### SREE NIDHI INSTITUTE OF SCIENCE & TECHNOLOGY

An Autonomous institution

Yamnampet, Ghatkesar, R.R. DIST.,Hyderabad– 501 301

**LAB MANUAL**

**FOR**

**INFORMATION SECURITY**

### FOR

**B.Tech (IT) IV – I SEMESTER**

****

### DEPARTMENT OF

### INFORMATION TECHNOLOGY

**2021**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | H | H |  | H |  |  |  |  |  |  |  |

Syllabus for B. Tech. IV Year I semester

Information Technology

**Information Security Lab**

|  |  |  |  |
| --- | --- | --- | --- |
| L | T | P | C |
| 0 | 0 | 3 | 1.5 |

**Code**: 7EC76

Course Objectives: To learn the fundamental of information security principles and services offered to secure the data.To apply conventional cryptographic techniques in order to do encryption.To apply Public key cryptography techniques in order to do encryption.

**Course Outcomes:**

At the end of this course, the student will be able to

1. Design and Implement symmetric key encryption algorithms
2. Simulate asymmetric key encryption algorithms.
3. Implement hashing and key exchange algorithms.
4. Simulate and execute Digital Signature and Digital envelope.
5. Install and execute various projects in NS3.

**List of Programs**

* 1. Implement Substitution Cipher.
  2. Implement Transposition Cipher.
  3. Implement DES

1. Generate Cipher text for the given Plaintext.
2. Retrieve the Plaintext from the given Ciphertext.
   1. Implement Diffie Hellman Algorithm and generate Secret Key.
   2. Implement RSA algorithm
3. Generate Public key and Private key pair
4. Generate Ciphertext for the Plaintext
5. Obtain the Plaintext from the Ciphertext
   1. Implement Hash Algorithm.
   2. Generate Digital Signature .
   3. Implement Digital Envelope.
   4. Installation of NS3.

10. Demonstration of NS3.

11. Executing simple projects in NS3.

**1. Implement Substitution Cipher.**

**Aim:** To Implement substitution cipher.

**Program**:

import java.io.\*;

import java.util.\*;

public class SubstitutionCipher

{

static Scanner sc=new Scanner(System.in);

static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

public static void main(String[] args)throws IOException

{

String a="abcdefghijklmnopqrstuvwxyz";

String b="zyxwvutsrqponmlkjihgfedcba";

System.out.println("Enter any string:");

String str=br.readLine();

String decrypt="";

char c;

for(int i=0;i<str.length();i++)

{

c=str.charAt(i);

int j=a.indexOf(c);

decrypt=decrypt+b.charAt(j);

{

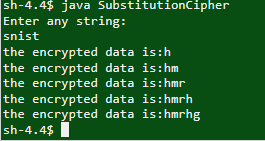
System.out.println("the encrypted data is:"+decrypt);

}

}

}

}

**OUTPUT**:

1. **Implement Transposition Cipher**

**Aim:**To understand the concept of Transposition cipher.

**Program**:

import java.util.\*;

class SimpleColumnar{

public static void main(String sap[]){

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter plaintext(enter in lower case): ");

String message = sc.next();

System.out.print("\nEnter key in numbers: ");

String key = sc.next();

/\* columnCount would keep track of columns \*/

int columnCount = key.length();

/\* rowCount will keep of track of rows...no of rows = (plaintextlength + keylength) / keylength \*/

int rowCount = (message.length()+columnCount)/columnCount;

/\*plainText and cipherText would be array containing ASCII values for respective alphabets \*/

int plainText[][] = new int[rowCount][columnCount];

int cipherText[][] = new int[rowCount][columnCount];

/\*Encryption Process\*/

System.out.print("\n-----Encryption-----\n");

cipherText = encrypt(plainText, cipherText, message, rowCount, columnCount, key);

// prepare final string

String ct = "";

for(int i=0; i<columnCount; i++)

{

for(int j=0; j<rowCount; j++)

{

if(cipherText[j][i] == 0)

ct = ct + 'x';

else{

ct = ct + (char)cipherText[j][i];

}

}

} System.out.print("\nCipher Text: " + ct);

/\*Decryption Process\*/

System.out.print("\n\n\n-----Decryption-----\n");

plainText = decrypt(plainText, cipherText, ct, rowCount, columnCount, key);

// prepare final string

String pt = "";

for(int i=0; i<rowCount; i++)

{

for(int j=0; j<columnCount; j++)

{

if(plainText[i][j] == 0)

pt = pt + "";

else{

pt = pt + (char)plainText[i][j];

}

}

}

System.out.print("\nPlain Text: " + pt);

System.out.println();

}

static int[][] encrypt(int plainText[][], int cipherText[][], String message, int rowCount, int columnCount, String key){

int i,j;

int k=0;

/\* here array would be filled row by row \*/

for(i=0; i<rowCount; i++)

{

for(j=0; j<columnCount; j++)

{

/\* terminating condition...as string length can be smaller than 2-D array \*/

if(k < message.length())

{

/\* respective ASCII characters would be placed \*/

plainText[i][j] = (int)message.charAt(k);

k++;

}

else

{

break;

}

}

}

/\* here array would be filled according to the key column by column \*/

for(i=0; i<columnCount; i++)

