1. **INTRODUCTION**
   1. **Introduction to Project:**

Wireless networks are totally different to wired networks, which can be physically secured. The information goes through the air, where anyone can transmit and anyone can receive. Signals are not guided by wires. WLANs are therefore essentially vulnerable to interception.

The most prominent feature about WLAN is the absence of wires and its mobility. However, as data travelled through air using radio frequency, it can easily be tapped by anyone including unauthenticated personnel using a sniffer.

Prevention during primary stages is always the best solution. It does not mean that some organizations do not need to worry about wireless security if their wireless LAN deployment was not too significant. At some point, most connect with the main organization’s backbone. It is possible for hackers to use the wireless LAN as a launch pad to the entire network. To achieve complete security in wireless network seems a near impossible task. Therefore adequate deterrence should be suggested instead while designing a wireless network. In this case it means that we need to take a hard look at the access point. The access point would be the first to look into for setting up a good secured wireless network.

**Wired Equivalent Privacy (WEP):**

WEP (Wired Equivalent Privacy), a security protocol for wireless local area networks (WLANs) defined in the 802.11 standard. WEP is so called because it was designed to provide the same level of security as that of a wired LAN. But LANs are actually more secure than WLANs because LANs are restricted by physical access that secures them from unauthorized access. The Wired Equivalent Privacy (WEP) algorithm was designed to be used to protect wireless communication from unauthorized eavesdropping and restricting access to a wireless network.

WEP relies on a secret key that is shared between a mobile station (e.g. a laptop with a wireless Ethernet card) and an access point (i.e. a base station). The secret key is used to encrypt packets before they are transmitted, and an integrity check is used to ensure that packets are not modified during the transition. The secret key is used to encrypt and decrypt the data. To avoid encrypting two cipher texts with the same key, an Initialization Vector (IV) is used to augment the shared WEP key (secret key).WEP key (secret key) is available in two types, 64-bits and 128-bits.Select one of the WEP key as the Key to encrypt and send the data and the same key is received by the user to decrypt the data. WEP employs the key encryption algorithm, Ron's Code 4 Pseudo Random Number Generator (RC4 PRNG). The same key is used to encrypt and decrypt the data.

**RC4 Stream Cipher:**

WEP uses RC4 encryption algorithm known as “stream cipher” to protect the confidentiality of its data. RC4 algorithm is used for encrypting and decrypting the data. Stream cipher operates by expanding a short key into an infinite pseudo-random key stream. Sender XORs the key stream with plaintext to produce cipher text. Receiver has the copy of the same key, and uses it to generate an identical key stream. XORing the key stream with the cipher text yields the original message.

**2. SYSTEM ANALYSIS**

**2.1 Existing System:**

To achieve complete security in wireless network is impossible task. Therefore necessary measures need to be taken while designing a wireless network. By using IP address access is provided to the clients.

**Demerits of Existing System:**

* IP addresses can be duplicated, therefore Intruders can connect to the network easily.
* Data can be modified by intruders while transferring files from one client to another.

**2.2 Proposed System:**

Every device has unique MAC address, so that MAC address and IP address combination is used for authentication. Security can be provided while transferring files from one device to another using WEP algorithm. WEP key is provided to encrypt the data which is to be transferred from one client to another client. RC4 algorithm is used for encrypting and decrypting the data.

**Merits of Proposed System:**

* Mac addresses cannot be duplicated.
* The Wired Equivalent Privacy (WEP) algorithm is used to protect wireless communication from unauthorized eavesdropping and restricting access to a wireless network.

**2.3 Algorithm Implementation:**

The Wired Equivalent Privacy (WEP) algorithm is used to protect wireless communication from eavesdropping, because wireless transmissions are easier to intercept than transmissions over wired networks, and wireless is a shared medium, everything that is transmitted or received over a wireless network can be intercepted. WEP relies on a secret key that is shared between a mobile station (e.g. a laptop with a wireless Ethernet card) and an access point (i.e. a base station). The secret key is used to encrypt packets before they are transmitted, and an integrity check is used to ensure that packets are not modified during the transition. WEP employs the key encryption algorithm, Ron's Code 4 Pseudo Random Number Generator (RC4 PRNG). The same key is used to encrypt and decrypt the data.

Plain text

CRC

Message

Key Sequence = RC4 (IV, k)

IV

Cipher text

Transmitted Data

Figure 2.1: Wired Equivalent Privacy (WEP)

WEP key(secret key) are available in two types, 64-bits and 128-bits. Most access points and clients have the ability to hold up to 4 WEP keys simultaneously. You need to specify one of the 4 keys as default Key for data encryption. To set up the Access Point, you will need to set the one of the following parameters:

64-bit WEP key (secret key) with 5 characters

64-bit WEP key (secret key) with 10 hexadecimal digits

128-bit WEP key (secret key) with 13 characters

128-bit WEP key (secret key) with 26 hexadecimal digits

Select one of the WEP key as default Key to encrypt wireless data transmission. The receiver will use the corresponding key to decrypt the data.

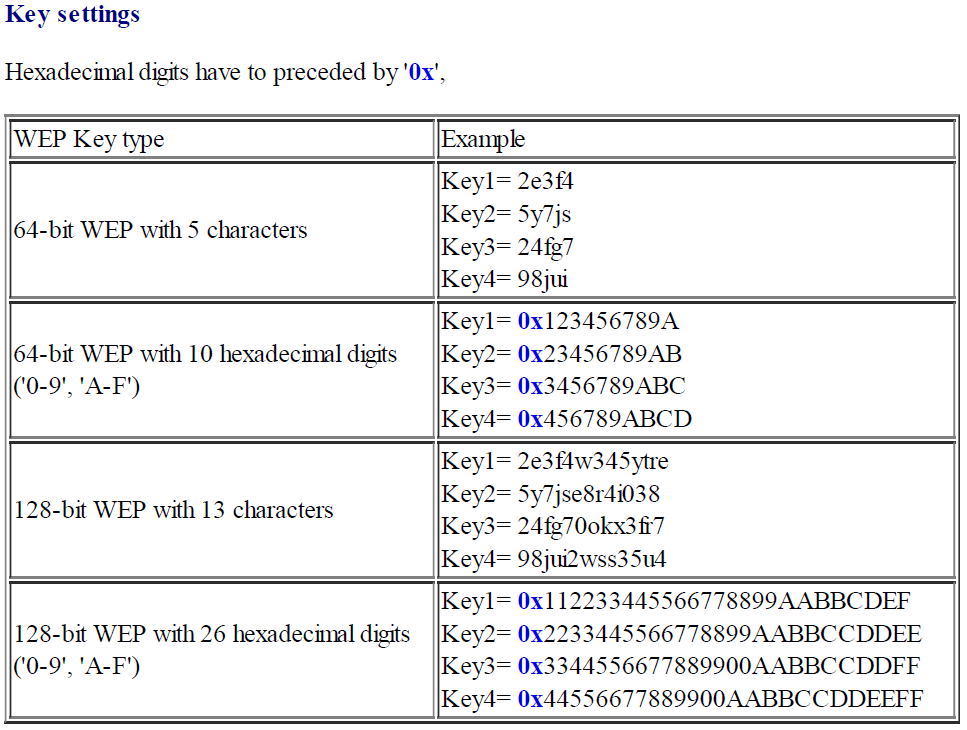


Figure 2.2: WEP Key Settings

**RC4 Algorithm:**

RC4 is a stream cipher, symmetric key algorithm. The same algorithm is used for both

encryption and decryption as the data stream is simply XORed with the generated key sequence. The key stream is completely independent of the plaintext used. It uses a variable length key from 1 to 256 bit to initialize a 256-bit state table. The state table is used for subsequent generation of pseudo-random bits and then to generate a pseudo-random stream which is XORed with the plaintext to give the cipher text. The algorithm can be broken into two stages: initialization, and operation. In the initialization stage the 256-bit state table, S is populated, using the key, K as a seed. Once the state table is setup, it continues to be modified in a regular pattern as data is encrypted. The initialization process can be summarized by the pseudo-code

j = 0;

for i = 0 to 255:

S[i] = i; //Initializes the 256 bit State Table

for i = 0 to 255:

j = (j + S[i] + K[i]) mod 256;

swap S[i] and S[j]; //Swapping

It is important to notice here the swapping of the locations of the numbers 0 to 255 (each of which occurs only once) in the state table. The values of the state table are provided.

Once the initialization process is completed, the operation process may be summarized as shown by the pseudo code below

i = j = 0;

for (k = 0 to N-1) { //input message consisting of N bits.

i = (i + 1) mod 256;

j = (j + S[i]) mod 256;

swap S[i] and S[j];

pr = S[ (S[i] + S[j]) mod 256] //stream of pseudo-random values

output M[k] XOR pr //input stream is XOR ed with these values

}

Where M[0..N-1] is the input message consisting of N bits. This algorithm produces a stream of pseudo-random values. The input stream is XORed with these values, bit by bit. The encryption and decryption process is the same as the data stream is simply XORed with the generated key sequence. If it is fed in an encrypted message, it will produce the decrypted message output, and if it is fed in plaintext message, it will produce the encrypted version.

Initial with Numbers from 0 to 255

Fill With Chosen key

Sbox 1

Sbox 2

Systematic Randomization

Systematic Randomization

Final Key Stream

Cipher Plaint Text

Plain Cipher Text

XOR

Figure 2.3:RC4 Encryption Algorithm

**RC4 Steps:**

The steps for RC4 encryption algorithm is as follows:

1- Get the data to be encrypted and the selected key.

2- Create two string arrays.

3- Initiate one array with numbers from 0 to 255.

4- Fill the other array with the selected key.

5- Randomize the first array depending on the array of the key.

6- Randomize the first array within itself to generate the final key stream.

7- XOR the final key stream with the data to be encrypted to give cipher text.

**Simplified RC4 Example :**

Let us consider the stream cipher RC4, but instead of the full 256 bytes, we will use 8 x 3-bits. That is, the state vector S is 8 x 3-bits. We will operate on 3-bits of plain text at a time since S can take the values 0 to 7, which can be represented as 3 bits.

Assume we use a 4 x 3-bit key of K = [1 2 3 6]. And a plaintext P = [1 2 2 2]

The first step is to generate the stream.

Initialise the state vector S and temporary vector T. S is initialised so the S[i] = i, and T is initialised so it is the key K (repeated as necessary).

S = [0 1 2 3 4 5 6 7] //State Vector

T = [1 2 3 6 1 2 3 6] //Temporary Vector

Now perform the initial permutation on S.

j = 0;

for i = 0 to 7 do

j = (j + S[i] + T[i]) mod 8

Swap(S[i],S[j]); //Swapping of State and Temporary Vector

end

For i = 0:

j = (0 + 0 + 1) mod 8

= 1

Swap(S[0],S[1]);

S = [1 0 2 3 4 5 6 7] //Modifies the initial State Vector

For i = 1:

j = 3

Swap(S[1],S[3])

S = [1 3 2 0 4 5 6 7];

For i = 2:

j = 0

Swap(S[2],S[0]);

S = [2 3 1 0 4 5 6 7];

For i = 3:

j = 6;

Swap(S[3],S[6])

S = [2 3 1 6 4 5 0 7];

For i = 4:

j = 3

Swap(S[4],S[3])

S = [2 3 1 4 6 5 0 7];

For i = 5:

j = 2

Swap(S[5],S[2]);

S = [2 3 5 4 6 1 0 7];

For i = 6:

j = 5;

Swap(S[6],S[4])

S = [2 3 5 4 0 1 6 7];

For i = 7:

j = 2;

Swap(S[7],S[2])

S = [2 3 7 4 0 1 6 5];

Hence, our initial permutation of S = [2 3 7 4 0 1 6 5];

Now we generate 3-bits at a time, k, that we XOR with each 3-bits of plaintext to produce the cipher text. The 3-bits k is generated by:

i, j = 0;

while (true) {

i = (i + 1) mod 8;

j = (j + S[i]) mod 8;

Swap (S[i], S[j]);

t = (S[i] + S[j]) mod 8;

k = S[t]; }

The first iteration:

S = [2 3 7 4 0 1 6 5]

i = (0 + 1) mod 8 = 1

j = (0 + S[1]) mod 8 = 3

Swap(S[1],S[3])

S = [2 4 7 3 0 1 6 5]

t = (S[1] + S[3]) mod 8 = 7

k = S[7] = 5

Remember, P = [1 2 2 2] //Plain Text

So our first 3-bits of cipher text is obtained by: k XOR P

5 XOR 1 = 101 XOR 001 = 100 = 4 //Key is XORed with Plain text

The second iteration:

S = [2 4 7 3 0 1 6 5]

i = (1 + 1 ) mod 8 = 2

j = (2 + S[2]) mod 8 = 1

Swap(S[2],S[1])

S = [2 7 4 3 0 1 6 5]

t = (S[2] + S[1]) mod 8 = 3

k = S[3] = 3

Second 3-bits of cipher text are:

3 XOR 2 = 011 XOR 010 = 001 = 1 //Key is XORed with Plain text

The third iteration:

S = [2 7 4 3 0 1 6 5]

i = (2 + 1 ) mod 8 = 3

j = (1 + S[3]) mod 8 = 4

Swap(S[3],S[4])

S = [2 7 4 0 3 1 6 5]

t = (S[3] + S[4]) mod 8 = 3

k = S[3] = 0

Third 3-bits of cipher text are:

0 XOR 2 = 000 XOR 010 = 010 = 2 //Key is XORed with Plain text

The final iteration:

S = [2 7 4 0 3 1 6 5]

i = (1 + 3 ) mod 8 = 4

j = (4 + S[4]) mod 8 = 7

Swap(S[4],S[7])

S = [2 7 4 0 5 1 6 3]

t = (S[4] + S[7]) mod 8 = 0

k = S[0] = 2

Last 3-bits of cipher text are:

2 XOR 2 = 010 XOR 010 = 000 = 0 //Key is XORed with Plain text

So to encrypt the plain text stream P = [1 2 2 2] with key K = [1 2 3 6] using our simplified RC4 stream cipher we get C = [4 1 2 0]. (or in binary: P = 001010010010, K = 001010011110 and C = 100001010000)

**3. SYSTEM REQUIREMENT SPECIFICATIONS**

**3.1 Functional Requirements:**

The functional requirement of the system defines a function of software system or its components. A function is described as a set of inputs, behaviour of a system and output.

**INPUT:**

* Enrolling nodes by giving MAC Address and IP Address
* Authenticate the Nodes which are Enrolled
* Providing Access to each node which are authenticated
* Start each node for transferring the file
* Select the Destination Nodes/Destination Node
* Browse the file to transfer
* Select the Type of Encryption

**PROCESSING:**

* Encrypt the uploaded file with the selected type of Encryption using RC4 Algorithm
* Send the file to destination node
* At the destination node decrypt the file using RC4 Algorithm
* Receive the file at Destination Node

**OUTPUT:**

* File received at the destination node

**3.2 Non-Functional Requirements:**

A non-functional requirement in software engineering systematic and pragmatic approach to ‘building quality into software’. System must exhibit software quality such as accuracy, performance, security and modifiability.

* + **Usability:** System provides a convenient type of GUI for handling all operations.
  + **Performance:** Response time of the system is high . And it can handle any kind of

errors that may be generated while the user accessing the system.

* + **Security:** The system should provide security for the authorized users and it

should not allow unauthorized users to access the resources.

**3.3 System Requirements:**

**3.3.1 Software Requirements:**

Operating System : Windows XP / 7

Technology : .NET

Back End Tool : SQL Server

**3.3.2 Hardware Requirements:**

Processor : Pentium – IV and higher versions

Memory : 2 GB RAM

Secondary storage : 80 GB Disk Space

**4. SYSTEM DESIGN**

## 4.1 Database Tables:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Key Type** |
| Uname | VARCHAR | 50 | NOT NULL |
| Pwd | VARCHAR | 50 | NOT NULL |

Table 4.1:Database table Admin

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Key Type** |
| Src | VARCHAR | MAX | NOT NULL |
| Cipher | VARCHAR | MAX | NOT NULL |
| Datakey | VARCHAR | MAX | NOT NULL |
| Dest | VARCHAR | MAX | NOT NULL |

Table 4.2:Database table Info

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Key Type** |
| MAC | VARCHAR | 50 | PRIME KEY |
| IP | VARCHAR | 50 | NOT NULL |
| Node | VARCHAR | 50 | NOT NULL |
| Auth | VARCHAR | 1 | NOT NULL |

Table 4.3:Database table Nodes

## 4.2 UML Diagrams:

The Unified Modeling Language (UML) is a modeling language for specifying, visualizing, constructing and documenting the artifact of system intensive process. UML allows us to express an analysis model using a modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. Some worth referring points about UML are

* The UML is not simply a notation for drawing diagrams, but a complete language for capturing knowledge about a subject and expressing knowledge regarding the subject for the purpose of communication.
* UML applies to modeling the systems. Modeling involves a focus on understanding a subject, capturing and being able to communicate this knowledge.
* It is based on object-oriented paradigm.

