INFORMATION AND

SECURITY SYSTEM

Topic

<u>ENHANCING AUTHENTICATION IN CHATTING APPLICATION</u>

Submitted by

Sahil Saxena-19BIT0216
Rachuri Ramu-19BIT0158
Devansh Chauhan-19BIT0153

TABLE OF CONTENTS:

ABSTRACT

In most of the organizations, the use of Wireless Local Area Network (WLAN) is increasing rapidly. WLAN security is mainly affected by two internal attacks: Rogue Access Point (RAP) and Rogue Dynamic Host Configuration Protocol (DHCP) Server. Rogue Access Point is installed either by employees or by hackers. Employees install rogue access point for flexibility and scalability. Hackers install rogue access point to breach the security either internally or externally. DHCP server normally does not use authentication when servicing clients that needs dynamic IP address. To secure the organizations from network attacks it is necessary to detect the installation of rogue access point and rogue DHCP server. There are several methods to detect these attacks but still need arise for suggesting a method that provides solution for Address Resolution Protocol (ARP) spoofing. Since spoofing can easily be performed so it will be a major threat in wireless security. In this project we will present a method to prevent this attack. The proposed method uses hash algorithm to assure authenticity and integrity. The proposed solution is feasible and easy to implement.

INTRODUCTION :

LAN is using in almost all networks. Within the Local Area Network, if one host wants to send Ethernet frame to another host, it requires Medium Address Control (MAC) address which specifies the interface to which the frame is destined. Address Resolution Protocol provides mapping from 32 bits Internet Protocol (IP) address to 48 bits MAC address. In this, sender broadcasts ARP REQUEST message containing IP address requesting for associated MAC address. Each host after receiving the request message further broadcasts it to other hosts in the network. The host of that IP address send ARP REPLY message containing MAC address. Each host cache reply message whether they send request message or not. This reduces the traffic, because if any host wants to communicate with the requested host the user does not need to send ARP REQUEST message. The user simply checks the cache table first, if hit is observed then fine otherwise sends request message. Since each host has an entry for IP/MAC address for every other hosts it becomes vulnerable. This leads to ARP spoofing/ARP cache poisoning. In this the attacker can easily change entry, because the whole traffic directed towards the user. ARP spoofing/ARP cache poisoning. In this the attacker can easily change entry, because the whole traffic directed towards the user.

ARP Spoofing based Attacks:

In Local Area Network there are two most familiar attacks:

1. Man-in-the-Middle (MITM)
2. Denial-of-Service (DoS).

These attacks can be launched together or separately with less effort.

1. **Man-in-the-Middle Attack**: Man-in-the-Middle attack also known as Bucket Brigade attack or off and on Janus attack, is an active eavesdropping in which the assailant makes independent connections with the targets and transfers messages between them, making them expect that they are talking straightforwardly to one another in excess of a private association. When attacker spoofs both sender’s and receiver’s IP address then whole communication is under the control of the attacker.
2. **Denial-of-Service Attack:** If the unintended user corrupts the ARP caches of two targets without empowering its IP packet routing then they will not be able to exchange packets. This is called Denial-of-Service attack. In view of this, the two intended parties are denied from exchanging messages with one other or with the Internet. This is done by incorporating wrong entries in ARP caches including non-existent MAC address.

Wireless network is most vulnerable network since everything in this network is broadcast in open media "air". The two major attacks discovered in WLAN are: Rogue Access Point

Rogue DHCP server

Objectives

Internal assaults such as Rogue Access Point (RAP) and Rogue Dynamic Host Configuration Protocol (DHCP) Server have the greatest impact on WLAN security. Rogue Access Points are set up by either employees or hackers. For flexibility and scalability, employees set up a rogue access point. Hackers set up rogue access points both internally and externally to undermine security. When serving clients who require dynamic IP addresses, the DHCP server usually does not use authentication. It is vital to identify the installation of rogue access points and malicious DHCP servers in order to protect companies against network threats. Although there are various methods for detecting these assaults, there is still a need to propose a strategy that gives a solution for ARP spoofing. Because spoofing is so simple to do, it will be a huge danger to wireless security. We will propose a solution to prevent this attack in this project. To ensure authenticity and integrity, the suggested method employs a hash algorithm. The solution provided is feasible and simple to implement.

MOTIVATION

Employees set up a rogue access point for flexibility and scalability. To compromise security, hackers set up rogue access points both within and externally. The DHCP server normally does not utilise authentication when servicing clients who want dynamic IP addresses. In order to defend enterprises from network risks, it is critical to recognise the installation of rogue access points and malicious DHCP servers. Despite the fact that there are a variety of methods for detecting these attacks, there is still a need to present a strategy that addresses ARP spoofing. Because spoofing is so easy to accomplish, it poses a significant threat to wireless security. In this project, we will present a way to prevent this assault.

LITERATURE SURVEY:

Paper	Methodology	Advantages	Dis Advantages	References
1) A Cryptographic Approach for Secure Client - Server Chat Application using Public Key Infrastructure (PKI)	1. authentication procedure has been performed to identify client itself to server. 2. connection and client connects to chat room via **ticket granting ticket (TGT)** request in this step. 3. Messages written by clients are sent to server cryptically through Advanced Encryption Standard (AES) in the third step	**More secure than passwords:** A malicious user must obtain both the private key and the corresponding passphrase to pose as a legitimate user. Provides stronger identity checking through secret private keys. Non-interactive login is possible.	**Speed:** Public key encryption works well and is secure, but it's based on complicated mathematics. **Certification Problems:** if the certification authority gets compromised, the criminal that did it could issue false certificates and fool people into sending data to the wrong place **False Sense of Security:** Public key encryption won't protect against that and, as such, it's only a part of an overall security system.	[1]Karabey and G. Akman, "A cryptographic approach for secure client - server chat application using public key infrastructure (PKI)," *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST),* Barcelona, 2016, pp. 442-446. doi: 10.1109/ICITST.2016.7856750 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7856750&isnumber=7856645

2) Implementation of Elliptic Curve Digital Signature Algorithm Using SHA-512	It has been modified by an attacker, the ECDSA detects it and signals to the transmitter for retransmission. The SHA-512 algorithm is the hash algorithm used in the ECDSA and is implemented for an 8-bit architecture.	ECC employs a relatively short encryption key -- a value that must be fed into the encryption algorithm to decode an encrypted message. This short key is faster and requires less computing power than other firstgeneration encryption public key algorithms	The ECC algorithm is more complex and more difficult to implement than RSA, which increases the likelihood of implementation errors, thereby reducing the security of the algorithm. Public key cryptography is computationally more expensive than private key encryption, which employs a single, shared encryption key	S. E. Mathe, L. Boppana and R. K. Kodali, "Implementation of Elliptic Curve Digital Signature Algorithm using SHA-512," *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, Pune, 2015, pp. 445-449. doi: 10.1109/IIC.2015.7150783 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7150783&isnumber=7150576