{

/\* currentCol would have current column number i.e. to be read...as there would be ASCII value stored in key so we would subtract it by 48 so that we can get the original number...and -1 would be subtract as array position starts from 0\*/

int currentCol= ( (int)key.charAt(i) - 48 ) -1;

for(j=0; j<rowCount; j++)

{

cipherText[j][i] = plainText[j][currentCol];

}

}

System.out.print("Cipher Array(read column by column): \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)cipherText[i][j]+"\t");

}

System.out.println();

}

return cipherText;

}

static int[][] decrypt(int plainText[][], int cipherText[][], String message, int rowCount, int columnCount, String key){

int i,j;

int k=0;

for(i=0; i<columnCount; i++)

{

int currentCol= ( (int)key.charAt(i) - 48 ) -1;

for(j=0; j<rowCount; j++)

{

plainText[j][currentCol] = cipherText[j][i];

}

}

System.out.print("Plain Array(read row by row): \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)plainText[i][j]+"\t");

}

System.out.println();

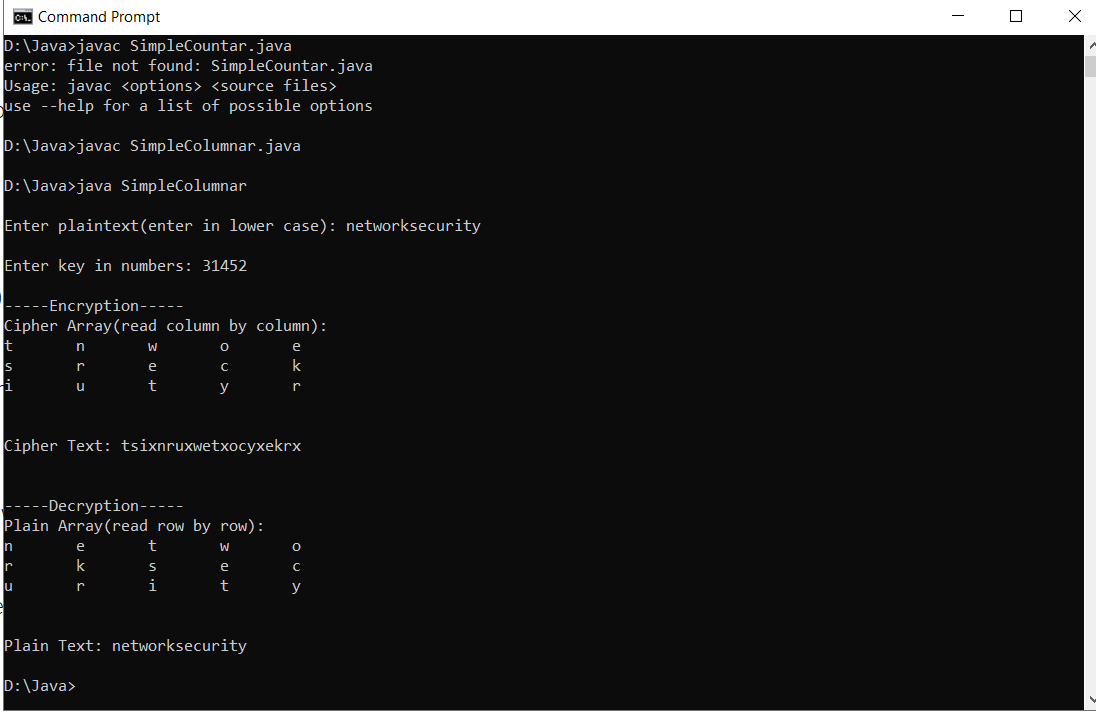
}

return plainText;

}

}

**output**:



**3.Implement DES**

**(a)Generate Cipher text for the given Plaintext.**

**(b)Retrieve the Plaintext from the given Ciphertext.**

**Aim**:To Understand the DES Algorithm .

**Program :**

import javax.swing.\*;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Random ;

class DES {

byte[] skey = new byte[1000];

String skeyString;

static byte[] raw;

String inputMessage, encryptedData,decryptedMessage;

public DES() {

try {

generateSymmetricKey();

inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");

byte[] ibyte = inputMessage.getBytes();

byte[] ebyte=encrypt(raw, ibyte);

String encryptedData = new String(ebyte);

System.out.println("Encrypted message "+encryptedData);

JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);

byte[] dbyte= decrypt(raw,ebyte);

String decryptedMessage = new String(dbyte);

System.out.println("Decrypted message "+decryptedMessage);

JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);

}

catch(Exception e) {

System.out.println(e);

}

}

void generateSymmetricKey() {

try {

Random r = new Random();

int num = r.nextInt(10000);

String knum = String.valueOf(num);

byte[] knumb = knum.getBytes();

skey=getRawKey(knumb);

skeyString = new String(skey);

System.out.println("DES Symmetric key = "+skeyString);

}

catch(Exception e) {

System.out.println(e);

}

}

private static byte[] getRawKey(byte[] seed) throws Exception {

KeyGenerator kgen = KeyGenerator.getInstance("DES");

SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");

sr.setSeed(seed);

kgen.init(56, sr);

SecretKey skey = kgen.generateKey();

raw = skey.getEncoded();

return raw;

}

private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");

Cipher cipher = Cipher.getInstance("DES");

cipher.init(Cipher.ENCRYPT\_MODE, skeySpec);

byte[] encrypted = cipher.doFinal(clear);

return encrypted;