**Use Case Diagram:**

A use case diagram is “a diagram that shows the relationships among actors and use cases within a system.”

* + Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model.
* Communicate the scope of a development project.

Use Case diagrams commonly contain:

* Use Cases
* Actors
* Dependency, generalization, and association relationship.

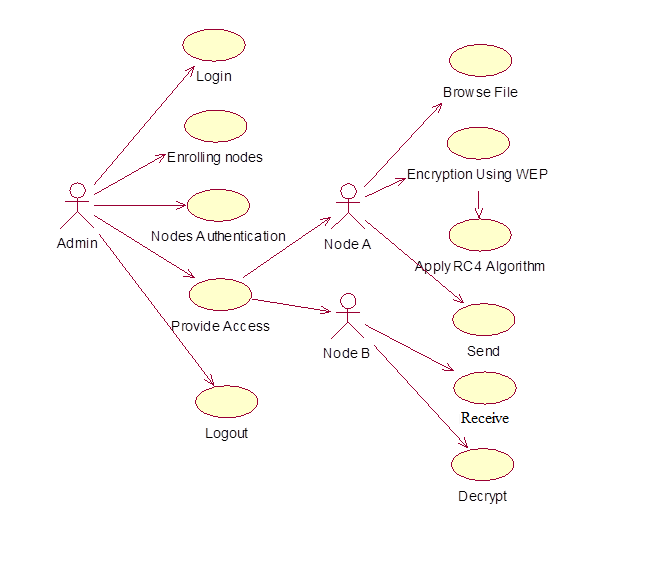


Figure 4.4: Use Case Diagram

**Class Diagram:**

Class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes.

Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling.

Class Diagrams commonly contain the following things:

* Classes
* Interfaces
* Collaborations
* Dependency, generalization and association relationships



Figure 4.5: Class Diagram

**Sequence Diagram:**

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

Sequence Diagrams commonly contain the following things:

* Object
* Link
* Messages
* Lifeline



Figure 4.6: Sequence Diagram

**Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

Activity Diagrams commonly contain the following things:

* Start State
* End State
* Transition
* Decision Box

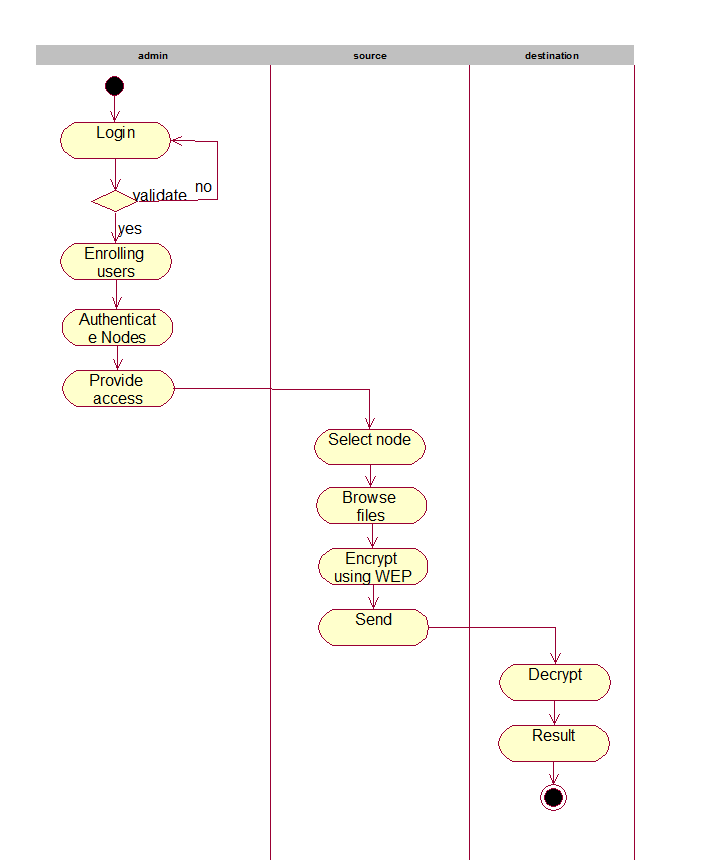


Figure 4.7: Activity Diagram

**5.SAMPLE CODE**

**Form1.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Security.Cryptography;

using System.Data.SqlClient;

namespace NodeA

{

public partial class Form1 : Form

{

private string source = “A”; //Source Node

private string dest = “”;

private bool enc = false;

private bool dec = false; // 4 WEP keys

private List<string> keySet1 = new List<string> { “2e3f4”, “5y7js”, “24fg7”, “98jui” };

private List<string> keySet2 = new List<string> { “0x123456789A”, “0x23456789AB”,

“0x3456789ABC”, “0x456789ABCD” };

private List<string> keySet3 = new List<string> { “2e3f4w345ytre”, “5y7jse8r4i038”,

“24fg70okx3fr7”, “98jui2wss35u4” };

private List<string> keySet4 = new List<string> {

“0x112233445566778899AABBCDEF”, “0x2233445566778899AABBCCDDEE”,

“0x3344556677889900AABBCCDDFF”, “0x44556677889900AABBCCDDEEFF” };

private string key;

private string cipher;

public Form1()

{

InitializeComponent();

SqlConnection con = Database.Connection; //Connectivity with SQL server

con.Open();

try

{

String query = “select node from nodes where auth=’y’ and node!=@node”;

SqlCommand cmd = new SqlCommand(query, con);

cmd = new SqlCommand(query, con);

cmd.Parameters.AddWithValue(“@node”, “NodeA”);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

listBoxDest.Items.Add((String)reader[“node”]);

}

}

catch (Exception ex)

{

MessageBox.Show(“Could not get destinations “ + ex.Message + “=====🡺 “ +

ex.StackTrace);

}

finally

{

if (con != null)

{

con.Close();

}

}

}

private void buttonBrowse\_Click(object sender, EventArgs e) //Browse file to transfer

{

try

{

enc = false;

DialogResult result = openFileDialog1.ShowDialog(); //Shows a list of files

if (result == DialogResult.OK)

{

String ilename = openFileDialog1.FileName; //Selected file is stored

textBoxFN.Text = ilename;

StreamReader reader = new StreamReader(textBoxFN.Text); //Reader is

attached to file

textBoxData.Text = reader.ReadToEnd();

}

}

catch (Exception ex)

{

MessageBox.Show(“Exception in browse file “ + ex.Message + “ ------- >” +

ex.StackTrace);

}

}

private void buttonEnc\_Click(object sender, EventArgs e) //Encryption of selected file

{

try

{

if (textBoxData.Text == String.Empty)

{

MessageBox.Show(“Please enter data in text area”);

return;

}

if (enc)

{

MessageBox.Show(“Data has already encrypted”);

return;

}

if (!radioButton1.Checked && !radioButton2.Checked && !radioButton3.Checked

&& !radioButton4.Checked)

{

MessageBox.Show(“Please enter type of encryption”);

return;

}

if (radioButton1.Checked) //Selection of Type of Encryption

among 4 WEP keys

{

Random random = new Random();

int pos = random.Next(4);

this.key = keySet1[pos];

}

else if (radioButton2.Checked)

{

Random random = new Random();

int pos = random.Next(4);

this.key = keySet2[pos];

}

else if (radioButton3.Checked)

{

Random random = new Random();

int pos = random.Next(4);

this.key = keySet3[pos];

}

else if (radioButton4.Checked)

{

Random random = new Random();

int pos = random.Next(4);

this.key = keySet4[pos];

}

if (!enc)

{

RC4 rc4 = new RC4(this.key, textBoxData.Text);

//RC4 is applied for Encrypting data

this.cipher = RC4.StrToHexStr(rc4.EnDeCrypt());

MessageBox.Show(“Cipher Text is “ + this.cipher); //Shows the Cipher text

textBoxData.Text = cipher;

this.enc = true;

this.dec = false;

}

}

catch (Exception ex)

{

MessageBox.Show(“Exception in Encryption : “ + ex.Message + “---🡪” +

ex.StackTrace);

}

}

private void buttonDec\_Click(object sender, EventArgs e)

{

try

{

if (textBoxData.Text == String.Empty)

{

MessageBox.Show(“Please enter data in text area”);

return;

}

if (dec)

{

MessageBox.Show(“Data has already decrypted”);

return;

}

RC4 rc4 = new RC4(this.key); //RC4 is applied for decrypting data

rc4.Text = RC4.HexStrToStr(this.cipher);

textBoxData.Text = rc4.EnDeCrypt();

dec = true;

enc = false;

}

catch (Exception ex)

{

MessageBox.Show(“Exception in Decryption : “ + ex.Message + “----🡪” +

ex.StackTrace);

}

}

private void buttonSend\_Click(object sender, EventArgs e) //used to send the encrypted

file to destination

{

try

{

this.dest = “”;

string data = textBoxData.Text;

if (data == string.Empty)

{

MessageBox.Show(“Enter data to send”);

return;

}

if (!enc)

{

MessageBox.Show(“Plese encrypt the data”);

return;

}

foreach (object item in listBoxDest.SelectedItems)

{

MessageBox.Show(“Dest “ + item.ToString());

this.dest += item.ToString();

}

if (this.dest == String.Empty)

{

MessageBox.Show(“Could not connect to any node”);

return;

}

SqlConnection con = Database.Connection;

con.Open();

SqlTransaction trans = con.BeginTransaction();

try

{

String delete = “delete info”;

String insert = “insert into

info(src,cipher,datakey,dest)values(@src,@cipher,@dataKey,@dest)”;

SqlCommand cmd = new SqlCommand(delete, con);

cmd.Transaction = trans;

cmd.ExecuteNonQuery();

cmd = new SqlCommand(insert, con);

cmd.Transaction = trans;

cmd.Parameters.AddWithValue(“@src”, this.source);

cmd.Parameters.AddWithValue(“@cipher”, this.cipher);

cmd.Parameters.AddWithValue(“@dataKey”, this.key);

cmd.Parameters.AddWithValue(“@dest”, this.dest);

cmd.ExecuteNonQuery();

trans.Commit();

MessageBox.Show(“Data Sent Successfully…....”);

}

catch (Exception ex)

{

trans.Rollback();

MessageBox.Show(“Failure in DataSending “ + ex.Message + “=====🡺 “ +

ex.StackTrace);

}

finally

{

if (con != null)

{

con.Close();

}

}

}

catch (Exception ex)

{

MessageBox.Show(“Exception in NodeA.Send() “ + ex.Message +

“==========🡺” + ex.StackTrace);

}

}

private void buttonRec\_Click(object sender, EventArgs e) // To receive a file from

sender

{

dec = false;

SqlConnection con = Database.Connection;

try

{

con.Open();

String query = “select src,cipher,datakey,dest from info”;

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader reader = cmd.ExecuteReader();

cmd = new SqlCommand(query, con);

if (reader.Read())

{

this.source = (string)reader[“src”];

this.cipher = (string)reader[“cipher”];

this.key = (string)reader[“dataKey”];

this.dest = (string)reader[“dest”];

}

if (!this.dest.Contains(“A”))

{

MessageBox.Show(“Nothing To Recieve….......”);

return;

}

String cipherData = System.Text.Encoding.ASCII.GetString

(System.Text.Encoding.ASCII.GetBytes(this.cipher));

textBoxData.Text = cipherData;

}

catch (Exception ex)

{

MessageBox.Show(“Failure in DataReceving “ + ex.Message + “=====🡺 “ +

ex.StackTrace);

}

finally

{

if (con != null)

{

con.Close();

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void listBoxDest\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

**Program.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace NodeA //Class for Node A

{

static class Program

{

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**RC4.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace NodeA

{

class RC4 //RC4 Algorithm

{

private const int N = 256;

private int[] sbox; //Temporary Vector

private string password;

private string text;

public RC4(string password, string text)

{

this.password = password;

this.text = text;

}

public RC4(string password)

{

this.password = password;

}

public string Text

{

get { return text; }

set { text = value; }

}

public string Password

{

get { return password; }

set { password = value; }

}

public string EnDeCrypt() //For encrypting and Decrypting the data

{

RC4Initialize(); //First Step of RC4 Algorithm: initialization

int i = 0, j = 0, k = 0;

StringBuilder cipher = new StringBuilder();

for (int a = 0; a < text.Length; a++) //Second step of RC4 Algorithm: Operation

{

i = (i + 1) % N;

j = (j + sbox[i]) % N;

int tempSwap = sbox[i];

sbox[i] = sbox[j];

sbox[j] = tempSwap;

k = sbox[(sbox[i] + sbox[j]) % N];

int cipherBy = ((int)text[a]) ^ k; //xor operation //XOR ed with Key value

cipher.Append(Convert.ToChar(cipherBy));

}

return cipher.ToString(); //Cipher Text

}

public static string StrToHexStr(string str)

{

StringBuilder sb = new StringBuilder();

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

{

int v = Convert.ToInt32(str[i]);

sb.Append(string.Format("{0:X2}", v));

}

return sb.ToString();

}

public static string HexStrToStr(string hexStr)

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < hexStr.Length; i += 2)

{

int n = Convert.ToInt32(hexStr.Substring(i, 2), 16);

sb.Append(Convert.ToChar(n));

}

return sb.ToString();

}

private void RC4Initialize() //First step of RC4 algorithm:Initialization

{

sbox = new int[N];

int[] key = new int[N];

int n = password.Length;

for (int a = 0; a < N; a++)

{

key[a] = (int)password[a % n];

sbox[a] = a;

}

int b = 0;

for (int a = 0; a < N; a++)

{

b = (b + sbox[a] + key[a]) % N;

int tempSwap = sbox[a];

sbox[a] = sbox[b];

sbox[b] = tempSwap;

}

}

}

}

**Database.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient;

namespace NodeA

{

class Database //For the database Connectivity

{

public static SqlConnection Connection {

get {

return new SqlConnection(@"Data Source=localhost\sqlexpress;Initial

Catalog=transmission;Integrated Security=True;Pooling=False");