3) Brief review	The secured hash functions	User- secure . Safe from		S. Debnath, A.

on journey of secured hash algorithms	are applied in various fields to provide a secure data transfer and authentication of messages and other user linked information through a series of algorithms. From the establishment of the first hash function MD5, followed by SHA 1, this data encryption system has undergone several upgradations and advanced to SHA 2 and SHA 3.	attack for probably a few decades. Speed- Fastest cryptographic hash function	Slowest hash function-For a system with high transaction rate, these hash functions can take a significant toll on the CPU. Lack of security- MD5 can be broken relatively easily and is no longer suitable for use in secure systems..	Chattopadhyay and S. Dutta, "Brief review on journey of secured hash algorithms," *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, Kolkata, 2017, pp. 1-5. doi: 10.1109/OPTRONIX.2017.8349971 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8349971&isnumber=8349658

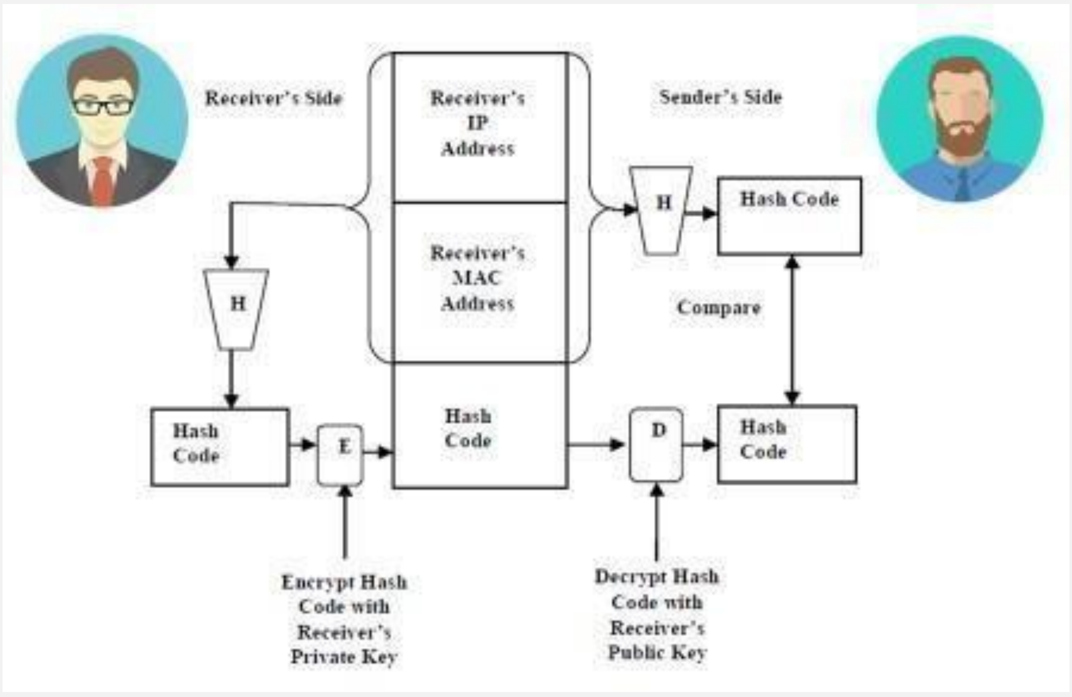
4)An application for end to end secure messaging service on Android supported device	encrypted message only can be decrypted by user in time of accessing the conversation by introduction of the encrypted user's define key.	As it is implemented in both hardware and software, it is most robust security protocol. It uses higher length key sizes such as 128, 192 and 256 bits for encryption. It is one of the most spread commercial and open source	1)It uses too simple algebraic structure. 2)Every block is always encrypted in the same way. 3)Hard to implement with software.	[1]S. Nayak *et al*., "An application for end to end secure messaging service on Android supported device," *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, 2017, pp. 290-294. doi: 10.1109/IEMCON.2017.8117222 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8117222&isnumber=8117121
5) ARP Spoofing Detection Algorithm Using ICMP Protocol	ICMP protocol, packet detection modules, spoofing detection modules		Packet injection is fairly minimal and the performance of the network will not be influenced.	[1] Jinhua, G., & Kejian,X.(2013). AR P spoofing detection algorithm using ICMP protocol. In *International Conference on Computer Communication and Informatics 2013*.
6) A Fault Attack on a Hardware- based Implementation of the Secure Hash Algorithm SHA- 512	end-of-rounds error and a last-block error		In the case of detection the secure implementation does not output the faulty values needed by the attacker to complete the attack.	1. Shoufan, A. (2013, December). A fault attack on a hardware-based implementation of the secure hash algorithm SHA-512. In *Reconfigurable Computing and FPGAs*

				(ReConFig), 2013 International Conference on (pp. 1-7). IEEE.
7) A Security Framework against ARP Spoofing	Binding logical address to physical address		Performance and cost	Saini, R. R., & Gupta, H. (2015, September). A security framework against ARP spoofing. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on* (pp. 1-6). IEEE.
8)Prevention of ARP Spoofing: A Probe Packet Based Technique	Probe packet based technique		SDE doesn't sand for strong attacker	Pandey,P.(2013,February). Prevention of ARP spoofing: A probe packet based technique. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International* (pp. 147-153). IEEE.
9)Defense Technique against Spoofing Attacks using Reliable ARP Table in Cloud Computing Environment.	Keystone authentication service by Open stack		Partial loss of resources	Kang, H. S., Son, J. H., & Hong, C. S. (2015, August). Defense technique against spoofing attacks using reliable arp table in cloud computing environment. In *Network Operations and Management Symposium (APNOMS), 2015 17th*

				AsiaPacific (pp. 592-595).IEEE.
10) Optimising the SHA-512 cryptographic hash function on FPGAs	Pipelining optimization techniques		Applicable for circuit-level only	1.Athanasiou, G. S., Michail, H. E., Theodoridis, G., & Goutis, C. E. (2013). Optimising the SHA-512 cryptographic hash function on FPGAs. IET Computers & Digital Techniques, 8(2), 70-82.
11) Detection of ARP Spoofing : A Command Line Execution Method	1.Network security enhancement 2.Works with dynamic IP Addresses 3.No network congestion 4. Light weight software any.		DOS attack	1.Sharma, D., Khan, O., & Manchanda, N. (2014, March). Detection of ARP Spoofing: A command line execution method. In Computing for Sustainable Global Development (INDIACom), 2014 International Conference on (pp. 861-864). IEEE.
12) ARP Spoofing Based Access Control for DLNA Devices	Access control methods of DLNA		Treats device as a Single access unit.	Wu, Y., & Zhi, X. (2015, August). ARP Spoofing Based Access Control for DLNA Devices. In Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015 IEEE 12th Intl Conf on(pp. 13711376). IEEE.

