gpg
gpg: encrypted with 2048-bit ELG-E key, ID F6F6EA8F, created 2007-01-01
      "Ian Atkinson <ian@atkinson.co.uk>"
gpg: decryption failed: secret key not available
ian@chewy:~>scpchewysecret.txt.gpgian@yoda:
Password:
chewysecret.txt.gpg    100%  618     0.6KB/s   00:00
ian@chewy:~>sshian@yoda
Password:
Last login: Sat Jan 20 02:13:47 2007 from obi-wan.home
Welcome to Darwin!
yoda:~ian$ gpg -d chewysecret.txt.gpg

You need a passphrase to unlock the secret key for
user: "Ian Atkinson <ian@atkinson.co.uk>"

Enter passphrase: my passphrase

2048-bit ELG-E key, ID F6F6EA8F, created 2007-01-01 (main key ID 34BEE591)
```

```
gpg: encrypted with 2048-bit ELG-E key, ID F6F6EA8F, created 2007-01-01
      "Ian Atkinson <ian@atkinson.co.uk>"
a secret from chewy
```

Questions:

1. How the keys are generated in GnuPG?
2. Which commands are used for encryption?
3. Which commands are used for decryption?

Experiment 6

Title: Implementation of various cryptographic algorithms using HashCalc.

Theory:

HashCalc is a calculator program developed by SlavaSoft, Inc. It is used for computing HMACs, messages digests, and checksums for files. It can also be used for calculating hex strings and text strings. The program provides 13 of the most common checksum and hash algorithms.

The program's interface is straightforward. It offers a standard window that provides all the options straight from the main interface. To use it, the user only has to choose the file to be calculated and the desired cryptographic hash functions. The program also features a drag-and-drop support when importing the files to be calculated. Users can choose from nine cryptographic hash functions including MD4, MD5, SHA256, SHA512, SHA384, and more. Selecting to calculate the HMAC is also possible. When this is selected, users can specify the key format (hex string or text string) and the key. The HMAC checkbox should be unchecked when calculating checksums and digests. Users can calculate almost any type of file including documents, video games, movies, software, music, and others.

Major Features:

- Support of 12 well-known and documented hash and checksum algorithms: MD2, MD4, MD5, SHA-1, SHA-2(256, 384, 512), RIPEMD-160, PANAMA, TIGER, ADLER32, CRC32.
- Support of a custom hash algorithm (MD4-based) used in eDonkey and eMule applications.
- Support of 2 modes of calculations: HASH/CHECKSUM and HMAC.
- Support of 3 input data formats: files, text strings and hex strings.
- Work with large size files. (Tested on file sizes up to 15 GB).
- Drag-and-drop support.
- Quick and simple installation.
- Calculates hash/checksum and HMAC for files of any type: music, audio, sound, video, image, icon, text, compression, etc., with the extensions: .mp3, .wav, .avi, .mpg, .midi, .mov, .dvd, .ram, .zip, .rar, .ico, .gif, .pif, .pic, .tif, .tiff, .txt, .doc, .pdf, .wps, .dat, .dll, .hex, .bin, .iso, .cpp, .dss, .par, .pps, .cue, .ram, .md5, .sfv, etc.

Questions:

1. State the major features of HashCalc.
2. Which algorithms included within HashCalc?

Experiment 7

Title: Study of how Firewall works in computing.