}

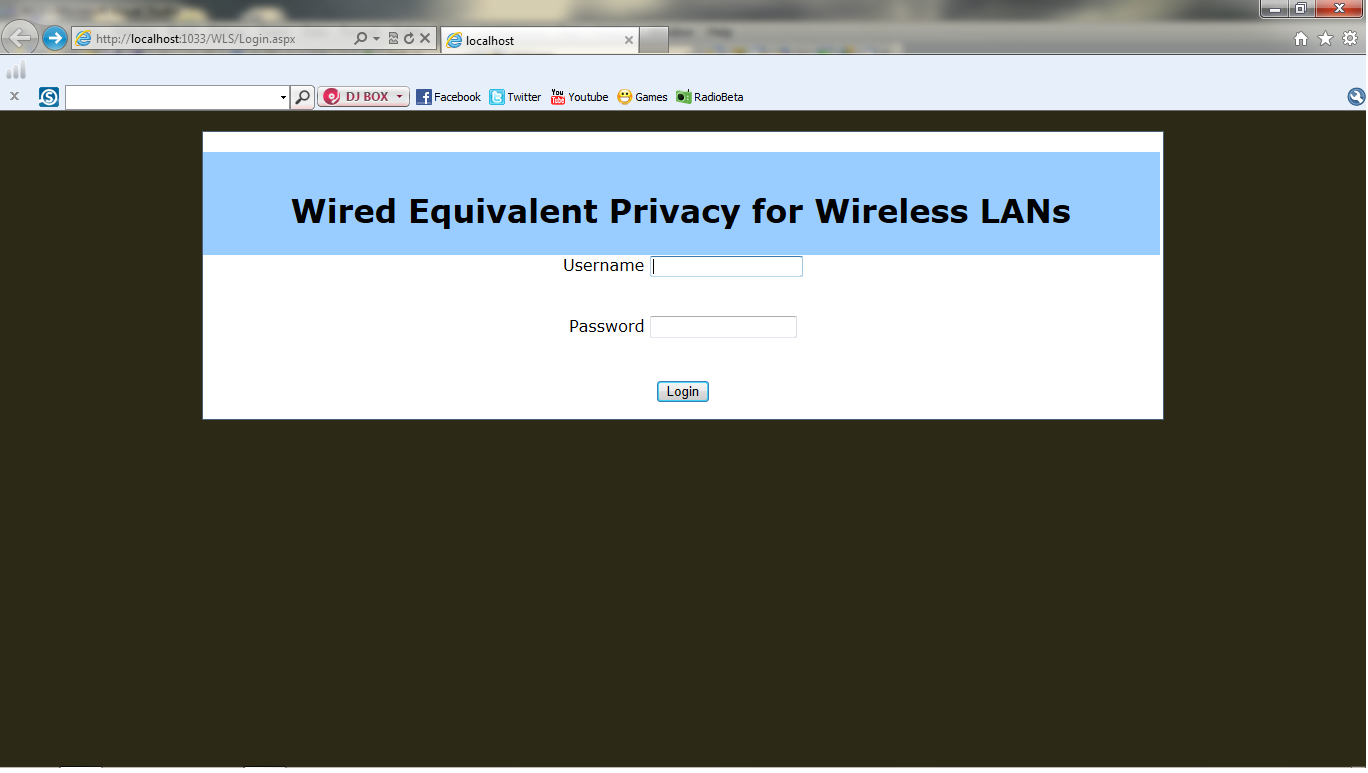
}

}

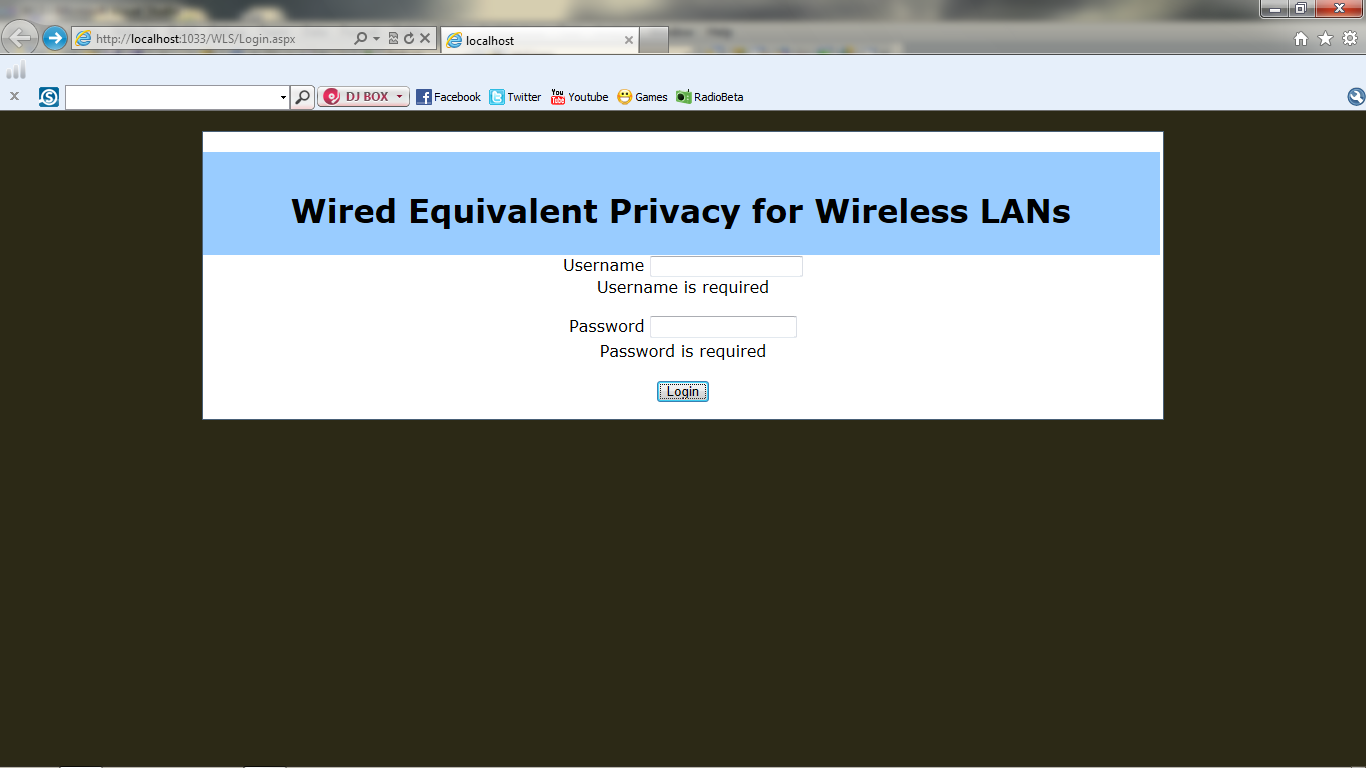
}

**6. SCREEN SHOTS**

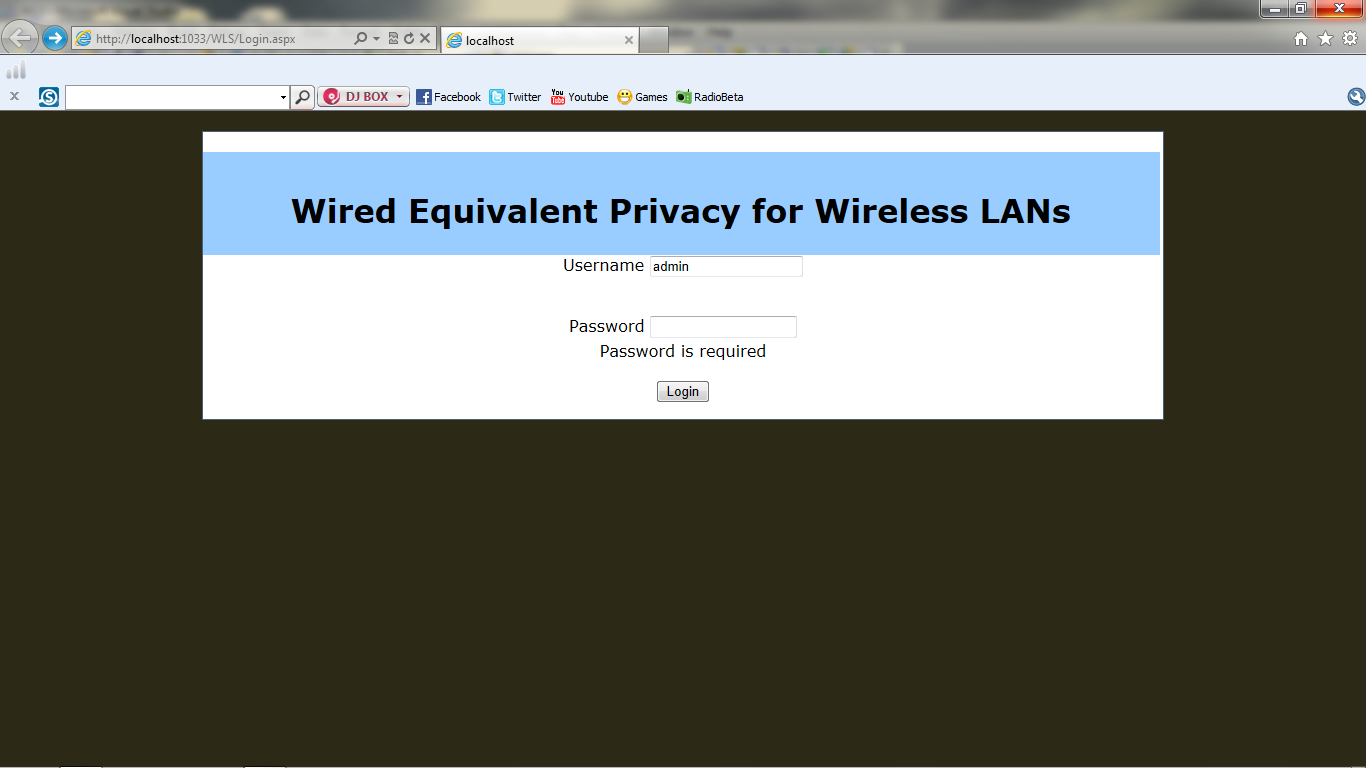
**Login page:**



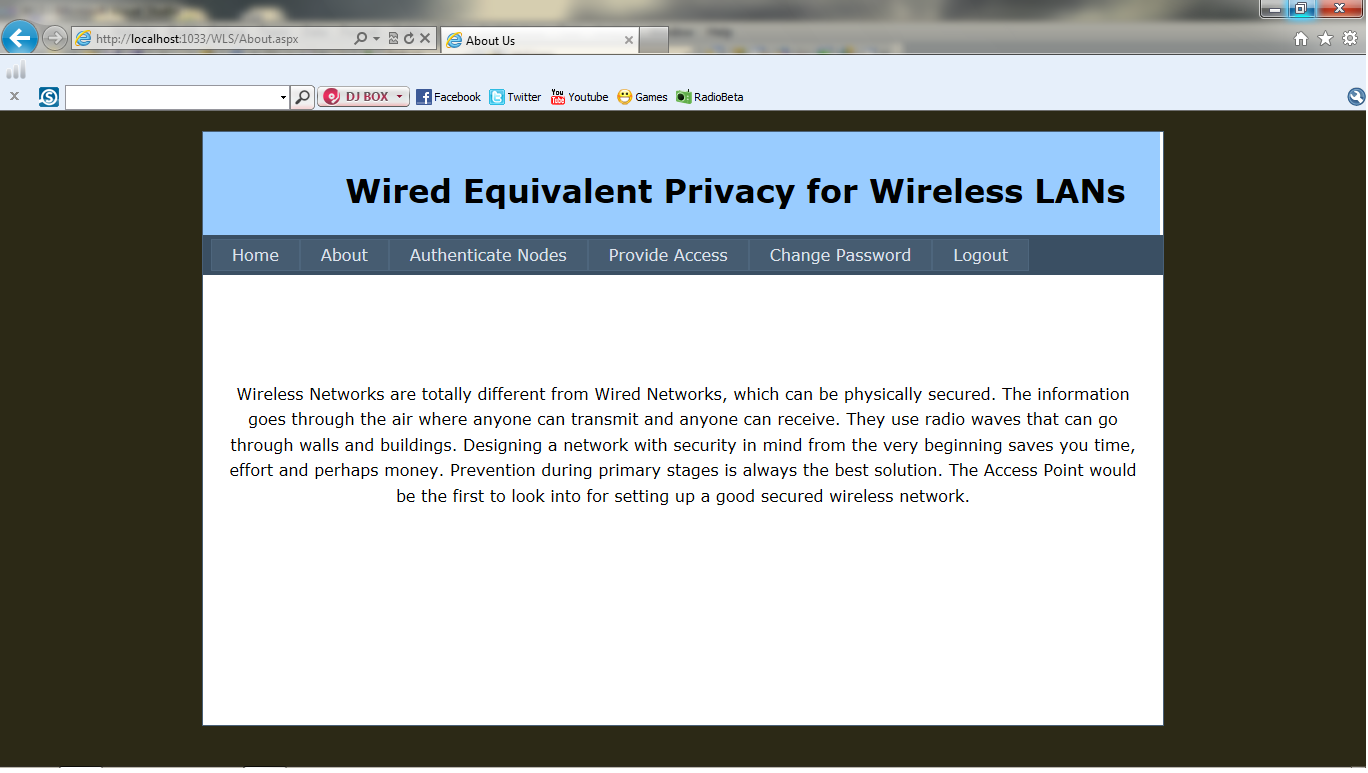
**Login Page:**



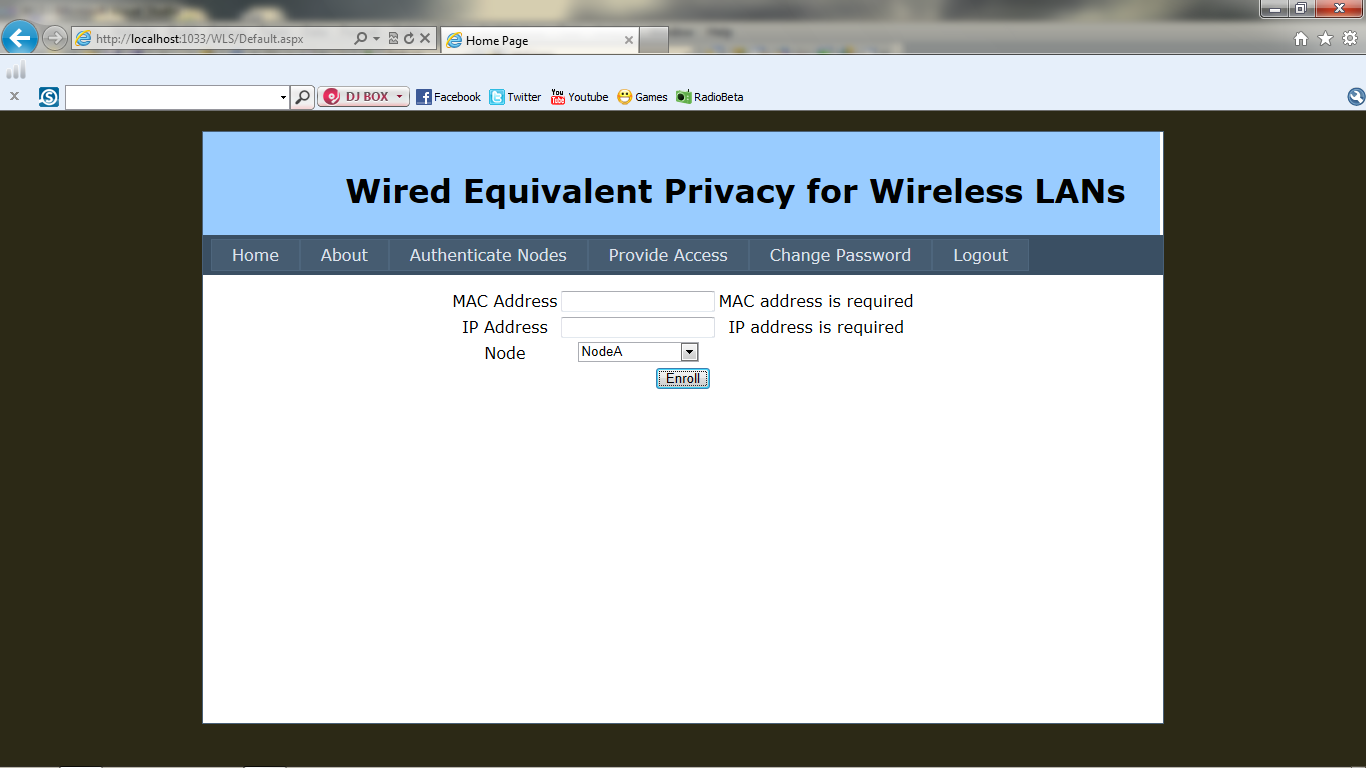
**Login Page:**



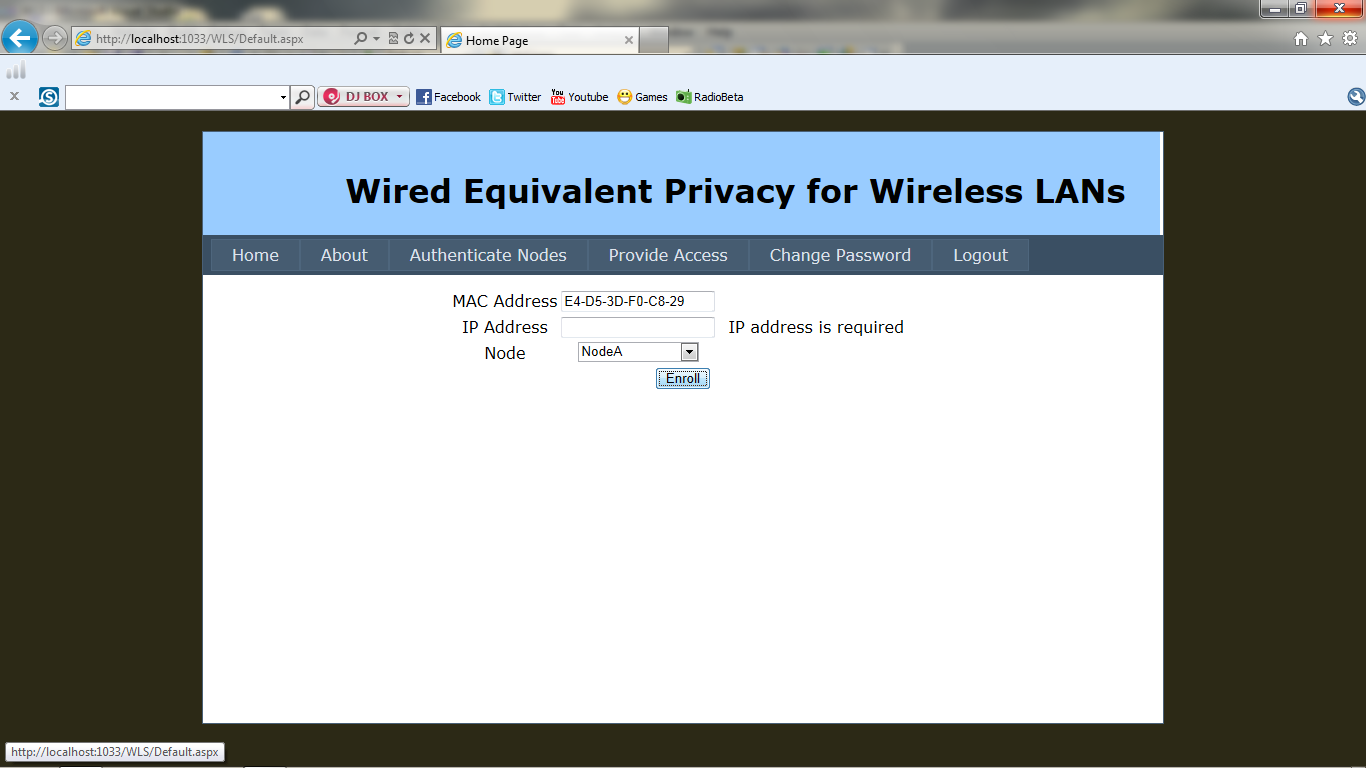