}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");

Cipher cipher = Cipher.getInstance("DES");

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] decrypted = cipher.doFinal(encrypted);

return decrypted;

}

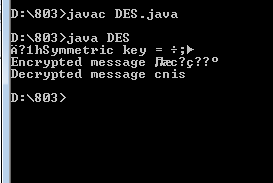
public static void main(String args[]) {

DES des = new DES();

}

}

OUTPUT:



**4.Implement Diffie Hellman Algorithm and generate Secret Key.**

**Aim**:To Understand the Diffie Hellman algorithm.

**Program**:

import java.math.BigInteger;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.SecureRandom;

import javax.crypto.spec.DHParameterSpec;

import javax.crypto.spec.DHPublicKeySpec;

public class DiffeHellman

{

public final static int pValue = 47;

public final static int gValue = 71;

public final static int XaValue = 9;

public final static int XbValue = 14;

public static void main(String[] args) throws Exception

{

BigInteger p = new BigInteger(Integer.toString(pValue));

BigInteger g = new BigInteger(Integer.toString(gValue));

BigInteger Xa = new BigInteger(Integer.toString(XaValue));

BigInteger Xb = new BigInteger(Integer.toString(XbValue));

createKey();

int bitLength = 512;

SecureRandom rnd = new SecureRandom();

p = BigInteger.probablePrime(bitLength, rnd);

g = BigInteger.probablePrime(bitLength, rnd);

createSpecificKey(p, g);

}

public static void createKey() throws Exception

{

KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");

kpg.initialize(512);

KeyPair kp = kpg.generateKeyPair();

KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");

DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("Public key is: " +kspec);

}

public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception

{

KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");

DHParameterSpec param = new DHParameterSpec(p, g);

kpg.initialize(param);

KeyPair kp = kpg.generateKeyPair();

KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");

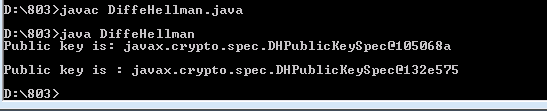
DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("\nPublic key is : " +kspec);

}

}

**OUTPUT**:



**5.Implement RSA algorithm**

**a)Generate Public key and Private key pair**

**b)Generate Ciphertext for the Plaintext**

**c)Obtain the Plaintext from the Ciphertext**

**Aim**: To understand the RSA Algorithm.

**Program:**

import java.util.\*;

import java.math.\*;

class RSA1

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int p,q,n,z,d=0,e,i;

System.out.println("Enter the number to be encrypted and decrypted");

int msg=sc.nextInt();

double c;

BigInteger msgback;

System.out.println("Enter 1st prime number p");

p=sc.nextInt();

System.out.println("Enter 2nd prime number q");

q=sc.nextInt();

n=p\*q;

z=(p-1)\*(q-1);

System.out.println("the value of z = "+z);

for(e=2;e<z;e++)

{

if(gcd(e,z)==1) // e is for public key exponent

{

break;

}

}

System.out.println("the value of e = "+e);

for(i=0;i<=9;i++)

{

int x=1+(i\*z);

if(x%e==0) //d is for private key exponent

{

d=x/e;

break;

}

}

System.out.println("the value of d = "+d);

c=(Math.pow(msg,e))%n;

System.out.println("Encrypted message is : -");

System.out.println(c);

//converting int value of n to BigInteger

BigInteger N = BigInteger.valueOf(n);

//converting float value of c to BigInteger

BigInteger C = BigDecimal.valueOf(c).toBigInteger();

msgback = (C.pow(d)).mod(N);

System.out.println("Derypted message is : -");

System.out.println(msgback);

}

static int gcd(int e, int z)

{

if(e==0)

return z;

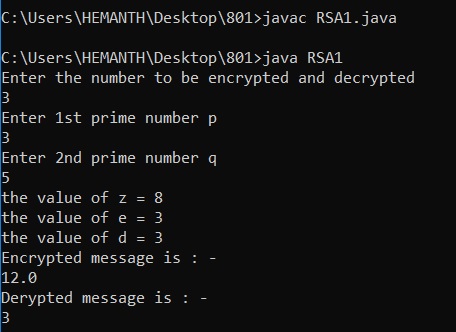
else

return gcd(z%e,e);

}

}

**OUTPUT**:



1. **Implement Hash Algorithm**

**Aim**:To understand the Hashing Technique.

**Program**:

class MD5

{ import java.security.\*;

public

public static void main(String[] a)

{

try

{

MessageDigest md = MessageDigest.getInstance("MD5");

System.out.println("Message digest object info: ");

System.out.println(" Algorithm = " +md.getAlgorithm());

System.out.println(" Provider = " +md.getProvider());

System.out.println(" ToString = " +md.toString());

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("MD5(\"" +input+"\") = " +bytesToHex(output));

System.out.println("");

}

catch (Exception e) { System.out.println("Exception: " +e); }

}

public static String bytesToHex(byte[] b)

{

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBuffer buf = new StringBuffer();

for (int j=0; j<b.length; j++)

{

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

buf.append(hexDigit[b[j] & 0x0f]);