13) Design of WinPcap Based ARP Spoofing Defense System	ARP spoofing detection and recovery method		Suitable only for WindowsXP	Yang, M., Wang, Y., & Ding, H. (2014, September). Design of Win Pcap Based ARP Spoofing Defense System. In Instrumentation and Measurement, Computer, Communication and Control (IMCCC), 2014 Fourth International Conference on (pp. 221-225). IEEE.
14) A Novel Web Content Spoofing Technique on WLAN and Its Countermeasures	1.Signature-Based detection 2.Anomaly-Based detection		Counter measures are not sufficient.	Jitpukdebodin, S., Chokngamwong, R., & Kungpisdan, S. (2014, September). A novel web content spoofing technique on WLAN and its countermeasures. In Communications and Information Technologies (ISCIT), 2014 14th International Symposium on (pp. 254-258). IEEE.

ARCHITECTURE DESIGN:



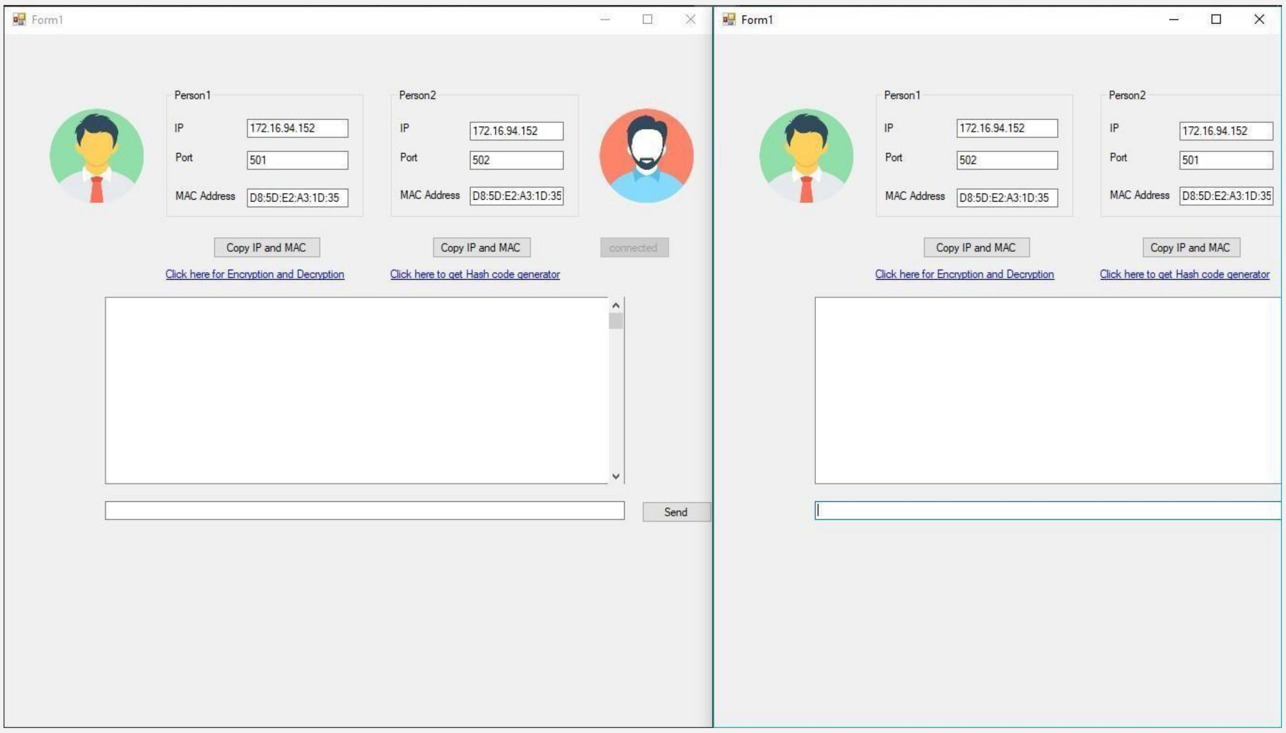
Technique Used:

In this technique, Receiver takes its own IP and MAC locations and after that utilizations secure hash work SHA-512, to deliver a hash esteem. In this manner scrambles the hash code utilizing its key, making a computerized signature. Receiver sends the ARP REPLY message alongside the mark. Sender create the hash code utilizing collector's IP and MAC addresses and unscrambles the mark utilizing beneficiary's critical. In the event that both produced hash codes are same, it implies no modification has been finished amid transmission.

This guarantees authenticity and integrity.

SCREENSHOTS:

Establishing Connection between sender and receiver.