**Home page:**



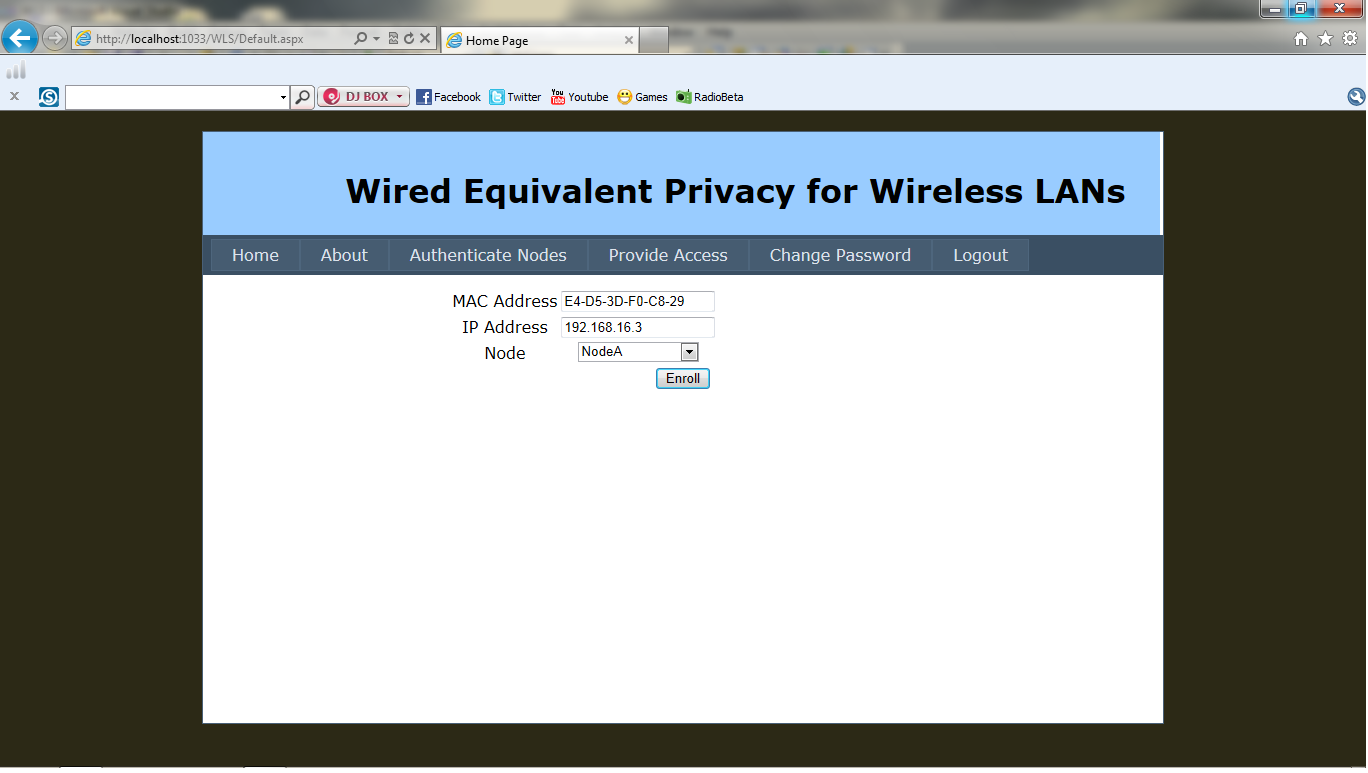
**Home Page:**



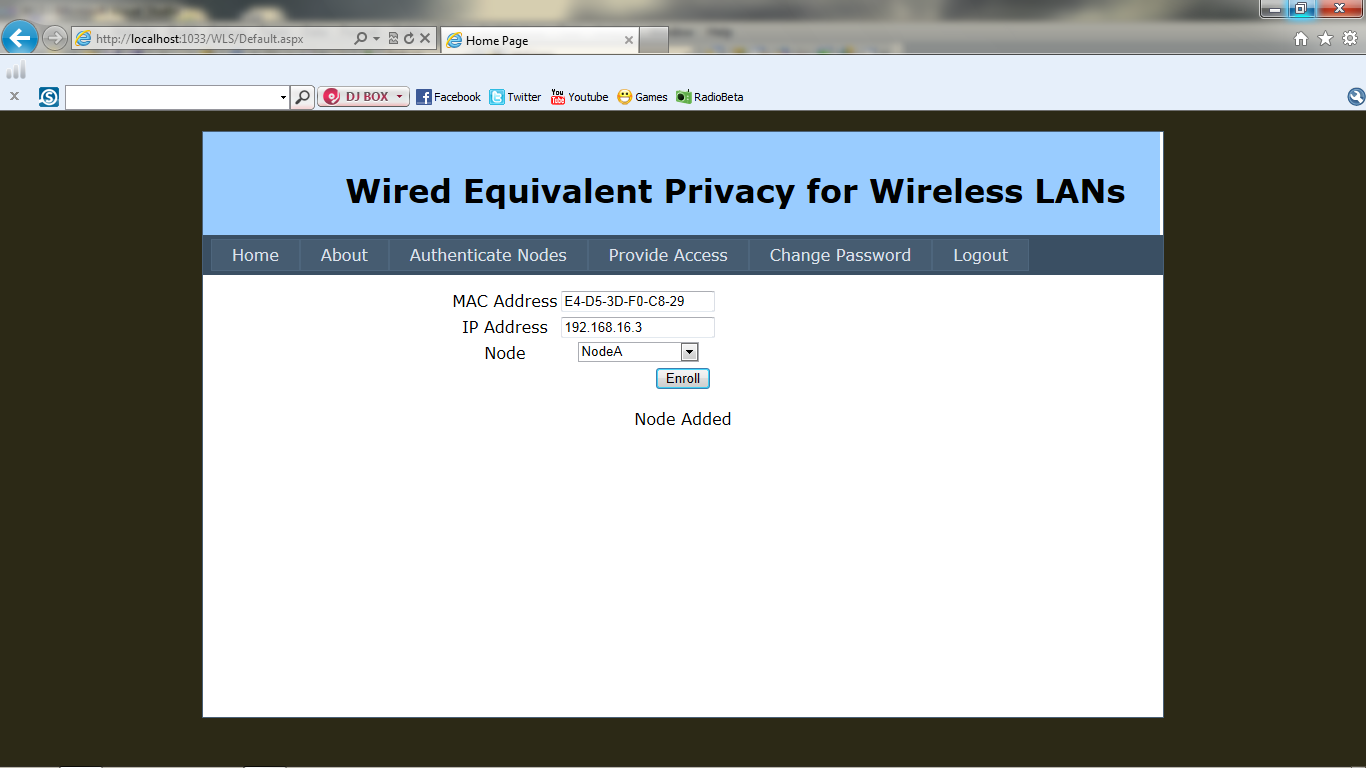
**Adding Nodes Page:**



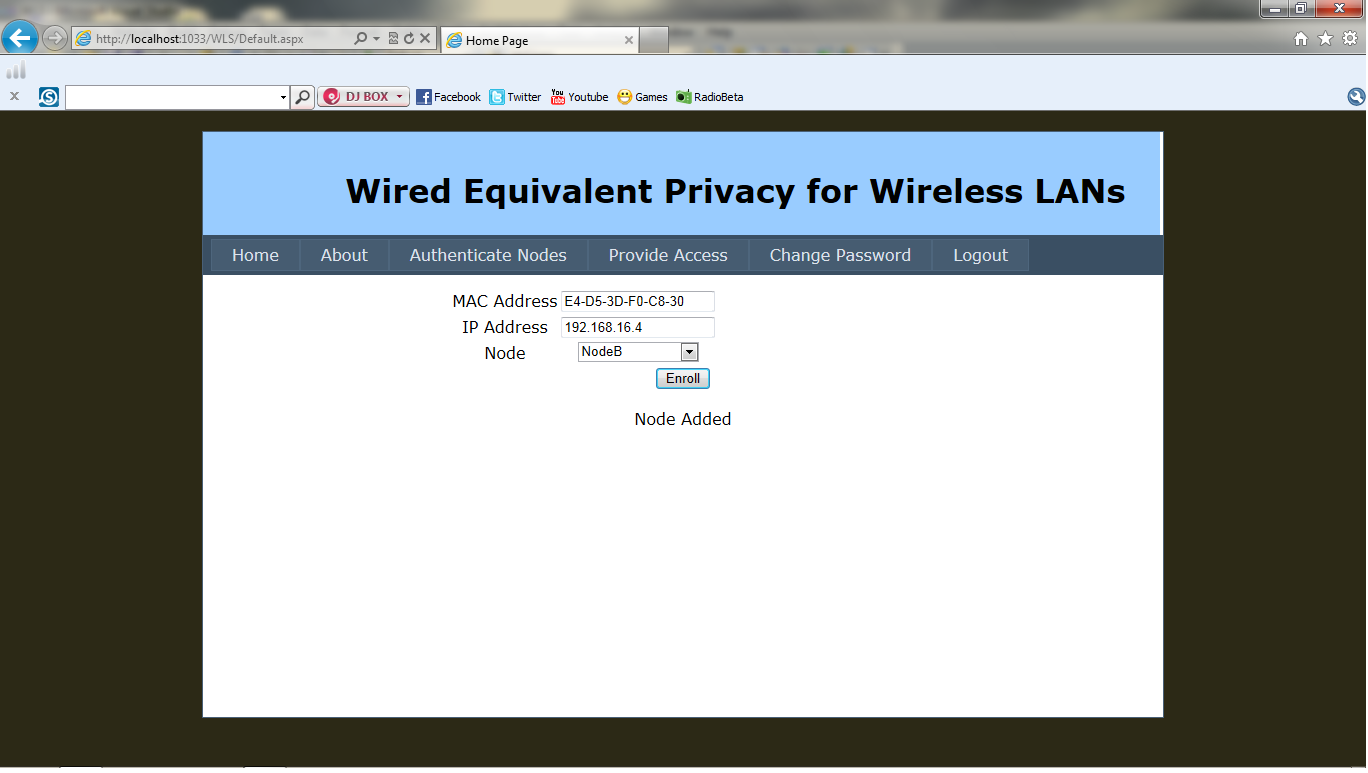
**Adding Nodes Page:**



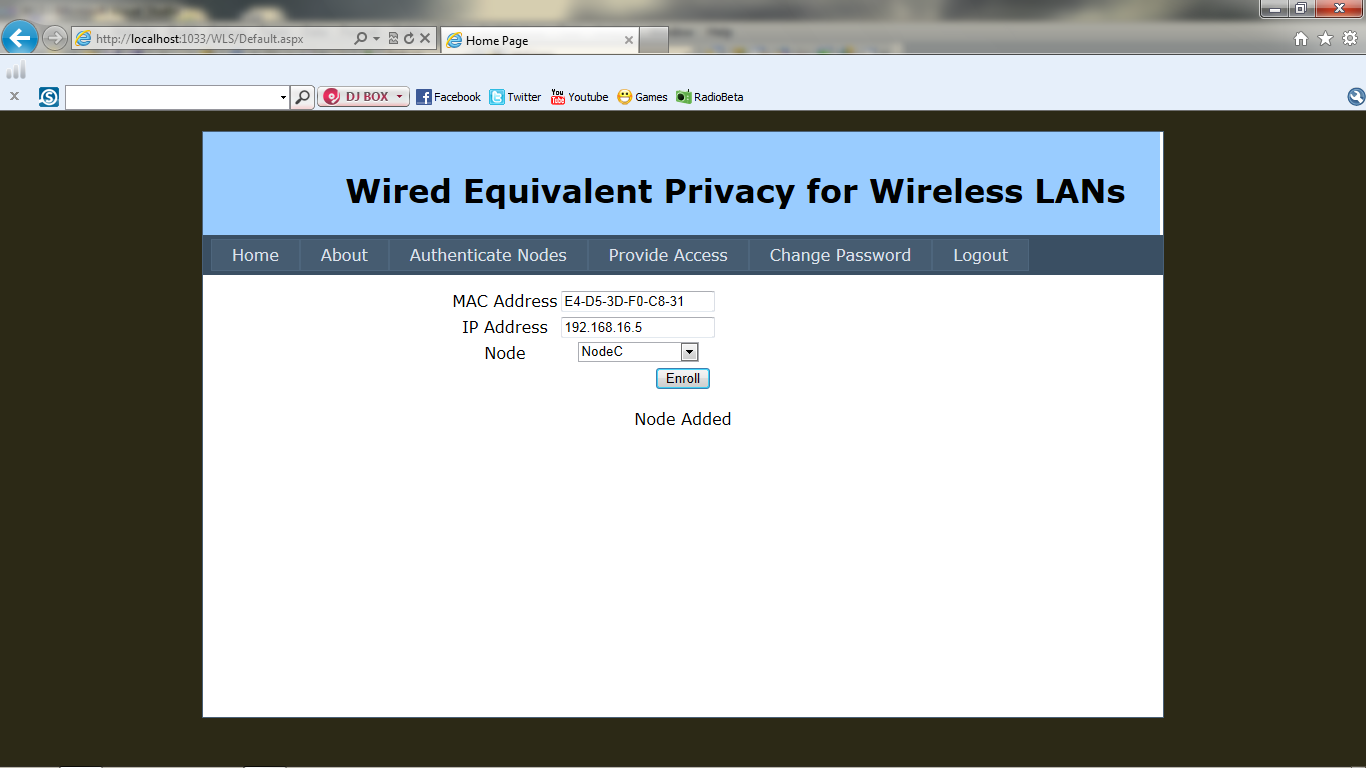
**Adding Nodes Page:**



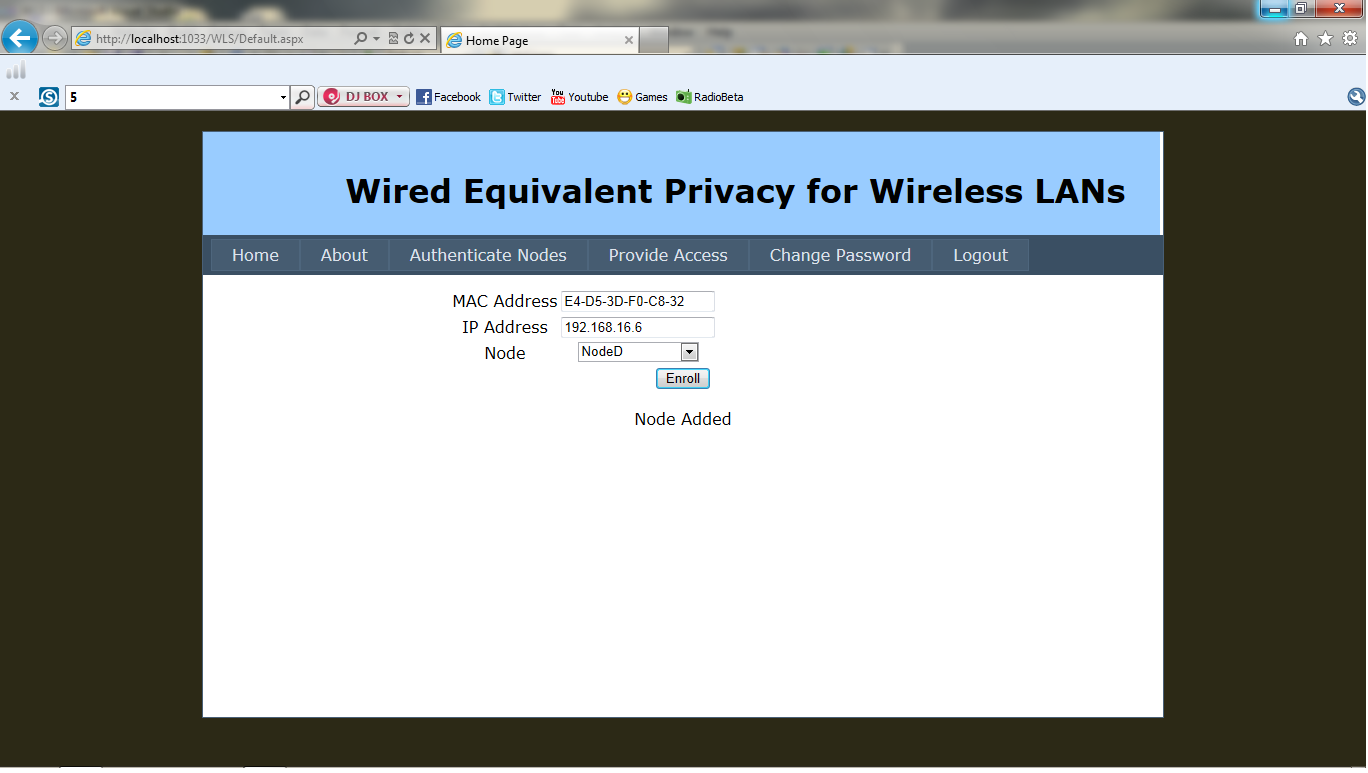
**Adding Nodes Page:**



**Adding Nodes Page:**



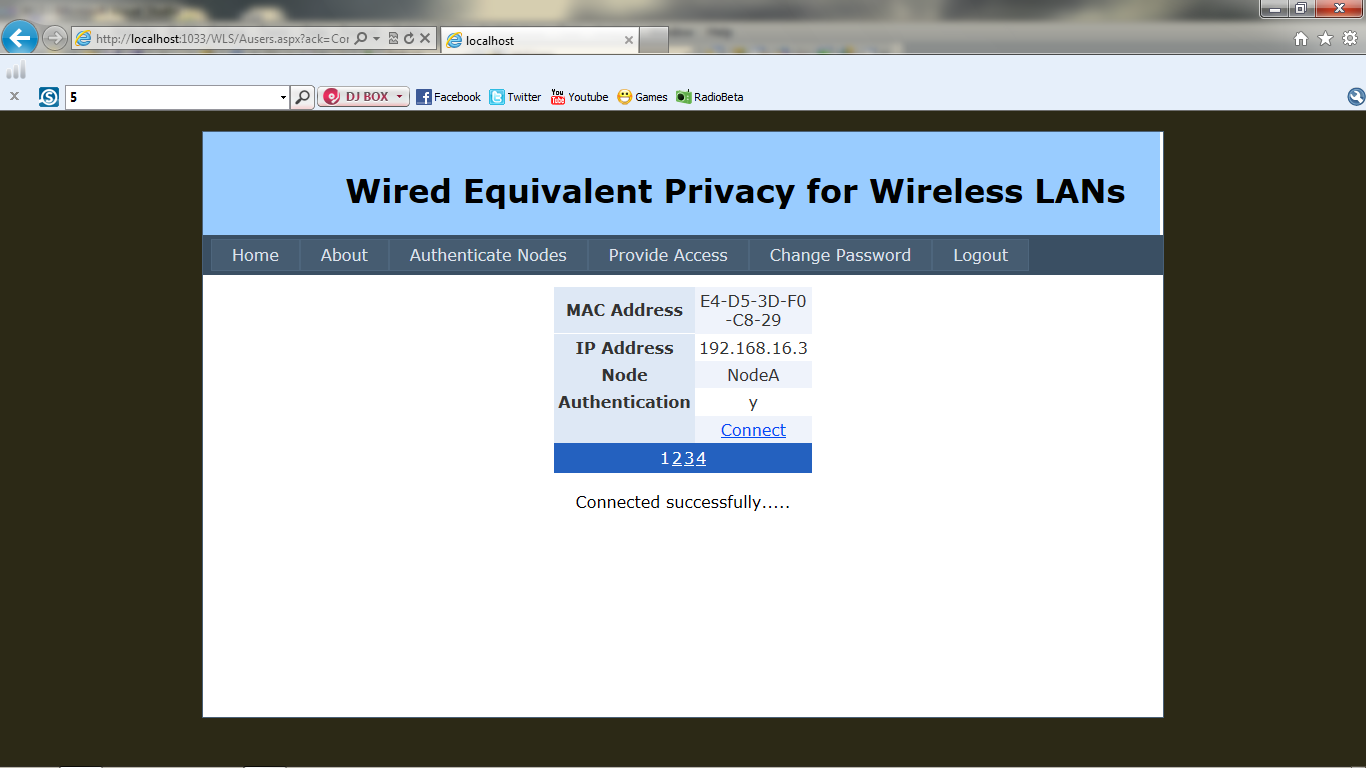