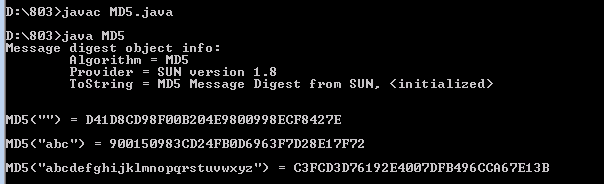
}

return buf.toString();

}

}

OUTPUT:



1. **Generate Digital Signature**

**Aim**:To Generate the Digital signature .

**Program**:

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.Signature;

import sun.misc.BASE64Encoder;

public class DigSign {

public static void main(String[] args) throws Exception {

// TODO code application logic here

KeyPairGeneratorkpg = KeyPairGenerator.getInstance("RSA");

kpg.initialize(1024);

KeyPairkeyPair = kpg.genKeyPair();

byte[] data = "Sample Text".getBytes("UTF8");

Signature sig = Signature.getInstance("MD5WithRSA");

sig.initSign(keyPair.getPrivate());

sig.update(data);

byte[] signatureBytes = sig.sign();

System.out.println("Signature: \n" + new BASE64Encoder().encode(signatureBytes));

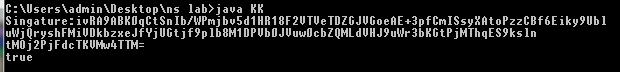
sig.initVerify(keyPair.getPublic());

sig.update(data);

System.out.println(sig.verify(signatureBytes));

}

}

OUTPUT:

1. **Implement Digital Envelope**

**Aim**:To Implement Digital Envelope.

**Program:**

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

public class DSAKeyGen

{

public static void main(String[] args) throws Exception

{

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");

keyGen.initialize(1024);

KeyPair keypair = keyGen.genKeyPair();

PrivateKey privateKey = keypair.getPrivate();

System.out.println(privateKey);

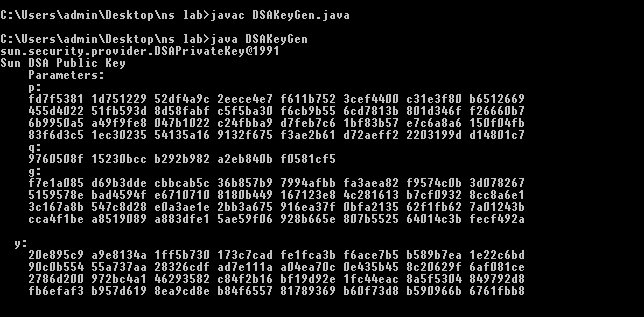
PublicKey publicKey = keypair.getPublic();

System.out.println(publicKey);

}

}

**OUTPUT**:



1. **Installation of NS3**

**Aim**:To Learn the Installation of NS3.

**Steps:**

# [**ns3 installation in Ubuntu 16.04**](https://www.nsnam.com/2018/03/ns3-installation-in-ubuntu-1604.html)

**OS Used**: Ubuntu 16.04.4  
**ns3 version**: ns3 version 3.27  
  
The same procedure will be applied for OS like Debian, Linux Mint.

## Fresh installation of Ubuntu 16.04

Let you try the fresh installation of ubuntu in your hard disk along with windows.

## Installation of ns3 dependencies

ns3 needs so many dependencies, developmental libraries, drivers, etc. so install all those

$] sudo apt update

$] sudo apt upgrade

$] sudo apt-get install build-essential autoconf automake libxmu-dev python-pygoocanvas python-pygraphviz cvs mercurial bzr git cmake p7zip-full python-matplotlib python-tk python-dev python-kiwi python-gnome2 python-gnome2-desktop-dev python-rsvg qt4-dev-tools qt4-qmake qt4-qmake qt4-default gnuplot-x11 wireshark

The above command make take some time to download, compile and install it, Be Patient.

## **Installing ns3**

Download the ns3 package from  [https://www.nsnam.org](https://www.nsnam.org/)

I have used ns version 3.27 and the file looks like ns-allinone-3.27.tar.bz2.

You can unzip it in two ways. Right click over the above file and extract it to the folder (/home/pradeepkumar/). Most preferred place to install is to put it in the home folder.

**Or**

go to the location of the download folder and copy the file to the home folder and open the terminal and give the command

$]  tar jxvf ns-allione-3.27.tar.bz2

$] cd ns-allinone-3.27/

$] ./build.py --enable-examples --enable-tests

This will take sometime for getting compiled and build. Once the installation is successful, you will get a screen like given below.

|  |
| --- |
| [ns3 installation](https://4.bp.blogspot.com/-bn2memgI8dI/WrZG1ZQ5ivI/AAAAAAABcrg/NHswT7qjtR8FVoNsv0meda1jaoh-oIGEgCLcBGAs/s1600/Screenshot+from+2018-03-24+17-43-46.png) |
| ns3 installation |

This indicates that ns3 is built successfully.

To check any application is running. do the following steps

$] cd ns-3.27/

$] ./waf --run hello-simulator

This will print the hello Simulator which indicates that ns3 is installed successfully.

**10. Demonstration of NS3**.

**ns-3** is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free, open-source software, licensed under the GNU GPLv2 license, and maintained by a worldwide community.

## **Supported platforms**

ns-3 is primarily developed on GNU/Linux and macOS platforms, and the minimal requirements to run basic simulations are a C++ compiler; either [g++](http://gcc.gnu.org/) or [clang++](http://clang.llvm.org/) compiler, and [Python (version 3)](http://www-python.org/) interpreter. The below instructions are per-platform instructions for supplemental packages that enable optional features of ns-3 or companion tools.