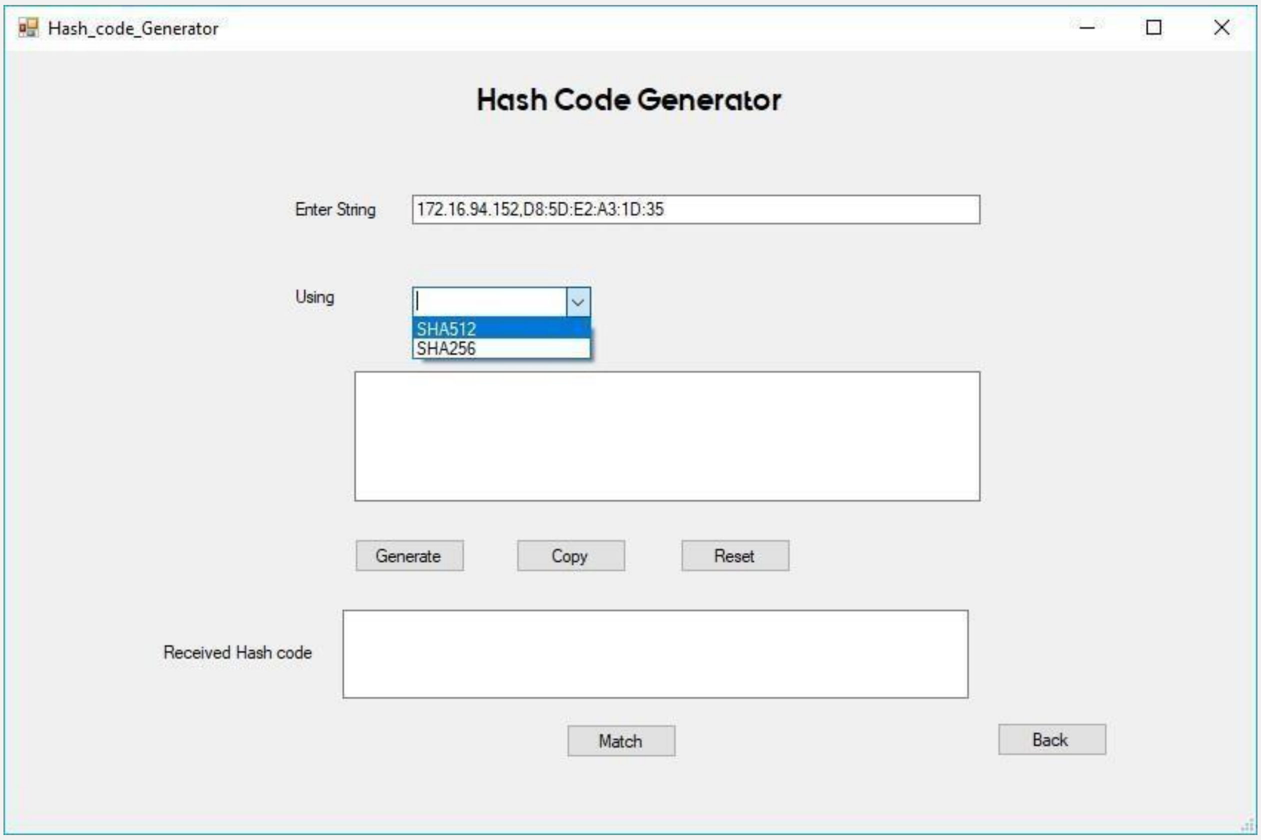

Form1

Person1		Person2	
IP	172.16.94.152	IP	172.16.94.152
Port	501	Port	502
MAC Address	D8:5D:E2:A3:1D:35	MAC Address	D8:5D:E2:A3:1D:35

Form1

Person1		Person2	
IP	172.16.94.152	IP	172.16.94.152
Port	502	Port	501
MAC Address	D8:5D:E2:A3:1D:35	MAC Address	D8:5D:E2:A3:1D:35

Copy IP and MAC
Copy IP and MAC
connected
Click here for Encryption and Decryption
Click here to get Hash code generator
Send

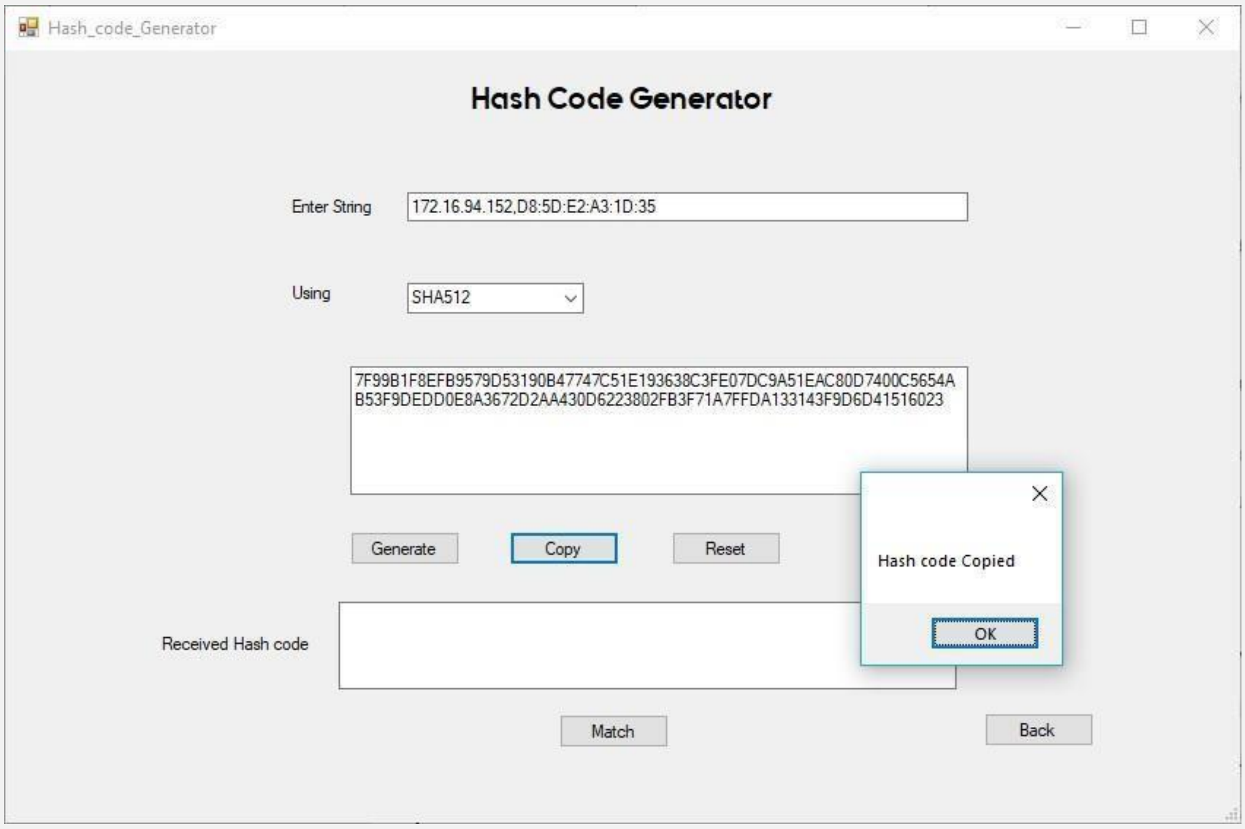

Hash_code_Generator

Hash Code Generator

Enter String: 172.16.94.152,D8:5D:E2:A3:1D:35

Using: SHA512, SHA256

Generating and Copying Hash Code.