**Adding Nodes Page:**



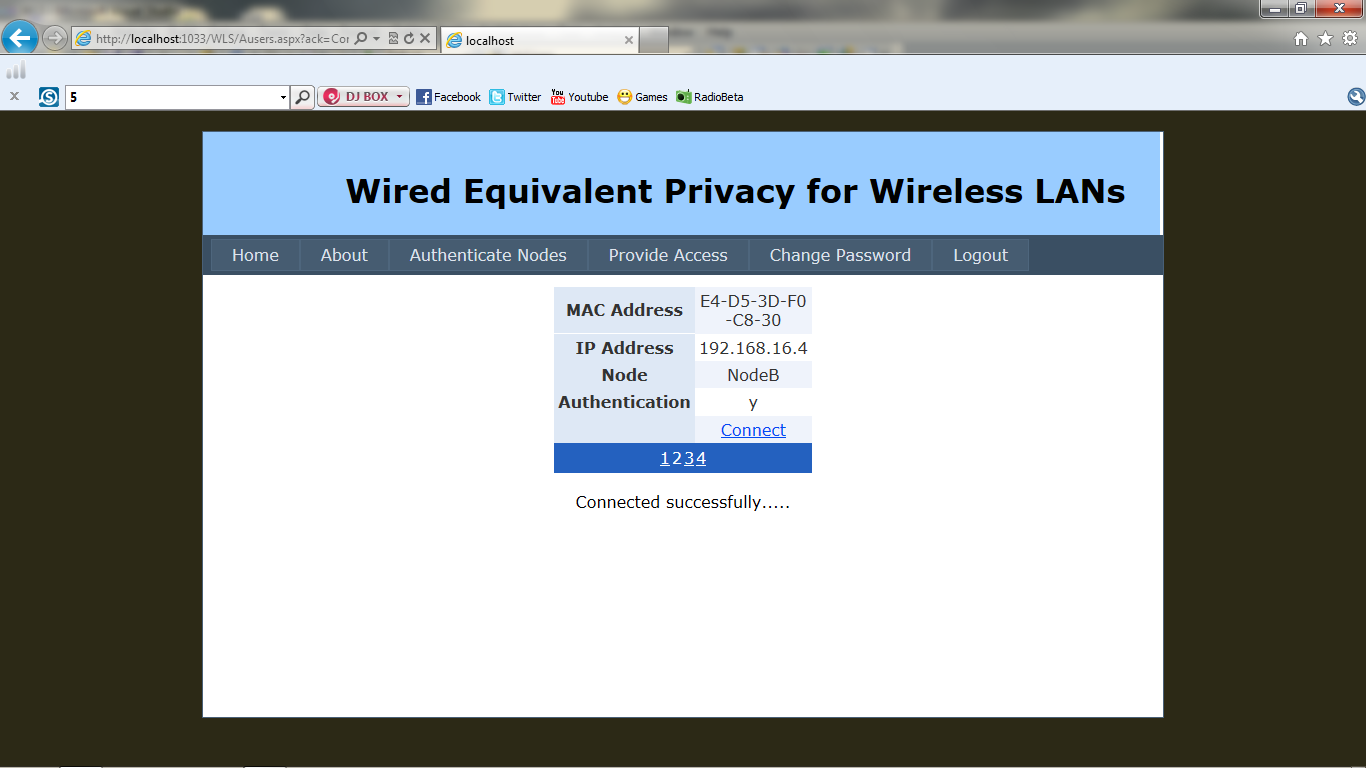
**Authenticate Nodes Page:**



**Authenticate Nodes Page:**



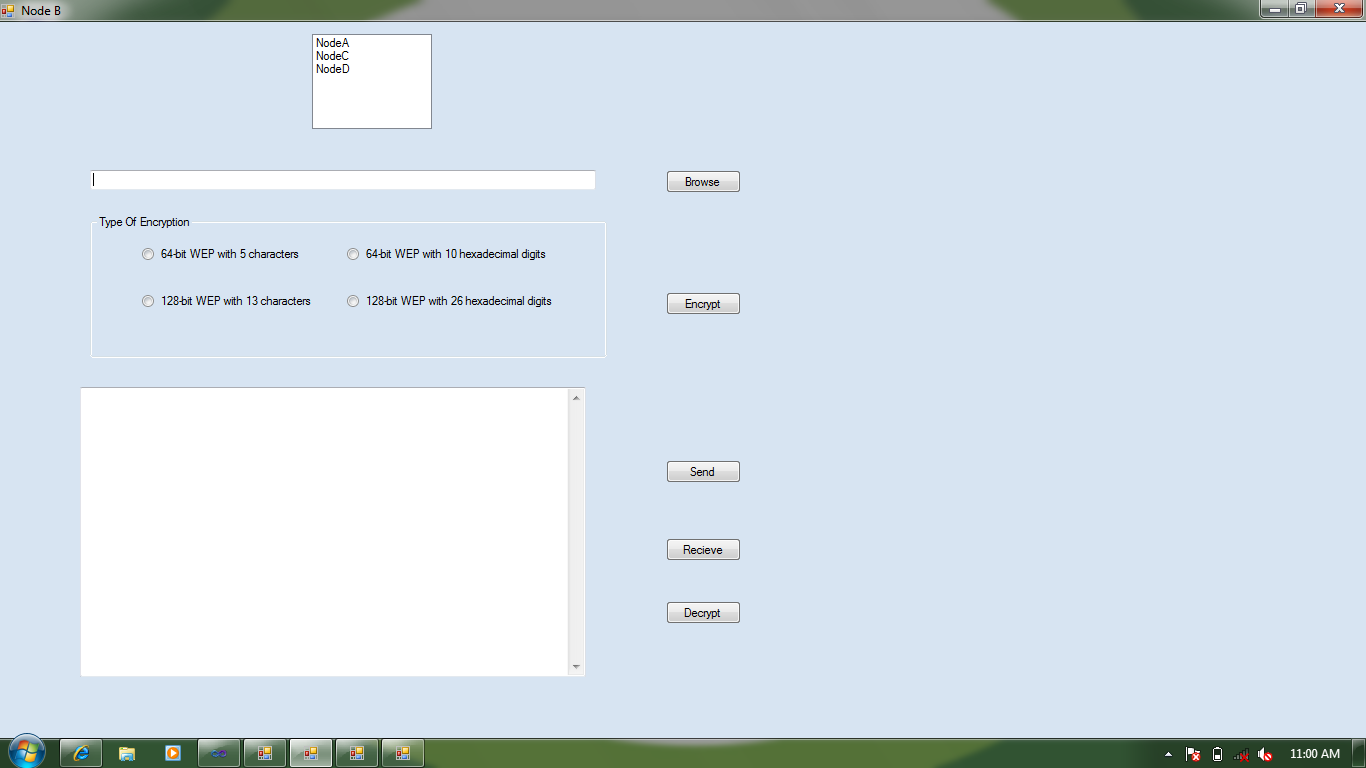
**Providing Access Page:**



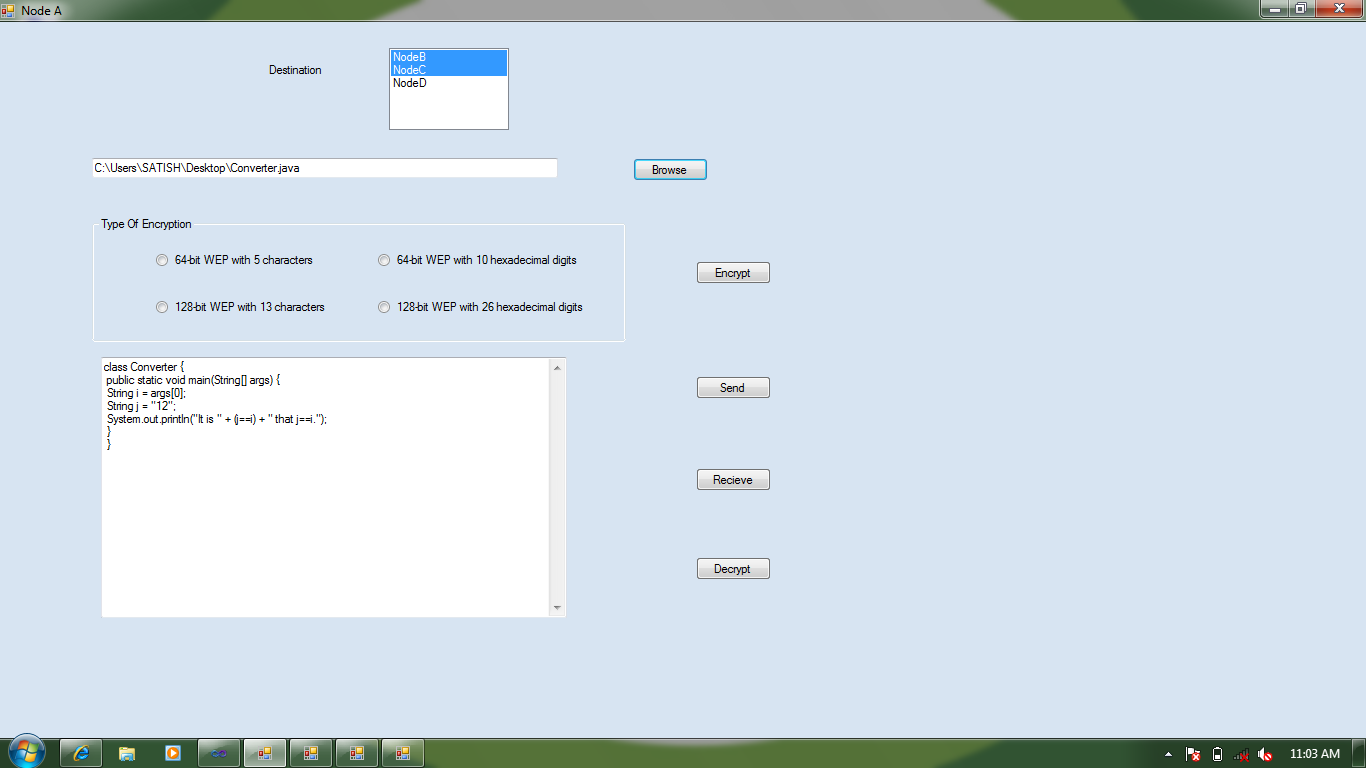
## Node ‘A’ Page:

### 14.png

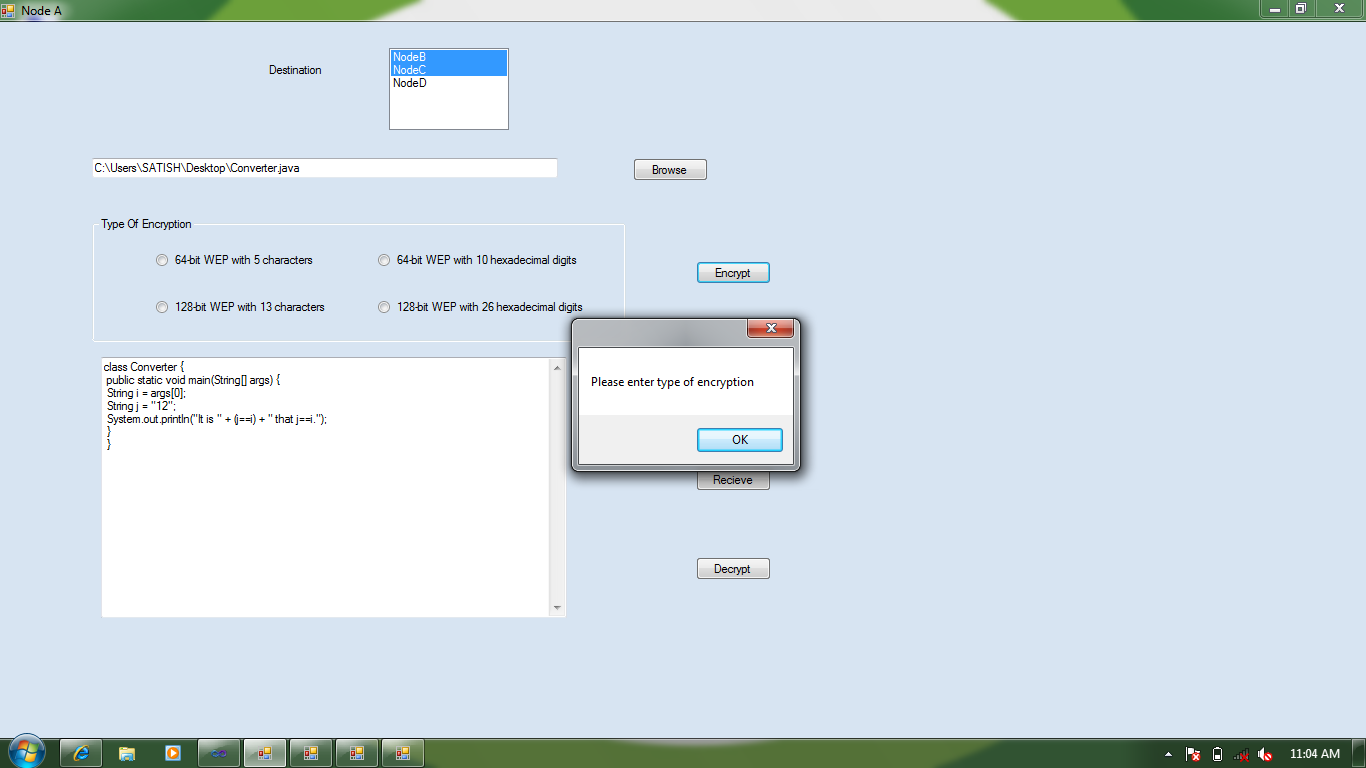
## Node ‘B’ Page:



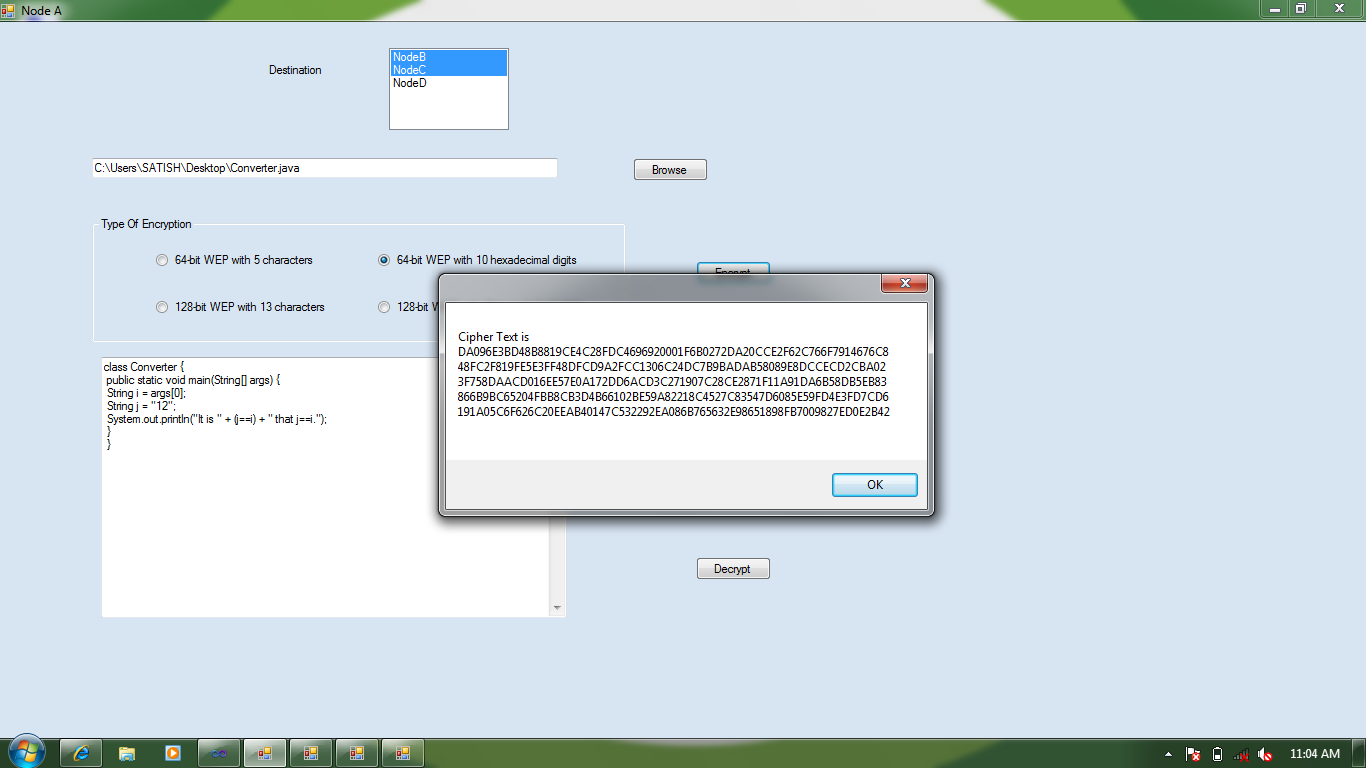
## Node ‘A’ Page:



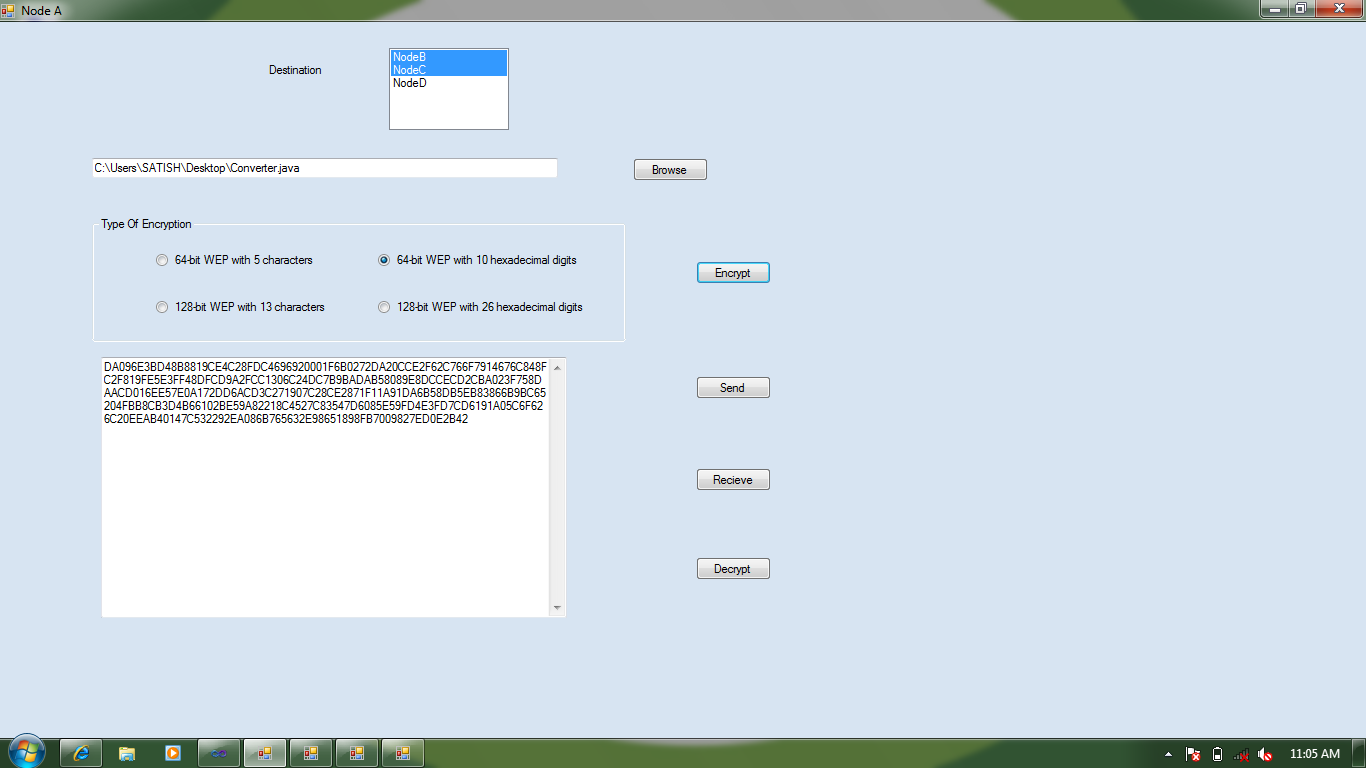
## Node ‘A’ Page:



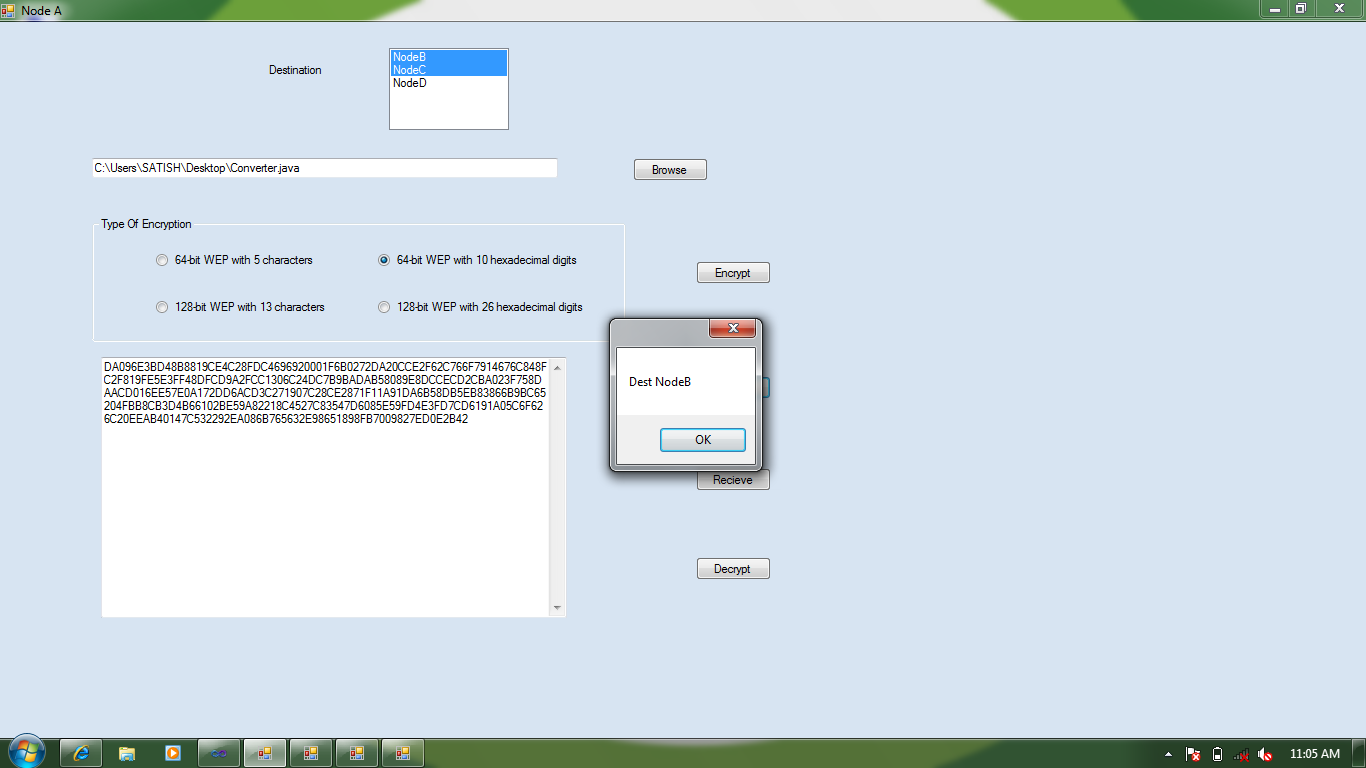
## Node ‘A’ Page:



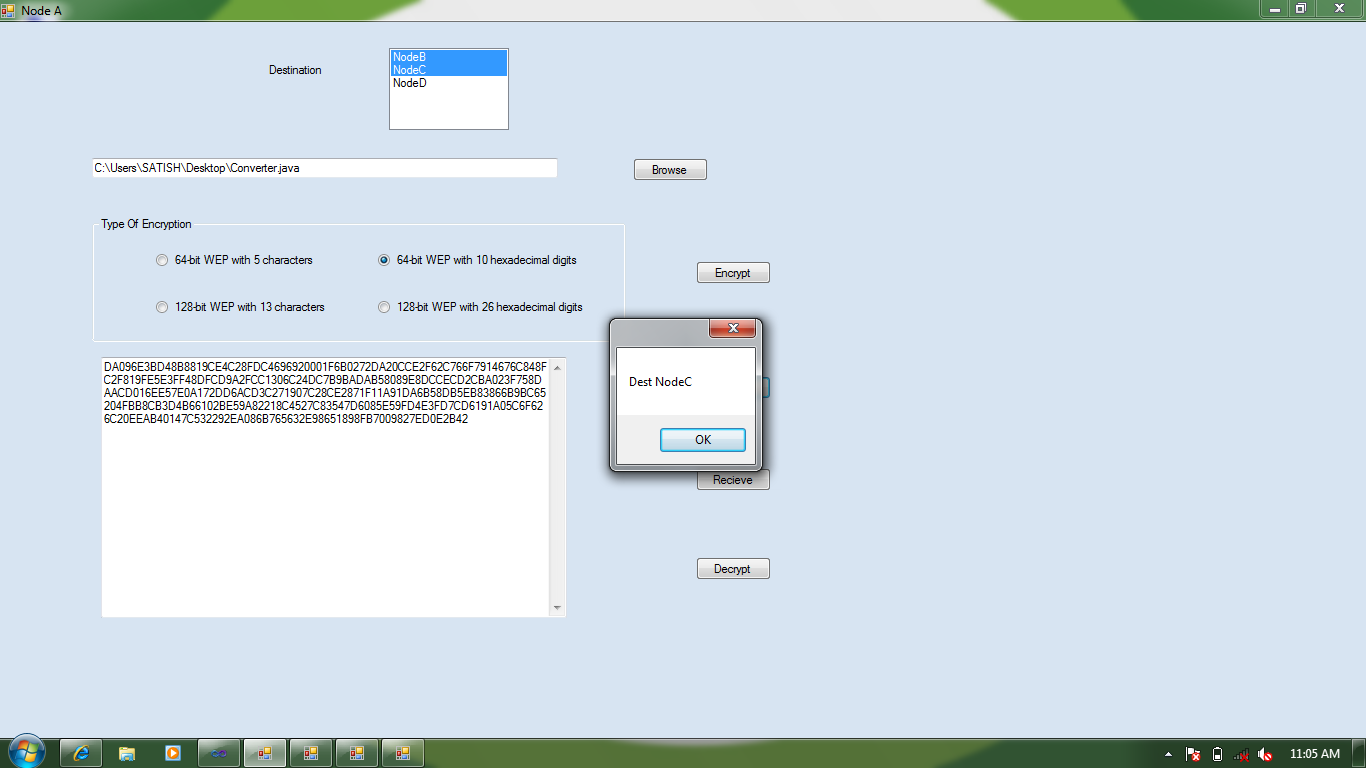
## Node ‘A’ Page:



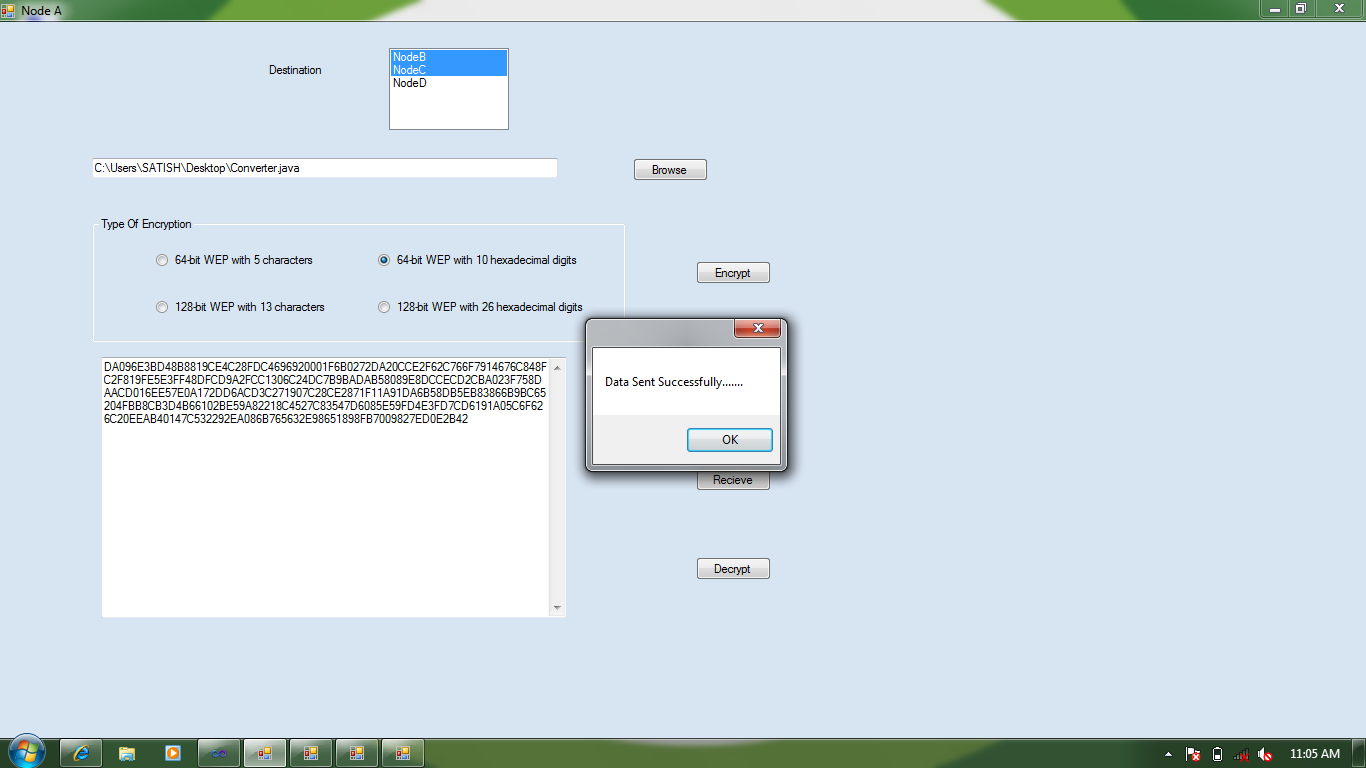
## Node ‘A’ Page:



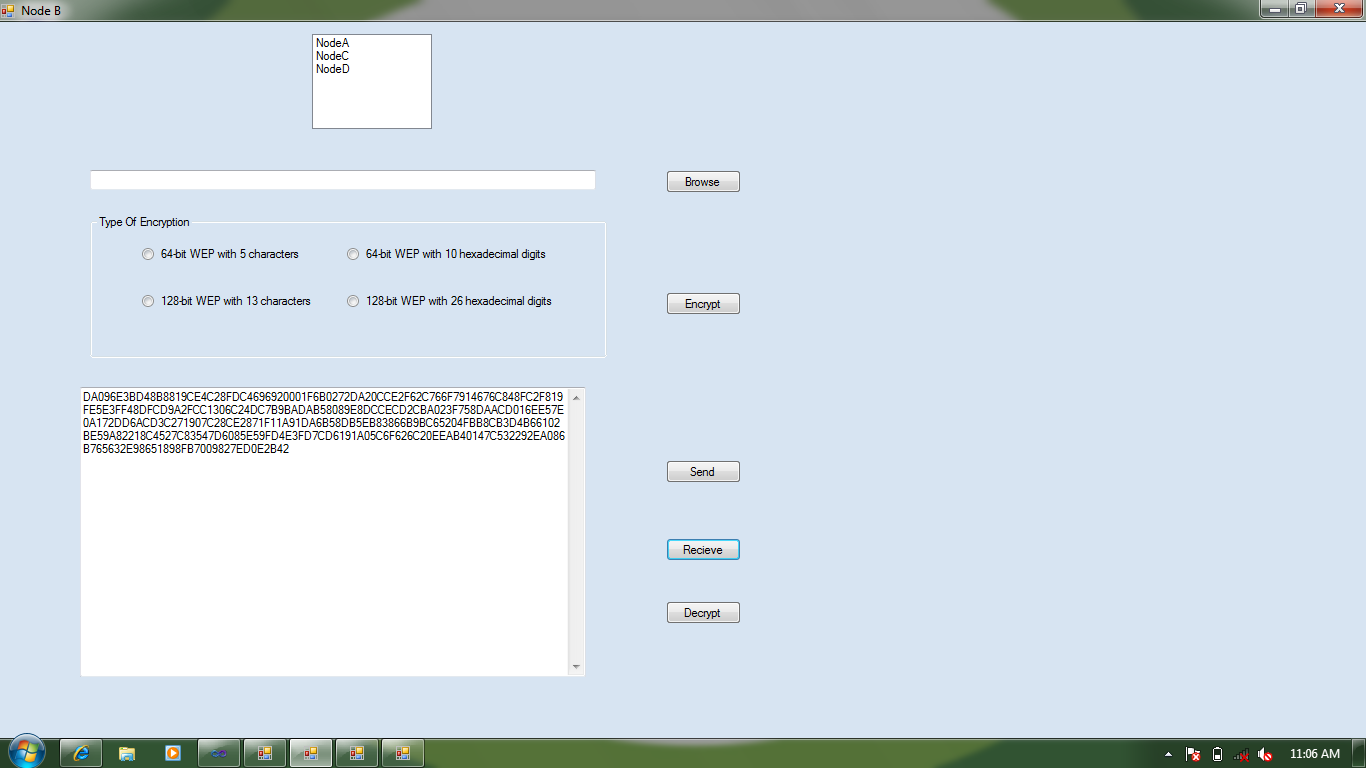
## Node ‘A’ Page:



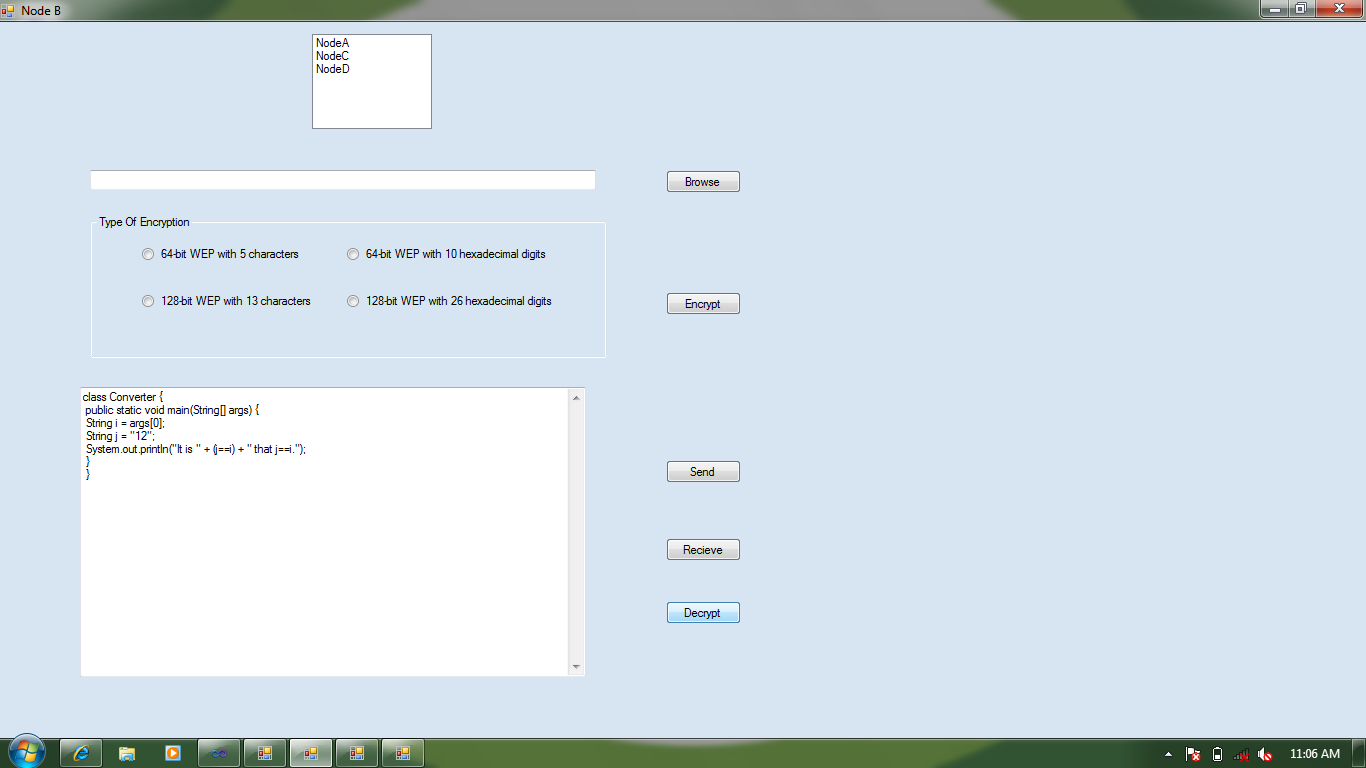
## Node ‘A’ Page:



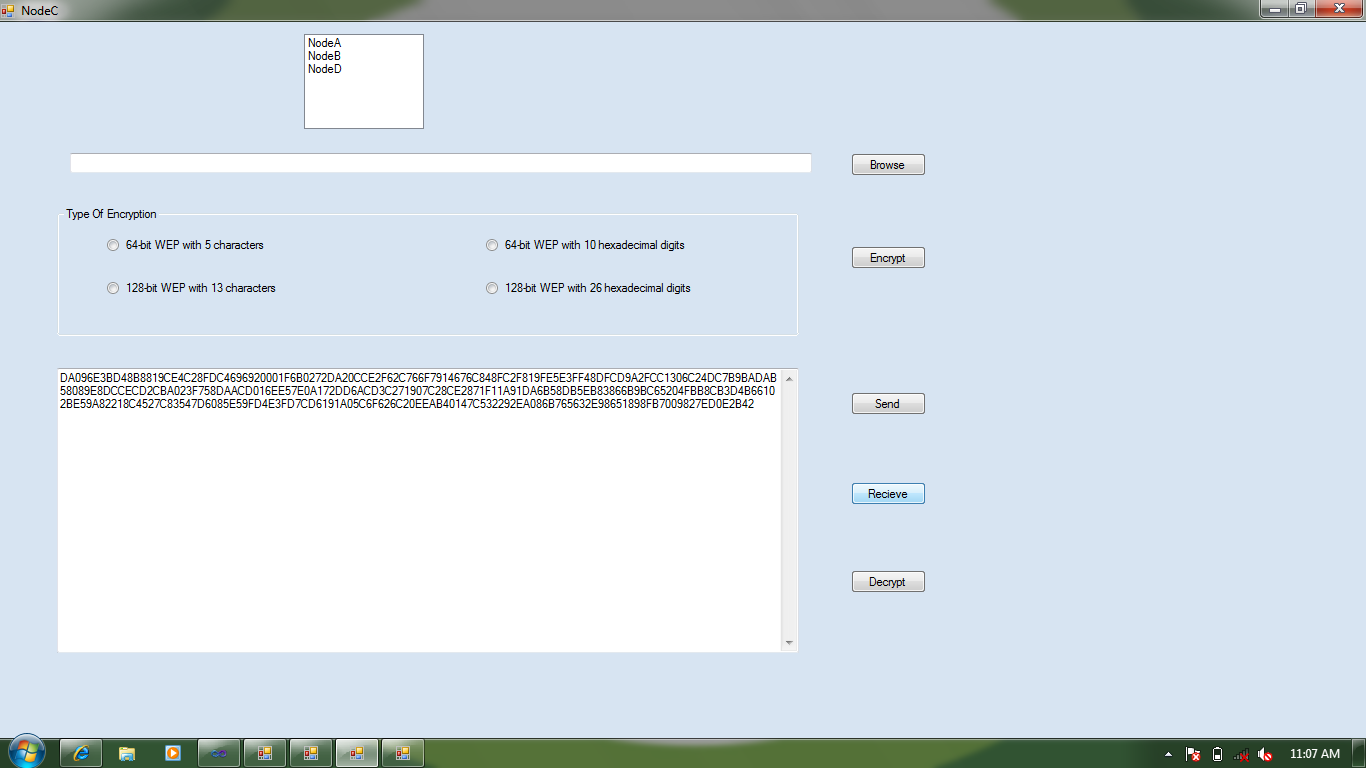
## Node ‘B’ Page:



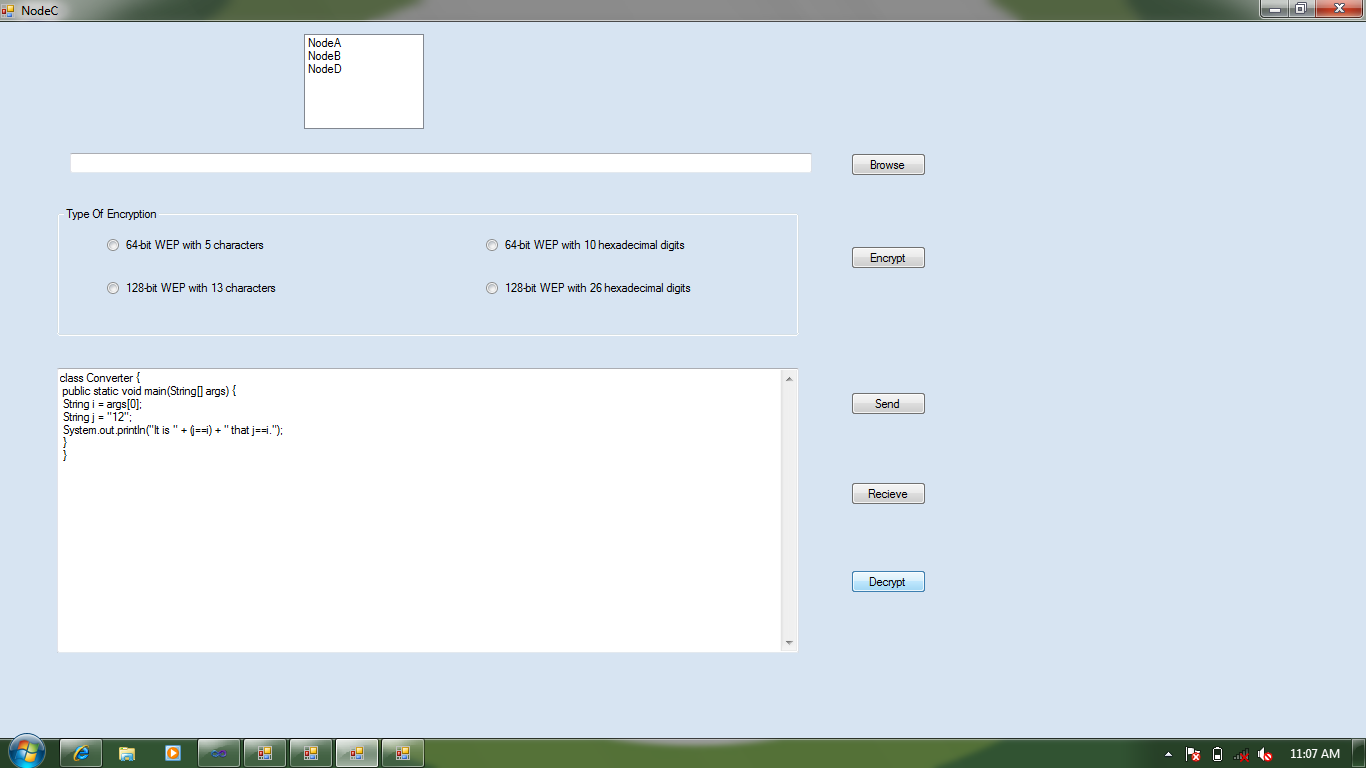
## Node ‘B’ Page:



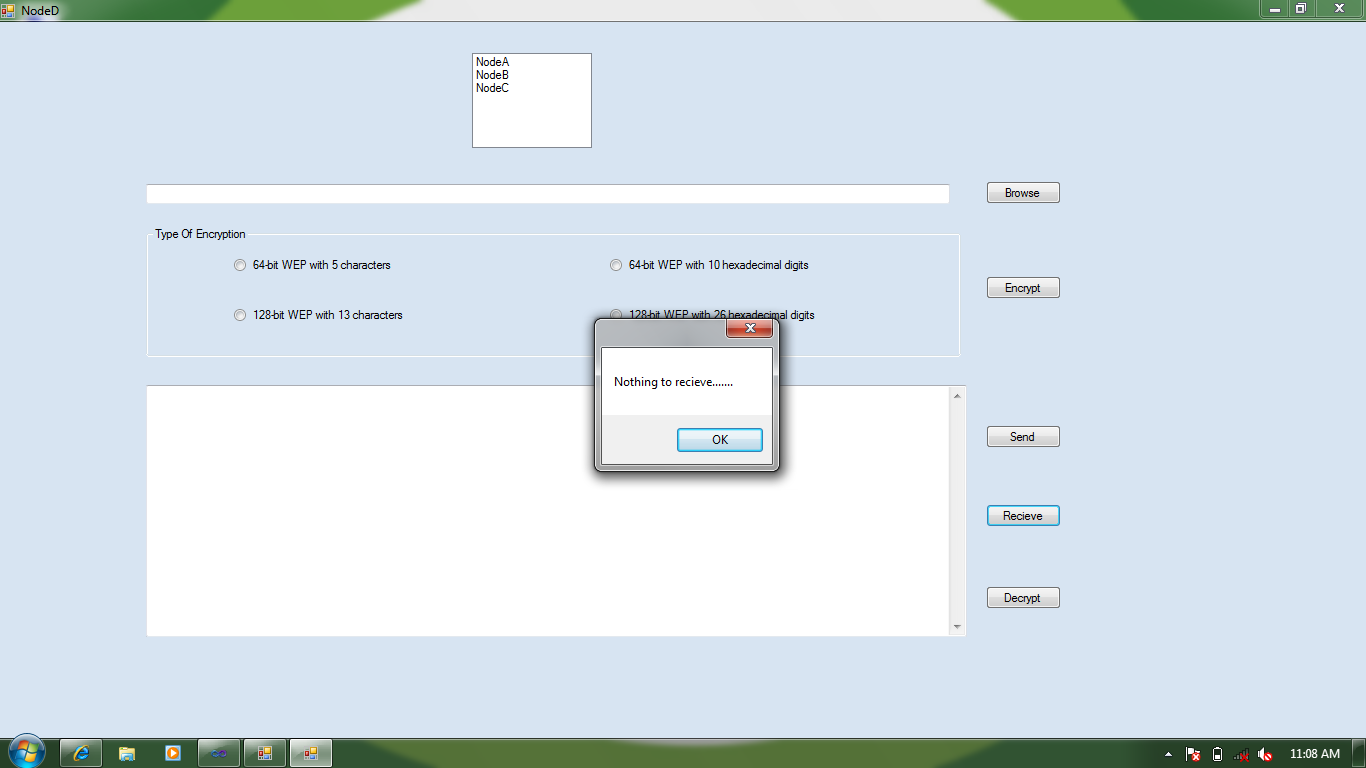
## Node ‘C’ Page:



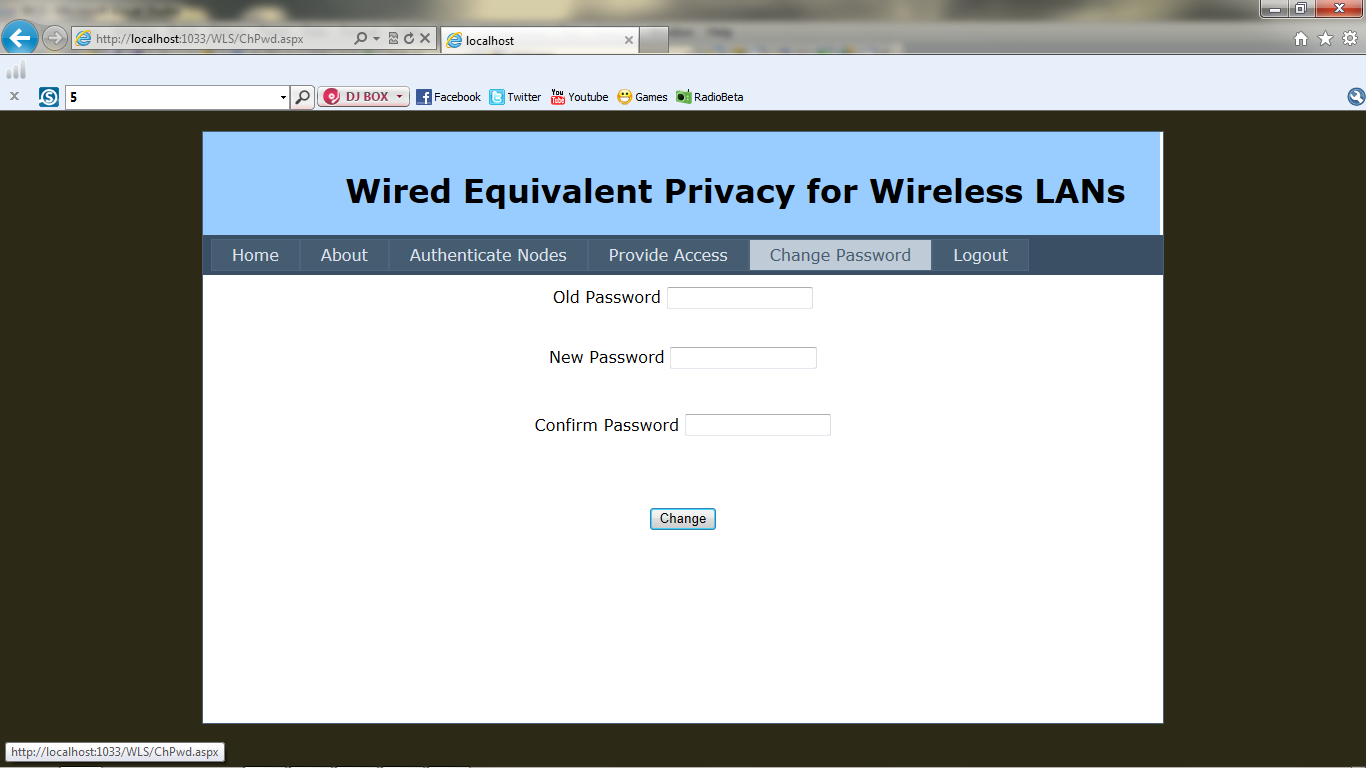
## Node ‘C’ Page:



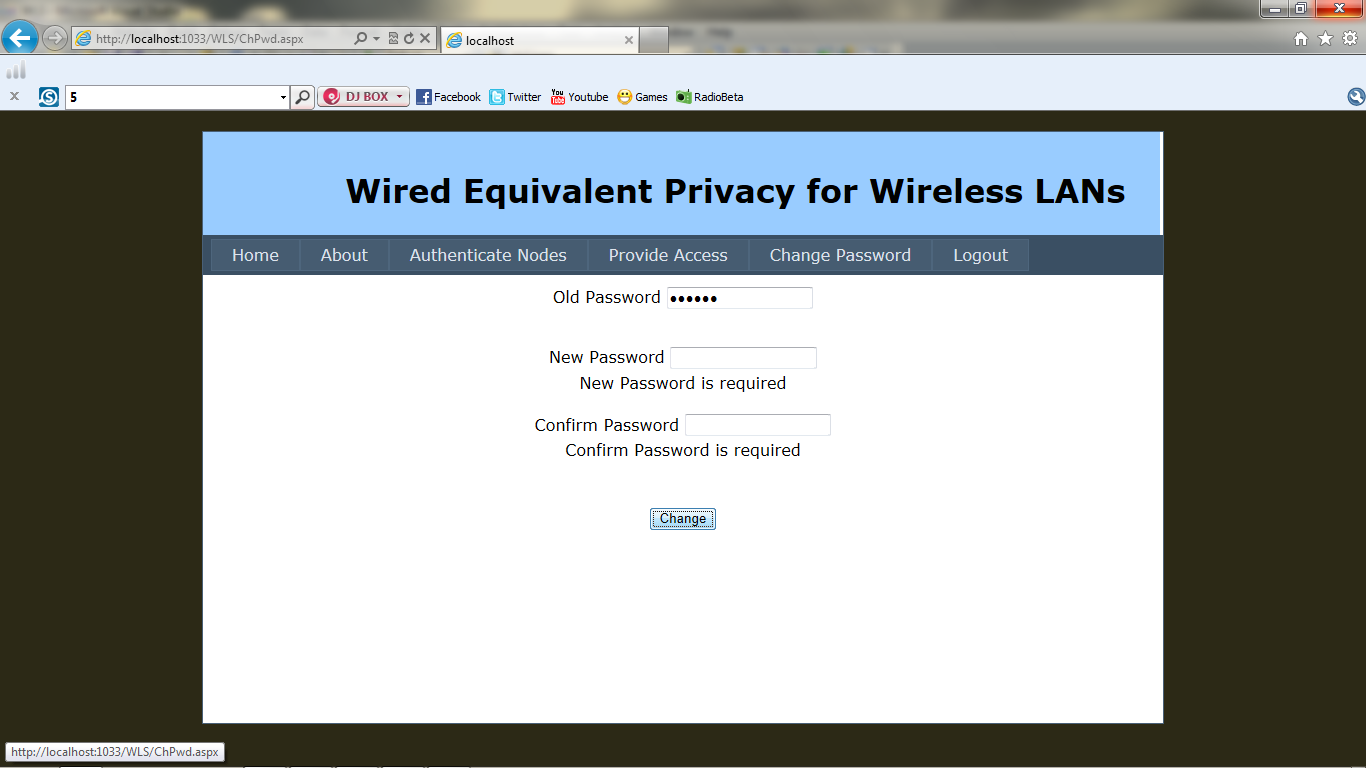
## Node ‘D’ Page:



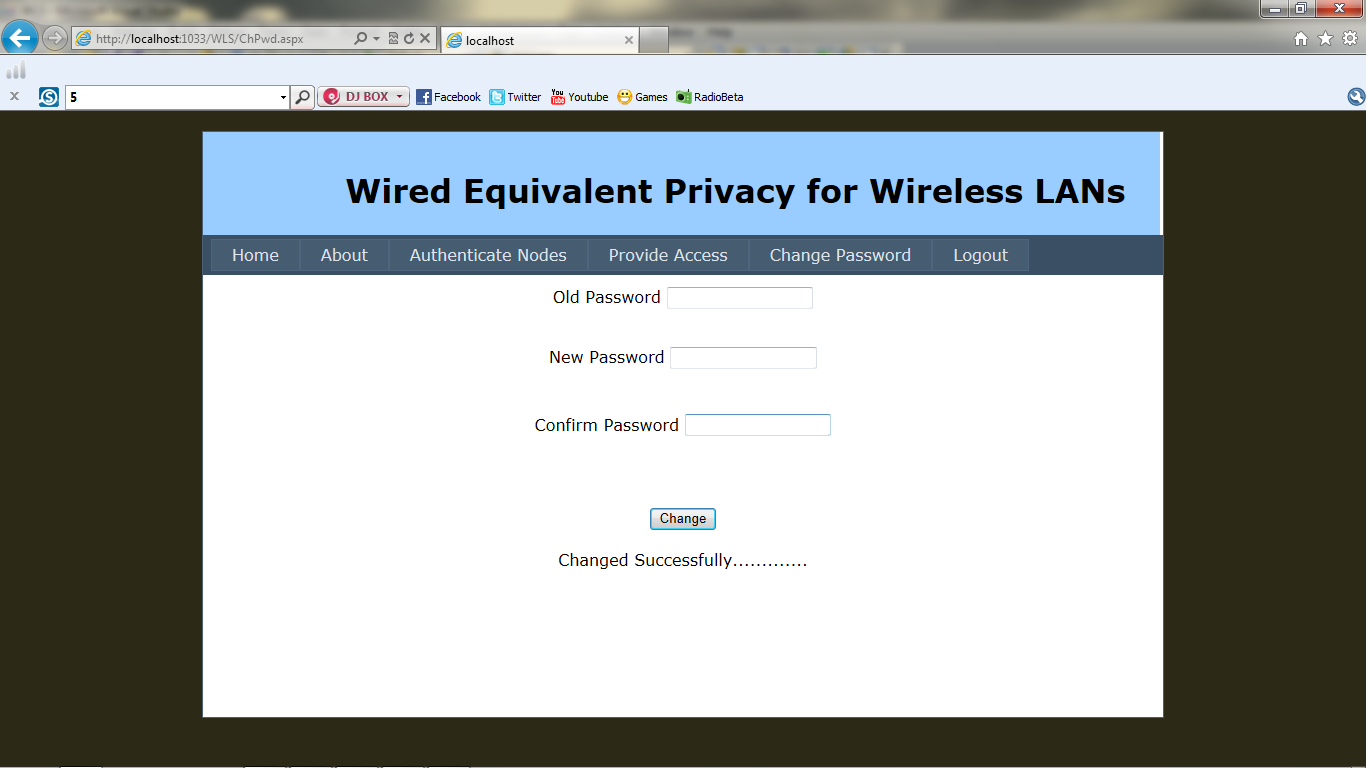
**Change Password Page:**



**Change Password Page:**



**Change Password Page:**



**7. SYSTEM TESTING**

**7.1 Testing:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTING:**

**Unit Testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration Testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**7.2 Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case**  **Id** | **Input Value** | **Test Case**  **Description** | **Expected Value** | **Observed Value** |
| 1. | Click login button in Login page | When the fields filled with proper details | Should successfully go to Home page | Successfully goes to Home page |
| Without filling and improper filling of fields | Error message should be displayed | Error message displayed |
| 2. | Click Authenticate Nodes button in Master page | When the nodes added properly | Should successfully go to Authenticate Nodes Page | Successfully goes to Authenticate Nodes Page |
| Without adding nodes | Error message should be displayed | Error message displayed |
| 3. | Click Provide Access button in Master page | When the nodes added properly | Should successfully go to Provide Access page | Successfully goes to Provide Access page |
| Without adding nodes | Error message should be displayed | Error message displayed |
| 4. | Click Encrypt button in Source Node page | When browse the file | Should successfully encrypt the file | Successfully encrypt the file |
| Without browsing the file | Error message should be displayed | Error message displayed |
| 5. | Click Encrypt button in Source Node page | When select the Type of Encryption | Should successfully Encrypt the File | Successfully Encrypt the File |
| Without selecting Type of Encryption | Error message should be displayed | Error message displayed |
| 6. | Click Send button in Source Node Page | When destination Node is selected | Should successfully send file to selected Destination | Successfully send file to selected Destination |
| Without selecting Destination Node | Error message should be displayed | Error message displayed |
| 7. | Click Receive button in Destination Node Page | When file is sent from source to corresponding Destination | Should successfully receive file to destination | Successfully receive file to destination |
| Without sending file to destination | Error message should be displayed | Error message displayed |
| 8. | Click Decrypt button in Destination Node Page | When file is received at destination | Should successfully decrypt the file at destination | Successfully decrypt the file at destination |
| Without receiving file at destination | Error message should be displayed | Error message displayed |

**8. CONCLUSION**

WLANs are used for its mobility and absence of wires. The clients are authenticated by using MAC address and IP address, so that the Intruders will not be allowed in that network. WEP employs the key encryption algorithm, RC4 for encrypting and decrypting the data. Security is achieved by applying RC4 algorithm.

**8.1 Future Scope:**

The proposed system allows to transfer single file at a time. It can be modified to transfer multiple files at a time.

**9. BIBLIOGRAPHY**

**9.1 Paper References:**

[1] Sia Sie Tung , Nurul Nadia Ahmad, Tan Kim Geok, IEEE 802.11 WEP (Wired

Equivalent Privacy)Concepts for WLAN Security, Faculty of Information Science and

Technology Multimedia University, Malaysia

[2] Wen Chuan Hsieh, Chi Chun Lo, Jing Chi Lee & Li Tsung Huang, The

Implementation of a Proactive Wireless Intrusion Detection System, IEEE 2004.

[3] Joe Scolamiero, Securing Your Wireless Access Point: What Do All Those Settings

Mean Anyways, SANS Institute 2004.

[4] Cisco System Press, edited by Laura Chappell, Introduction to Cisco Router

Configuration, Macmillan Technical Publishing, 1999

**9.2 Book References:**

[1] “Network Security Essentials”,William Stallings ,4th Edition,Pearson Publication

[2] “Software Engineering”, Roger S. Pressman, 6th Edition, McGraw-Hill

Publications

[3] “Computer Networks”,Andrew S.Tanenbaum, 4th Edition,Pearson Publication

[4] “The Unified Modeling Language User Guide”, Grady Booch, James Rambaugh,

Ivar Jacobson, Low Price Edition, Pearson education

**9.3 Web References:**

[1] *http://www.kismetwireless.net/*

[2] *http://www.ilabs.interop.net/WLANSec/What\_is\_wrong\_with\_WEP-lv03.pdf*

[3] *http://www.wifiplanet.com/tutorials/article.php/1492071*

[4] Remote-exploit.org. *http://www.remoteexploit.org/*

**10. APPENDIX**

**10.1 List of Tables:**

**Table No. Name of the Table Page No.**

4.1 Database Table Admin……………………………………………14

4.2 Database Table Info………………………………………………14

4.3 Database Table Nodes……………………………………………14

**10.2 List of Figures:**

**Fig.No. Name of the Figure Page No.**

2.1 Wired Equivalent Privacy (WEP)…………………………………....04

2.2 WEP Key Settings…………………………………………………...05

2.3 RC4 Encryption Algorithm…………………………………………..07

4.4 Use Case Diagram…………………………………………...……….16

4.5 Class Diagram………………………………………………………..17

4.6 Sequence Diagram ………………………….. 18

4.7 Activity Diagram ..19