### Operating system and compiler support

**ns-3 is supported and currently tested on the following primary platforms**:

1. Linux (x86 and x86\_64): gcc/g++ versions 4.9 and above
   1. Note: If you are using RHEL or Centos, you will likely need to install a more up-to-date compiler than the default; search for how to enable 'software collections' or 'devtoolset' on these distributions. Other Linux distributions typically have a suitable default compiler (at least version 4.9).
2. MacOS Apple LLVM: version 8.0.0 and above (version 7.0.0 may work)
3. FreeBSD and Linux (x86\_64): clang/LLVM version 3.9 and above (older versions down to 3.3 may work)

**The minimum Python version supported is currently version 3.5 or greater (major version 3).**

By supported, we mean that the project tries to support most or all of the build options on these platforms unless there is a good reason to exclude the option; and at least the debug build will compile. If you intend to do serious work using ns-3, and are forced by circumstances to use a Windows platform, consider virtualization of a popular Linux platform or using [Windows Subsystem for Linux](https://msdn.microsoft.com/en-us/commandline/wsl/install_guide).

**The following platforms are lightly supported:**

* Windows Visual Studio 2012 (presently being [upgraded](https://www.nsnam.org/bugzilla/show_bug.cgi?id=2726))

Some aspects of ns-3 depend on Unix (or specifically Linux) support, such as the emulation or TapBridge features, and those components are not enabled on the Windows or MacOS versions cited above.

Additional maintainers are invited to make more platforms, compilers and environments supported.

### Integrated development environment support

#### **Eclipse**

The [Eclipse IDE](http://www.eclipse.org/) is not an officially supported platform, but some developers use it and have compiled a [HOWTO](https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3).

#### **NetBeans**

[NetBeans](https://netbeans.org/) is not officially supported either, but there is a [HOWTO](https://www.nsnam.org/wiki/HOWTO_configure_NetBeans_with_ns-3) as well.

#### **QtCreator**

Same rule applies to [Qt Creator](http://qt-project.org/wiki/category:tools::qtcreator); it's not officially supported, but there are developers that use it and [HOWTO](https://www.nsnam.org/wiki/HOWTO_configure_QtCreator_with_ns-3) is available.

### Support for optional features

There are a few options that are not enabled by default and are not available on all platforms. At the end of the configuration process (explained below), the status of these options are shown as detected by a waf script:

---- Summary of optional NS-3 features:

Python Bindings  : not enabled (Python library or headers missing)

BRITE Integration  : not enabled (BRITE not enabled (see option --with-brite))

NS-3 Click Integration  : not enabled (nsclick not enabled (see option --with-nsclick))

GtkConfigStore  : not enabled (library 'gtk+-2.0 >= 2.12' not found)

XmlIo  : not enabled (library 'libxml-2.0 >= 2.7' not found)

Threading Primitives  : enabled

Real Time Simulator  : enabled

Emulated Net Device  : not enabled (<netpacket/packet.h> include not detected)

Network Simulation Cradle  : not enabled (architecture None not supported)

MPI Support  : not enabled (option --enable-mpi not selected)

NS-3 OpenFlow Integration  : not enabled (OpenFlow not enabled (see option --with-openflow))

SQlite stats data output  : not enabled (library 'sqlite3' not found)

Tap Bridge  : not enabled (<linux/if\_tun.h> include not detected)

PyViz visualizer  : not enabled (Python Bindings are needed but not enabled)

Use sudo to set suid bit  : not enabled (option --enable-sudo not selected)

Build tests  : not enabled (defaults to disabled)

Build examples  : not enabled (defaults to disabled)

GNU Scientific Library (GSL)  : not enabled (GSL not found)

Generally if the platform is missing some **requirement** for an option it is marked as "not enabled." Note that "disabled by user request" will be shown when the user explicitly disables a feature (such as "--disable-python"); and if a feature defaults to disabled this will also be noted (e.g., option --enable-sudo not selected).

The table below is meant to help sort out the different features and on which platforms they are supported. This table reflects the status as of ns-3.15 and may have changed since then:

|  |  |  |  |
| --- | --- | --- | --- |
| Option status | | | |
| Option | Linux | FreeBSD | Mac OS X |
| Optimized build | Y | Y | Y |
| Python bindings | Y | Y | Y |
| Threading | Y | Y | Y |
| Real-time simulator | Y | Y | N |
| Emulated Net Device | Y | N | N |
| Tap Bridge | Y | N | N |
| Network simulation cradle | Y1 | ? | N |
| Static builds | Y | Y | Y |

*Key:* Y = supported; N = not supported; ? = unknown; dev = support in ns-3-dev (next release)

*Notes:*

1. NSC works best with gcc-3.4 or gcc-4.2 or greater series. Try to avoid using gcc-4.0 and gcc-4.1 series; some build problems have been found with these versions of compilers.