Hash_code_Generator

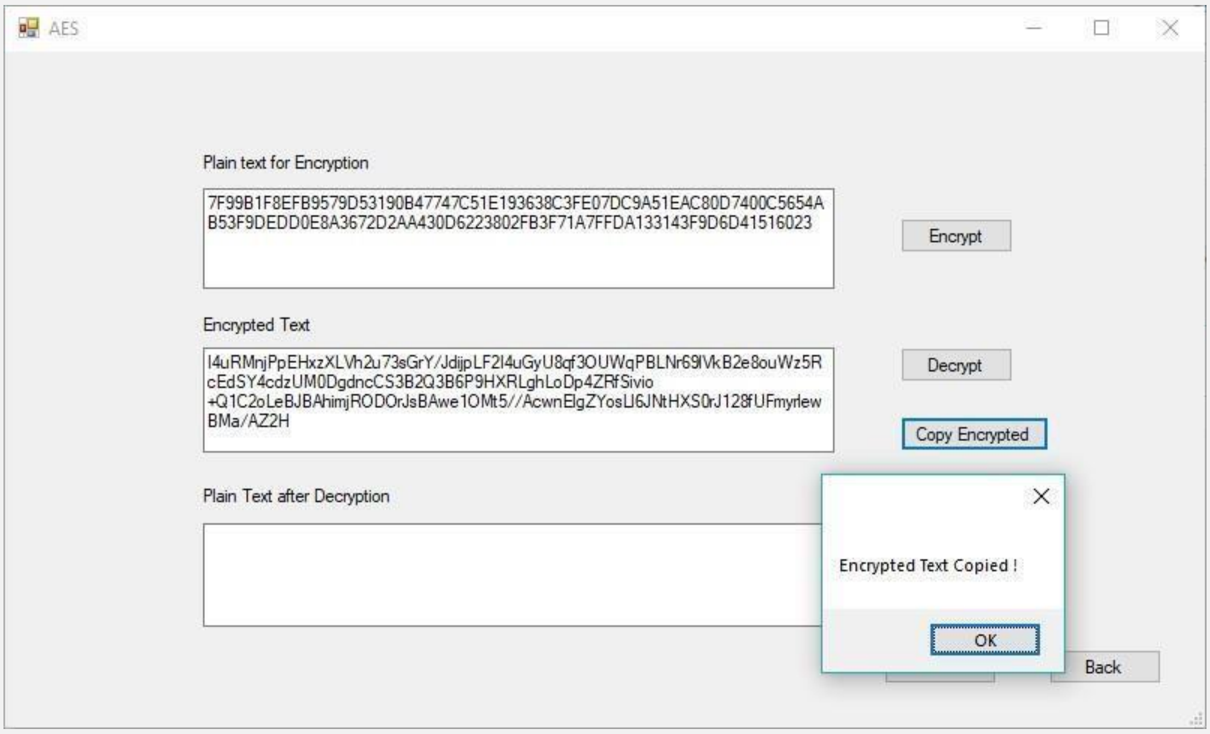
Hash Code Generator

Enter String: 172.16.94.152,D8:5D:E2:A3:1D:35

Using: SHA512

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC80D7400C5654A
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9D6D41516023
```

Hash code Copied



Plain text for Encryption

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC80D7400C5654A
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9D6D41516023
```

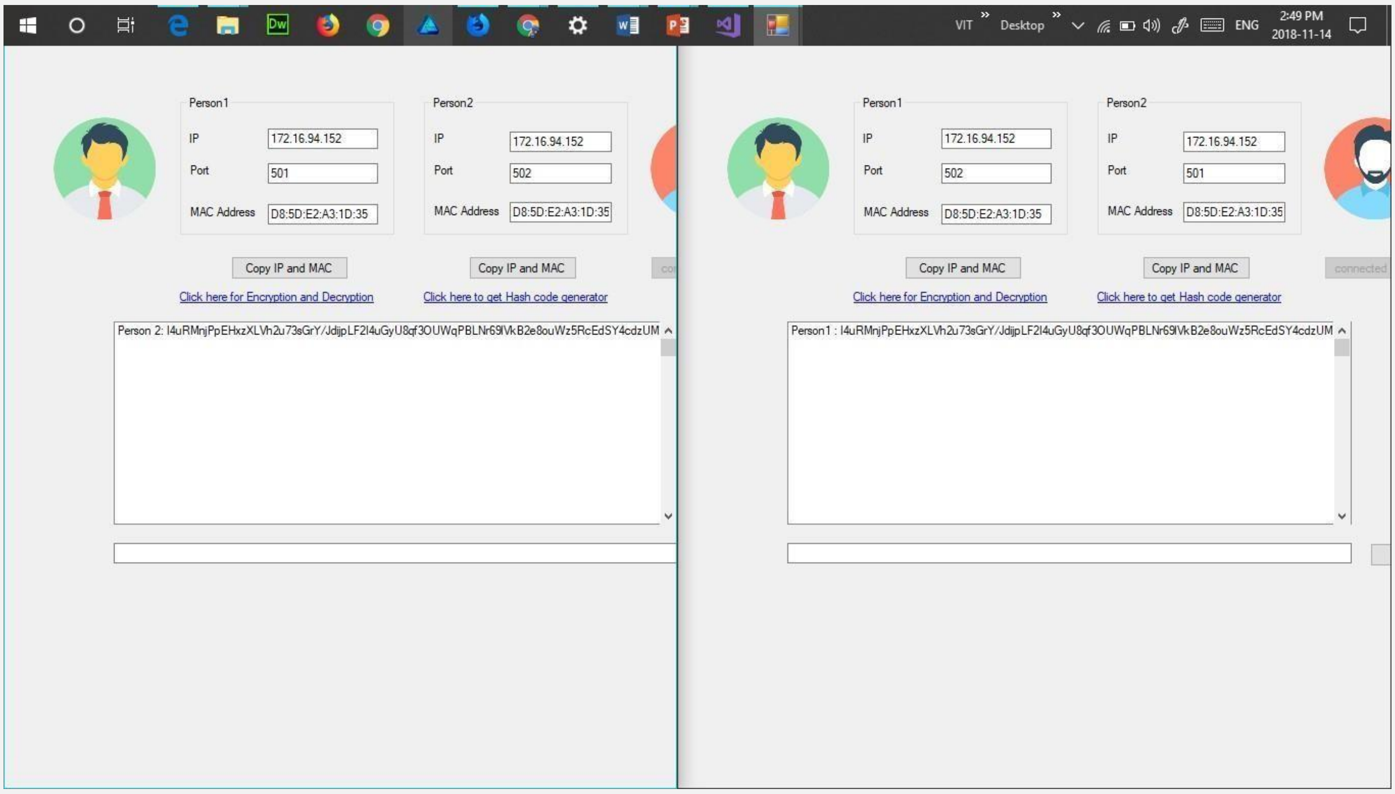
Encrypted Text

```
I4uRMnjPpEHxzXLVh2u73sGrY/JdijpLF2I4uGyU8qf3OUWqPBLNr69lVkB2e8ouWz5R
cEdSY4cdzUM0DgdncCS3B2Q3B6P9HXRLghLoDp4ZRfSivio
+Q1C2oLeBJBAhimjRODOrJsBAwe1OMt5//AcwnElgZYosLl6JNtHXS0rJ128fUFmyrlew
BMa/AZ2H
```

Plain Text after Decryption

Encrypted Text Copied !