Theory:

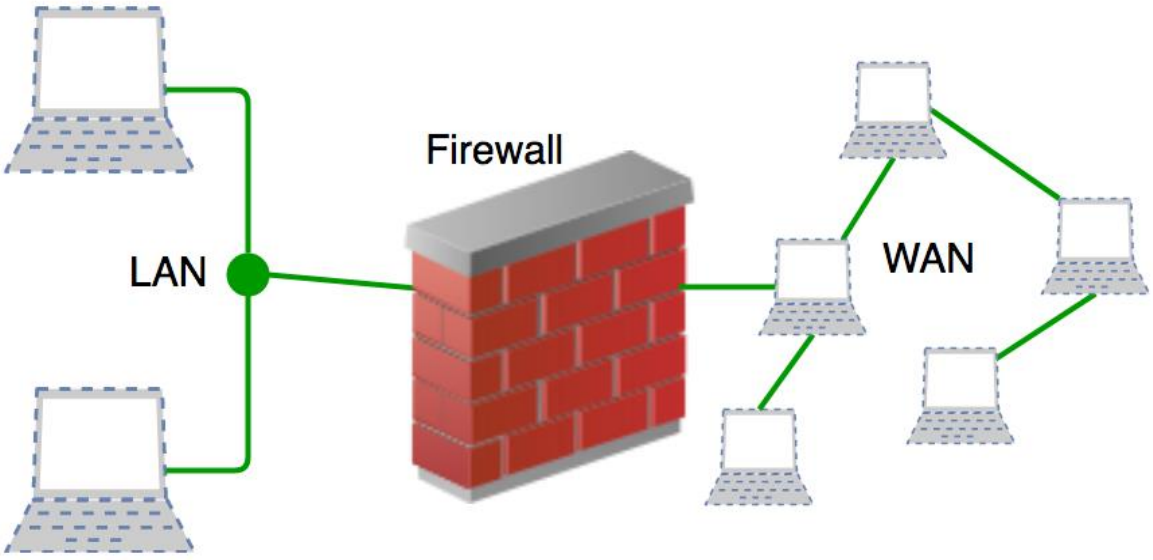
A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules it accepts, rejects or drops that specific traffic.

Accept : allow the traffic

Reject :block the traffic but reply with an "unreachable error"

Drop : block the traffic with no reply

A firewall establishes a barrier between secured internal networks and outside untrusted network, such as the Internet.



Before Firewalls, network security was performed by Access Control Lists (ACLs) residing on routers. ACLs are rules that determine whether network access should be granted or denied to specific IP address.

But ACLs cannot determine the nature of the packet it is blocking. Also, ACL alone does not have the capacity to keep threats out of the network. Hence, the Firewall was introduced.

Connectivity to the Internet is no longer optional for organizations. However, accessing the Internet provides benefits to the organization; it also enables the outside world to interact with the internal network of the organization. This creates a threat to the organization. In order to secure the internal network from unauthorized traffic, we need a Firewall.

How Firewall Works

Firewall match the network traffic against the rule set defined in its table. Once the rule is matched, associate action is applied to the network traffic. For example, Rules are defined as any employee from HR department cannot access the data from code server and at the same time another rule is defined like system administrator can access the data from both HR and technical

department. Rules can be defined on the firewall based on the necessity and security policies of the organization.
From the perspective of a server, network traffic can be either outgoing or incoming. Firewall maintains a distinct set of rules for both the cases. Mostly the outgoing traffic, originated from the server itself, allowed to pass. Still, setting a rule on outgoing traffic is always better in order to achieve more security and prevent unwanted communication.
Incoming traffic is treated differently. Most traffic which reaches on the firewall is one of these three major Transport Layer protocols- TCP, UDP or ICMP. All these types have a source address and destination address. Also, TCP and UDP have port numbers. ICMP uses *type code* instead of port number which identifies purpose of that packet.

Default policy: It is very difficult to explicitly cover every possible rule on the firewall. For this reason, the firewall must always have a default policy. Default policy only consists of action (accept, reject or drop).Suppose no rule is defined about SSH connection to the server on the firewall. So, it will follow the default policy. If default policy on the firewall is set to *accept*, then any computer outside of your office can establish an SSH connection to the server.
Therefore, setting default policy as *drop* (or reject) is always a good practice.

Types of Firewall

Firewalls are generally of two types: Host-based and Network-based.