## **Prerequisites**

The core of ns-3 requires a gcc/g++ installation of 4.9 or greater (Linux), or a recent version of clang compiler (OS X, Linux, or BSD), and Python 3.5 or greater. As mentioned above, different options require additional support. This is a list of packages (for Debian/Ubuntu systems) that are needed to support different ns-3 options. Note that other distributions (e.g., Fedora, FreeBSD) may have different package names or capitalization (e.g. ImageMagik). Installation should be similar for Red Hat/Fedora based systems, with "yum" replacing "apt-get", but some differences exist, so below is a guide for both Ubuntu (should generally apply to Debian) and Fedora/RedHat-based systems:

### Linux

#### CentOS

**Note**: CentOS version 6 series (currently 6.8) requires an upgrade of both gcc and Python to meet current ns-3 requirements. See these instructions if you need to upgrade: <https://www.nsnam.org/bugzilla/show_bug.cgi?id=2667#c1>

The below instructions are based on a CentOS 6.6 install in November 2014; other RedHat/Fedora-based installs are likely similar.

* minimal requirements for C++ (release): This is the minimal set of packages needed to run ns-3 C++ programs from a released tarball.

yum install gcc-c++ python3

* minimal requirements for Python (release): Python development headers are necessary to enable the Python bindings (for writing ns-3 programs from Python):

yum install python3-devel

The following additional packages add functionality to the build or documentation.

* The netanim animator requires Qt5 development packages:

yum install qt5-devel

* Mercurial is needed to work with ns-3 development repositories.

yum install mercurial

* Doxygen and related inline documentation:

yum install doxygen graphviz ImageMagick

* The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia:

yum install python3-sphinx dia texlive texlive-latex

* MPI-based parallel, distributed simulation support requires openmpi:

yum install openmpi openmpi-devel

* To read pcap packet traces generated by ns-3

yum install tcpdump wireshark

* Database support for statistics

yum install sqlite sqlite-devel

* Xml-based version of the config store (requires libxml2 >= version 2.7)

yum install libxml2 libxml2-devel

* Support for utils/check-style.py style check program

yum install uncrustify

* Support for openflowswitch requires libxml2, if not installed above, and Boost libraries

yum install libxml2 libxml2-devel boost-devel

* Support for ns-3-pyviz visualizer

This is a bit more involved due to lack of package support in the standard repositories.

yum install graphviz graphviz-devel python3-setuptools-devel ipython3

sudo easy\_install pygraphviz

Some additional packages are needed (goocanvas and pygoocanvas). It is suggested to enable the RPMForge repo as described here: <http://wiki.centos.org/AdditionalResources/Repositories/RPMForge>. Then, try this:

yum install goocanvas pygtk2-devel

Then obtain the RPM for pygoocanvas and pygoocanvas-devel from here: <http://li.nux.ro/download/nux/dextop/el6/x86_64/>

rpm -ivh pygoocanvas-0.14.1-3.el6.nux.x86\_64.rpm

rpm -ivh pygoocanvas-devel-0.14.1-3.el6.nux.x86\_64.rpm

Or, if you prefer, build pygoocanvas from source code found here: <https://wiki.gnome.org/Projects/PyGoocanvas>

Note, if you perform this install successfully on a CentOS server that does not have a desktop installed (i.e. a CentOS 'minimal install'), you will still not be able to see pyviz enabled; you will see the configuration report:

PyViz visualizer  : not enabled (Missing python modules: gtk)

because the Python gtk module opens the display upon import.

* Git is needed to fetch click modular routing and pygccxml

yum install git

* An optional but recommended package (for improving some wireless model fidelity) is GNU scientific library:

yum install gsl gsl-devel

* A GTK-based configuration system

yum install gtk2 gtk2-devel

* Debugging:

yum install gdb valgrind

[Jump to installation](https://www.nsnam.org/wiki/Installation#Installation)

#### Ubuntu/Debian/Mint

The following list of packages should be accurate through the Ubuntu 21.04 release; other releases or other Debian-based systems may slightly vary. Ubuntu 16.04 LTS release is probably the oldest release that is known to work with recent ns-3 releases.

Note: As of ns-3.30 release (August 2019), ns-3 uses Python 3 by default, but earlier releases depend on Python 2 packages, and at least a Python 2 interpreter is recommended. If working with an earlier release, one may in general substitute 'python' for 'python3' in the below (e.g. install 'python-dev' instead of 'python3-dev').

* minimal requirements for C++ users (release): This is the minimal set of packages needed to run ns-3 from a released tarball.

apt install g++ python3

* minimal requirements for Python API users (release 3.30 and newer, and ns-3-dev): This is the minimal set of packages needed to work with Python bindings from a released tarball.

apt install g++ python3 python3-dev pkg-config sqlite3

* minimal requirements for Python (development): For use of ns-3-allinone repository (cloned from Git), additional packages are needed to fetch and successfully install pybindgen and netanim.

apt install python3-setuptools git

* Netanim animator: qt5 development tools are needed for Netanim animator; qt4 will also work but we have migrated to qt5.

apt install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools

**Note**: For Ubuntu 20.10 and earlier, the single 'qt5-default' package suffices

apt install qt5-default

* Support for ns-3-pyviz visualizer
  + For Ubuntu 18.04 and later, python-pygoocanvas is no longer provided. The ns-3.29 release and later upgrades the support to GTK+ version 3, and requires these packages:

apt install gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3

* + For ns-3.28 and earlier releases, PyViz is based on GTK+ 2, GooCanvas, and GraphViz:

apt install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython

* Support for MPI-based distributed emulation

apt install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev

* Support for bake build tool:

apt install autoconf cvs bzr unrar

* Debugging:

apt install gdb valgrind

* Support for utils/check-style.py code style check program

apt install uncrustify

* Doxygen and related inline documentation:

apt install doxygen graphviz imagemagick

apt install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils dvipng latexmk