Sending Encrypted hash code through message.


VIT
Desktop
ENG
2:49 PM
2018-11-14
Copy IP and MAC
Copy IP and MAC
Click here for Encryption and Decryption
Click here to get Hash code generator
Copy IP and MAC
Copy IP and MAC
connected
Click here for Encryption and Decryption
Click here to get Hash code generator

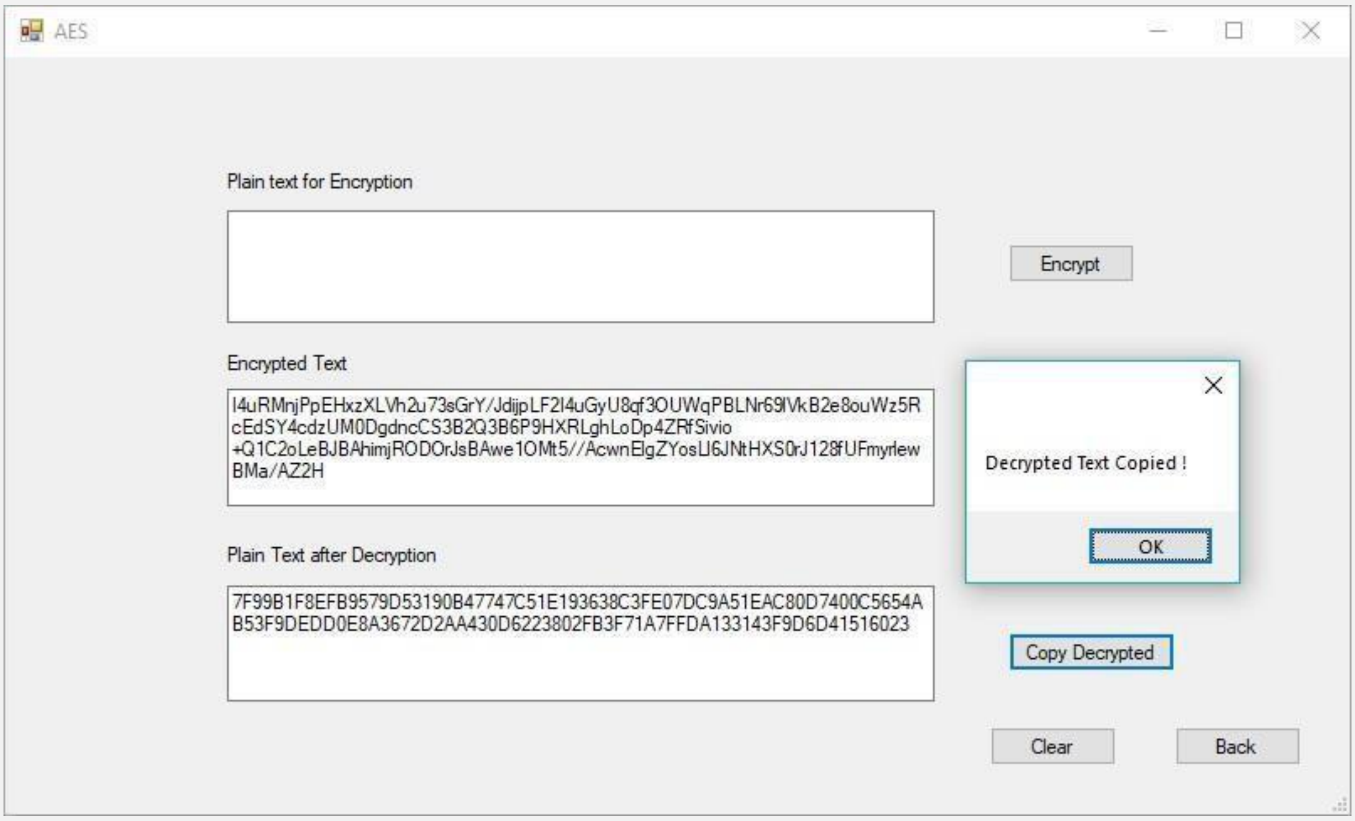
Person1		Person2	
IP	172.16.94.152	IP	172.16.94.152
Port	501	Port	502
MAC Address	D8:5D:E2:A3:1D:35	MAC Address	D8:5D:E2:A3:1D:35

```
Person 2: I4uRMnjPpEHxzXLVh2u73sGrY/JdijpLF2I4uGyU8qf3OUWqPBLNr69lVkB2e8ouWz5RcEdSY4cdzUM
```

Person1		Person2	
IP	172.16.94.152	IP	172.16.94.152
Port	502	Port	501
MAC Address	D8:5D:E2:A3:1D:35	MAC Address	D8:5D:E2:A3:1D:35

```
Person1 : I4uRMnjPpEHxzXLVh2u73sGrY/JdijpLF2I4uGyU8qf3OUWqPBLNr69lVkB2e8ouWz5RcEdSY4cdzUM
```

Copying Decrypted Hash Code:



Plain text for Encryption

Encrypted Text

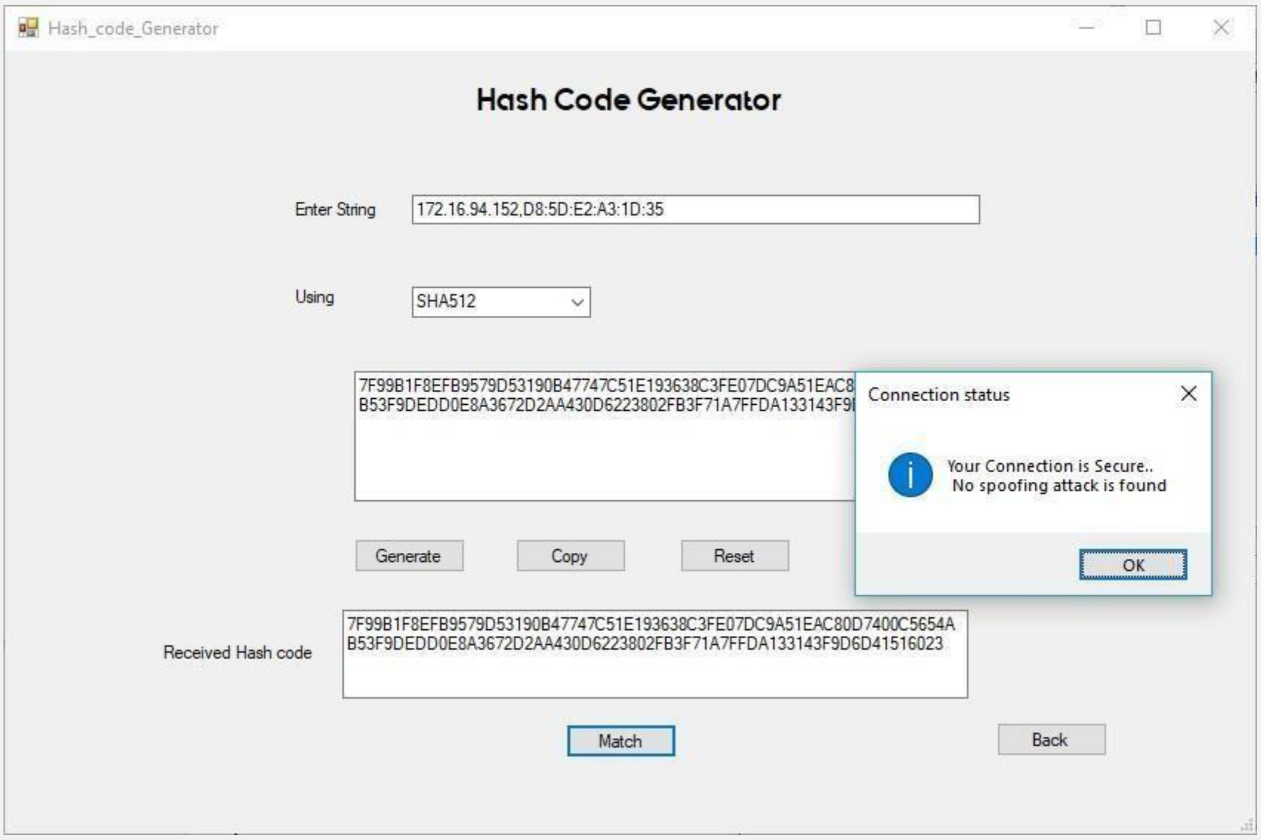
```
l4uRMnjPpEHxzXLVh2u73sGrY/JdijpLF2l4uGyU8qf3OUWqPBLNr69lVkB2e8ouWz5R
cEdSY4cdzUM0DgdncCS3B2Q3B6P9HXRLghLoDp4ZRfSivio
+Q1C2oLeBJBAhimjRODOrJsBAwe1OMt5//AcwnElgZYosLl6JNtHXS0rJ128fUFmyrlew
BMa/AZ2H
```

Plain Text after Decryption

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC80D7400C5654A
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9D6D41516023
```

Decrypted Text Copied !

If both hash code are same then a message will be displayed that "Your Connection is Secure".



Hash Code Generator

Enter String: 172.16.94.152,D8:5D:E2:A3:1D:35

Using: SHA512

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC8
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9I
```

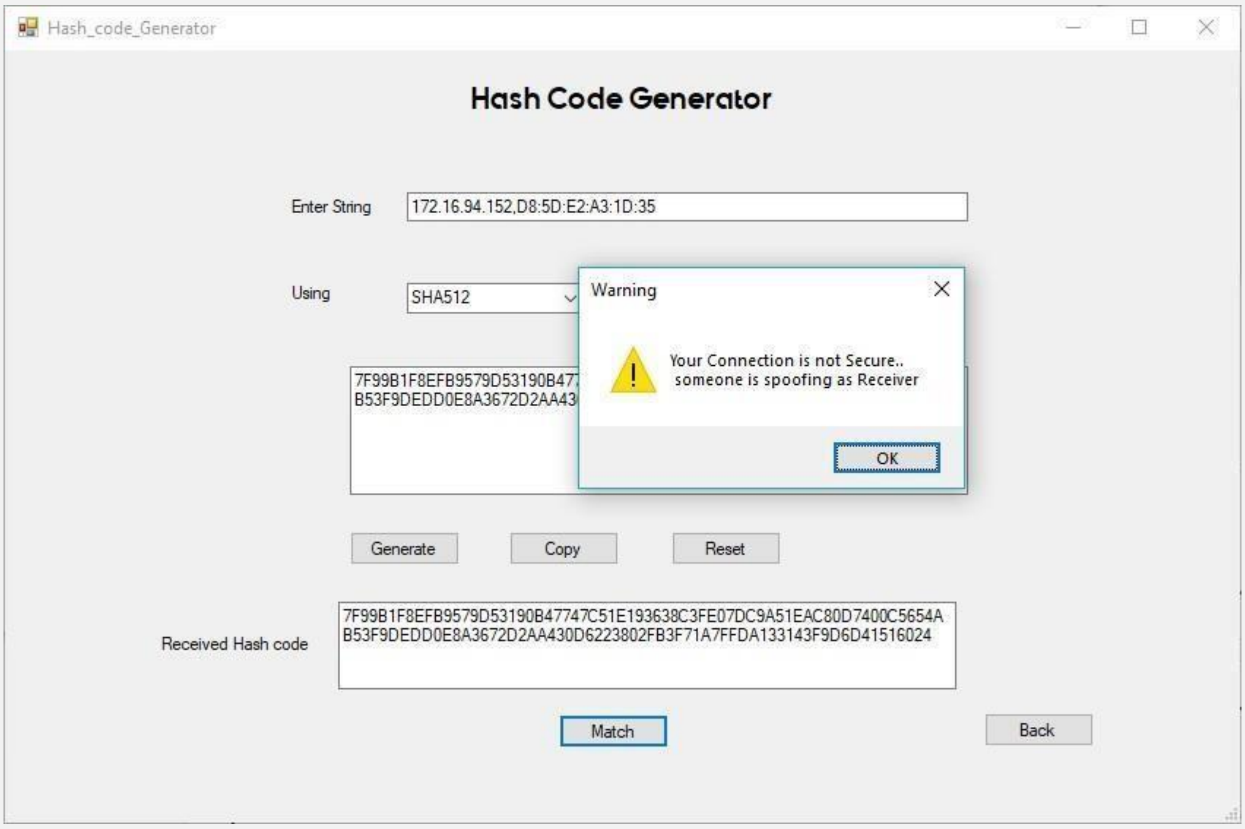
Connection status

Your Connection is Secure..
No spoofing attack is found

Received Hash code

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC80D7400C5654A
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9D6D41516023
```

If there are any changes in the hash code received and generated then it will display "Your Connection is Not Secure".



Enter String 172.16.94.152,D8:5D:E2:A3:1D:35

Using SHA512