1. **Host- based Firewalls:** Host-based firewall is installed on each network node which controls each incoming and outgoing packet. It is a software application or suite of applications, comes as a part of the operating system. Host-based firewalls are needed because network firewalls cannot provide protection inside a trusted network. Host firewall protects each host from attacks and unauthorized access.
2. **Network-based Firewalls:** Network firewall function on network level. In other words, these firewalls filter all incoming and outgoing traffic across the network. It protects the internal network by filtering the traffic using rules defined on the firewall. A Network firewall might have two or more network interface cards (NICs). A network-based firewall is usually a dedicated system with proprietary software installed.

Questions:

1. What is firewall?
2. Discuss types of firewall.
3. What is the use of Network based firewalls?

Experiment 8

Title: Study of how Antivirus works according to offline or online mode.

Theory:

Antivirus software scans the file comparing specific bits of code against information in its database and if it finds a pattern duplicating one in the database, it is considered a virus, and it will quarantine or delete that particular file.

Antivirus software, sometimes known as anti-malware software, is designed to detect, prevent and take action to disarm or remove malicious software from your computer such as viruses, worms and Trojan horses. It may also prevent or remove unwanted spyware and adware in addition to other types of malicious programs. The first versions of antivirus software can be traced as far back as the 1980s.

Antivirus software will begin by checking your computer programs and comparing them to known types of malware. It will also scan your computer for behavior that may signal the presence of a new, unknown malware. Typically, antivirus software uses all three scanning detection processes:

- **Specific Detection**– This works by looking for known malware by a specific set of characteristics.
- **Generic Detection**– This process looks for malware that are variants of known "families," or malware related by a common codebase.
- **Heuristic Detection** – This process scans for previously unknown viruses by looking for known suspicious behaviour or file structures.

All program files (executable) that enter a system go through the antivirus scan. Those that match the signatures are classified as viruses and are blacklisted. The other program files then pass through the Defense + HIPS (Host Intrusion Prevention System). Here the known files would be allowed entry and would run in the system while the unknown ones, irrespective of whether they are good or bad, are sent to the Defense+ Sandbox. These would be allowed to run, but only in this restricted environment. Those that the user allows as good files would be added to the Whitelist while all others would remain in the sandbox, after which they would go to the Comodo labs for analysis.

Question:

1. What is virus?
2. Explain Heuristic detection process.
3. Explain the working of antivirus software.

Experiment 9

Title:Implement mini project to develop antivirus application.

In this experiment the students have to implement project to develop antivirus application by using appropriate resources.

Experiment 10

Title: Case study on cyber security.

Theory:
In this experiment students have to study some cyber security related case studies.

Sample case study
Ransomware Aftershock: The Road To Recovery After A Cyber Data Hijack

Preparing Your Organization To Cope In A Similar Cyber-Attack Situation

The disclosure of ransomware attacks grows as U.S. municipalities and city services – all rich in data – find themselves the target of cyber hijacking. Beyond payment (or not) of the ransom, little has been shared about the cost for an organization to recover from this form of attack. Recent news from three entities is helping shed light on the recovery costs and on-going learnings from ransomware attacks.

City of Baltimore Discloses Data Loss From Ransomware Attack

Hackers successfully infiltrated systems operated by the City of Baltimore this past May. The attackers encrypted data files and demanded a ransom in exchange for the decryption keys. Mayor Bernard C. "Jack" Young refused to pay and IT leaders were instructed to rebuild the municipality's computer systems. City of Baltimore officials placed a price tag of $18 million on the estimated cost of the ransomware attack.

In August, city leaders voted to divert $6 million of parks and recreation funding to IT "cyber-attack remediation and hardening of the environment," according to the city's spending panel known as the Board of Estimates.

Now, Baltimore's auditor told city officials that IT performance data was lost during the attacks, according to reports in the Baltimore Sun. Without backups of the locally stored data, the auditor is unable to verify some claims made by the IT department. This is the first notification made by City of Baltimore than data loss occurred from the attack.

Questions:

1. Define cyber security.