* + If you get an error such as 'convert ... not authorized source-temp/figures/lena-dual-stripe.eps', see this post about editing ImageMagick's security policy configuration: <https://cromwell-intl.com/open-source/pdf-not-authorized.html>. In brief, you will want to make this kind of change to ImageMagick security policy:

--- ImageMagick-6/policy.xml.bak 2020-04-28 21:10:08.564613444 -0700

+++ ImageMagick-6/policy.xml 2020-04-28 21:10:29.413438798 -0700

@@ -87,10 +87,10 @@

<policy domain="path" rights="none" pattern="@\*"/>

- <policy domain="coder" rights="none" pattern="PS" />

+ <policy domain="coder" rights="read|write" pattern="PS" />

<policy domain="coder" rights="none" pattern="PS2" />

<policy domain="coder" rights="none" pattern="PS3" />

<policy domain="coder" rights="none" pattern="EPS" />

- <policy domain="coder" rights="none" pattern="PDF" />

+ <policy domain="coder" rights="read|write" pattern="PDF" />

<policy domain="coder" rights="none" pattern="XPS" />

</policymap>

* The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia (also needs the texlive packages above):

apt install python3-sphinx dia

Note: Sphinx version >= 1.12 required for ns-3.15. To check your version, type "sphinx-build". To fetch this package alone, outside of the Ubuntu package system, try "sudo easy\_install -U Sphinx".

* GNU Scientific Library (GSL) support for more accurate 802.11b WiFi error models (not needed for OFDM):

apt install gsl-bin libgsl-dev libgslcblas0

If the above doesn't work (doesn't detect GSL on the system), consult: <https://coral.ise.lehigh.edu/jild13/2016/07/11/hello/>. But don't worry if you are not using 802.11b models.

* To read pcap packet traces

apt install tcpdump

* Database support for statistics framework

apt install sqlite sqlite3 libsqlite3-dev

* Xml-based version of the config store (requires libxml2 >= version 2.7)

apt install libxml2 libxml2-dev

* Support for generating modified python bindings

apt install cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake python3-pip

python3 -m pip install --user cxxfilt

and you will want to install castxml and pygccxml as per the instructions for python bindings (or through the *bake* build tool as described in the tutorial). The 'castxml' and 'pygccxml' packages provided by Ubuntu 18.04 and earlier are not recommended; a source build (coordinated via bake) is recommended. If you plan to work with bindings or rescan them for any ns-3 C++ changes you might make, please read the [chapter in the manual](https://www.nsnam.org/docs/manual/html/python.html) on this topic.

Note: Ubuntu versions (through 19.04) and systems based on it (e.g. Linux Mint 18) default to an old version of clang and llvm (3.8), when simply 'libclang-dev' and 'llvm-dev' are specified. The packaging on these 3.8 versions is broken. Users of Ubuntu will want to explicitly install a newer version by specifying 'libclang-6.0-dev' and 'llvm-6.0-dev'. Other versions newer than 6.0 may work (not tested).

* A GTK-based configuration system

apt install libgtk-3-dev

* To experiment with virtual machines and ns-3

apt install vtun lxc uml-utilities

* Support for openflow module (requires libxml2-dev if not installed above) and Boost development libraries

apt install libxml2 libxml2-dev libboost-all-dev

**11.Executing simple projects in NS3.**

**Aim**:To Execute simple projects in NS3 to get Practical knowledge.

#### **A PREVIEW OF NS3**

   NS3 puts up with the help of C++ as well as Python. It aids in cracking the snags extant in NS2. It is a free tool too. [NS3](https://networksimulationtools.com/ns3-projects/) fits in with real networks, which then act as an Emulator.  So as to drive real-time, it uses virtualization system. It works on Ubuntu, FreeBSD, Mac OS X, and Windows. It supports call driven events and connections.

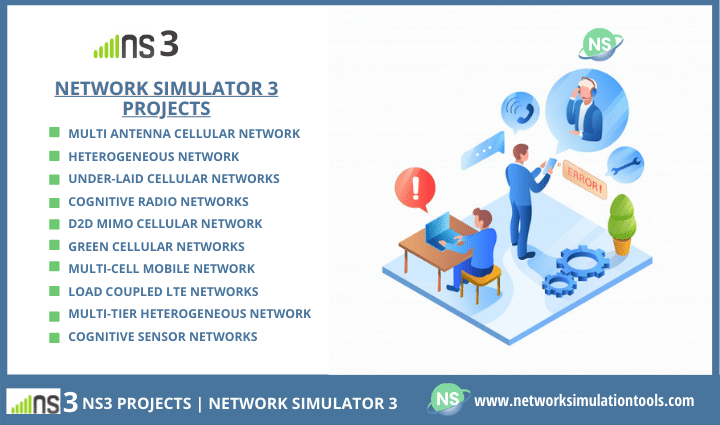
#### **NS3 SIMULATION FLOW**

* **Define Network Topology**
  + performance Analysis
  + figuration
  + d also doing a project by all the latest list of updates of NS3. Node containers (defining nodes)
  + Fix FiFixFFix Channel, Mobility, IP, and also Application
* **Model Development**
  + Develop by a Defined Channel and Mobility Models
  + Simulate a Model according to Events
* **Node and also Link Configuration**
  + Define Node and Link Attributes (Data rate, Latency and also Packet Loss)
  + Manage Command Arguments
* **Simulation Execution**
  + Building a File and also Analyze Outputs in a Command Prompt
* **Performance Analysis**
  + Modeling Network by using NetAnim
  + Analyze the Performance under GNU plot

   For your project in NS3, we make a plan by anyone of latest version as follows.

#### **NS3 LATEST RELEASES**

* NS – 3.30
* NS – 3.29
* NS – 3.28
* NS – 3.27
* NS – 3.26



With the aim to begin [PhD projects in ns3](https://phdservices.org/phd-projects-in-ns3/), one must craft the above case in point. At that time, we have to set all of the parameters. It is simple but perfect to work a project by using NS3. Given that we write the code in accordance with your project ideas. Through this page, we address the following questions.

* Which are the most impacted network areas as per the current situation?
* How can NS3 easily adopt new concepts as much as easily?
* What kind of modules available in NS3?
* What are the issues addressing by using NS3?

   Let’s start with the one by one. In the first question, we talk about the impact areas of NS3. Today, NS3 is a widely accepted tool in all wireless networks. NS3 has distinct features over the ad hoc, sensor, and other wireless networks. To tell the truth, it gives the weights on the advanced to build a project. It ropes each nook and corner of all network fields. Some of them are,

* Multi-Antenna Cellular Network
* Heterogeneous Network
* Under-laid Cellular Networks
* Cognitive Radio Networks
* D2D MIMO Cellular Network
* Green Cellular Networks
* Multi-Cell Mobile Network
* Load Coupled LTE Networks
* Multi-Tier Heterogeneous Network
* Multimedia and also Cognitive Sensor Networks

   The new sets of reasons are available to choose why to use NS3. Most likely, we can see the closeness of each of the reasons. Due to the following main reasons, any complex or real-time project make from it.

#### **THE MAIN REASONS – WHY WE GO FOR NS3?**

* **Scalability features**
  + Any number of nodes can care from it
  + No memory will allocate for virtual zero bytes
* **Cross-Layer Features**
  + Packet Tags and also Tracing
* **Real World Integration Features**
  + Packets save into PCAP files in real format
  + Most of the tools can read this file (e.g., Wireshark)
  + Supports of a real-time scheduler
* **Easy to interface with any other tools**
  + SNS3, Matlab, [Mininet](https://networksimulationtools.com/mininet-projects/" \t "_blank), and also so on

   Besides the above list, it has wide lists of modules for a specific purpose. All in all, current research efforts are ongoing in the modules, too, since it eases the project to integrate any number of wireless networks. For instance, WiFi, and LTE modules support to interact with real-time systems.

#### **CURRENT NS3 MODULES**

* IR-WPAN and WiFi
* LTE along with Network
* Energy and also Wi-Max
* TCP, Internet, and also UAN
* Traffic Control and Propagation
* Application Module UAN

   It’s recently intense in industry and academic domains by reason of its extensibility, high sensing and fault tolerance. Now, we talk about some of the issues addressed by using NS3.

#### **WHAT RESEARCH ISSUES CURRENTLY DEAL WITH NS3?**

* Traffic Rate Adaptation
* Spectrum Sensing
* Bandwidth Allocation
* Fair Resource Scheduling
* Security Assessment
* Networking Protocols Design
* Performance enrich of Physical Layer

   As we know, each new concept invents after a survey and also the use of years of skills in that field. That is to say, we not only offer you just a concept but stay until with the project completion. In this event, we first carry a novel idea under your selected domain. Significantly, we explain the ideas by a list below.

* [Hybrid BeamForming](https://networksimulationtools.com/hybrid-beamforming-projects/)
* Vertical and also Horizontal Handover
* Service Discovery
* V2X Communication
* [Intrusion Detection Systems](https://networksimulationtools.com/intrusion-detection-system-projects/)
* And also many more

   To conclude, we work on a best effort basis and also doing a project by all the latest lists of updates of NS3 for your [NS3 projects](https://networksimulationtools.com/ns3-projects/). When you call for your project, we talk to you explicitly with an open mind for discussing all possibilities.

Example for NS3 Project:

# MANET- Secure/Fast Data Transfer

A mobile adhoc network is a type of network in which various mobile hosts are tied together in a multihop wireless link. The network is very mobile that means the nodes in it can change their positions randomly. A routing protocol capable of adjusting to topology changes is needed for fast data transfers. So here our system proposes a secure yet quick protocol having a proactive as well as reactive in nature. Proactive protocols undergo routing table updates as soon as a topology change takes place. Routers having dynamic routing protocols need to have fast convergence to avoid incorrect forwarding decisions before full network convergence. In proactive networks, a newly added node may require time to converge in it. Immediate data transfer through that node is not transfer until the node has fully converged into the network. On such times we use reactive protocols. Reactive protocols are used for the times when proactive nodes are being converged into the network. Proactive protocol structure has a low packet size and possesses only limited fields. Our system proposes to make data transmission as early as possible by removing the wait time. Also our system integrates a security framework for detecting malicious nodes with the use of central and process algorithm.