```
7F99B1F8EFB9579D53190B47
B53F9DEDD0E8A3672D2AA43
```

Warning

Your Connection is not Secure..
someone is spoofing as Receiver

Received Hash code

```
7F99B1F8EFB9579D53190B47747C51E193638C3FE07DC9A51EAC80D7400C5654A
B53F9DEDD0E8A3672D2AA430D6223802FB3F71A7FFDA133143F9D6D41516024
```

CONCLUSION:

Authentication plays a major role for communication to maintain the Confidential information in a secured manner. Due to some attacks or spoofing the authentication fails and confidential information may get leaked. ARP spoofing/poisoning is very common attack in WLAN even after preventing rogue access point and rough DHCP server. We presented a feasible method to detect ARP spoofing. The reason behind ARP spoofing is the lack of authenticity. By our system we can provide high security using SHA-512 for hash code and AES for encryption. The spoofing can be detected at early stages or beginning of the communication so that the user may protect confidential information without sharing it to a spoofing hacker.

APPENDIX-1:

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.Threading.Tasks; using
System.Windows.Forms;

using System.Net; using
System.Net.Sockets;
using System.Net.NetworkInformation;

namespace chatting_app
{
    public partial class Form1 : Form
    {        public static Form1 staticVar = null;        public
static Form1 staticVar2 = null;
        Socket sck;
        EndPoint eplocal, epremote;        public Form1()
        {
            InitializeComponent();
            sck = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
            sck.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReuseAddress, true);
             txtperson1_ip.Text = GetLocalIP();            txtperson2_ip.Text =
GetLocalIP();            txtMac1.Text = GetMACAddress().ToString();
txtMac2.Text = GetMACAddress().ToString();
        }        private string GetLocalIP()
        {
            IPHostEntry host;
            host = Dns.GetHostEntry(Dns.GetHostName());

            foreach(IPAddress ip in host.AddressList)
            {
                if(ip.AddressFamily == AddressFamily.InterNetwork)
                {
                    return ip.ToString();
                }
            }
            return "127.0.0.1";
        }

        private static string GetMACAddress()
        {            foreach (NetworkInterface nic in
NetworkInterface.GetAllNetworkInterfaces())
            {
                if (nic.OperationalStatus == OperationalStatus.Up)                    return
AddressBytesToString(nic.GetPhysicalAddress().GetAddressBytes());
            }

            return string.Empty;
        }
        private static string AddressBytesToString(byte[] addressBytes)
        {
            return string.Join(":", (from b in addressBytes
                                    select b.ToString("X2")).ToArray());
        }
        private void btnStart_Click(object sender, EventArgs e)
        {            try
{
                eplocal = new IPEndPoint(IPAddress.Parse(txtperson1_ip.Text),
Convert.ToInt32(txtperson1_port.Text));                sck.Bind(eplocal);

                epremote = new IPEndPoint(IPAddress.Parse(txtperson2_ip.Text),
Convert.ToInt32(txtperson2_port.Text));                sck.Connect(epremote);

                byte[] buffer = new byte[1500];
                sck.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epremote, new
AsyncCallback(MessageCallBack), buffer);
                 btnStart.Text = "connected";
btnStart.Enabled = false;                btnSend.Enabled = true;
message_send.Focus();

            }
            catch(Exception exe)
            {
                MessageBox.Show(exe.ToString());
            }
        }
        private void btnSend_Click(object sender, EventArgs e)
        {            try
            {
                System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();                byte[] msg =
new byte[1500];                msg = enc.GetBytes(message_send.Text);

                sck.Send(msg);
                messagelist.Items.Add("Person 2: " + message_send.Text);                message_send.Clear();
            }
            catch(Exception exe)
            {
                MessageBox.Show(exe.ToString());
            }
        }


        private void linklblencryptdecrypt_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {            staticVar2 = this;
this.Hide();            AES aes = new AES();
aes.ShowDialog();
        }
        private void linklblcrypto_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {            staticVar = this;
this.Hide();
            Hash_code_Generator gen = new Hash_code_Generator();            gen.ShowDialog();
        }
        private void btncopyIpMac1_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(txtperson1_ip.Text + "," + txtMac1.Text);
        }
        private void btncopyIpMac2_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(txtperson2_ip.Text + "," + txtMac2.Text);
        }
        private void Form1_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                btnSend.PerformClick();
            }
        }
        private void MessageCallBack(IAsyncResult aResult)
        {           try
{
                int size = sck.EndReceiveFrom(aResult, ref epremote);                 if(size > 0)
                {
                    byte[] receivedData = new byte[1464];                    receivedData =
(byte[])aResult.AsyncState;

                    ASCIIEncoding eEncoding = new ASCIIEncoding();
                    string receivedMessage = eEncoding.GetString(receivedData);
                     messagelist.Items.Add("Person1 : " + receivedMessage);
                }
                byte[] buffer = new byte[1500];
                sck.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epremote, new
AsyncCallback(MessageCallBack), buffer);
            }
            catch(Exception exp)
            {
                MessageBox.Show(exp.ToString());
            }
        }
    } }
```

Hash_code_generator.cs

```
using System;
using System.Collections.Generic; using
System.ComponentModel; using System.Data;
using System.Drawing; using System.Linq;
using System.Text; using
System.Threading.Tasks; using
System.Windows.Forms; using
System.Security.Cryptography;

namespace chatting_app
{
    public partial class Hash_code_Generator : Form
    {
        public Hash_code_Generator()
        {
            InitializeComponent();
        }
        public static string GenerateSHA256String(string inputString)
        {
            SHA256 sha256 = SHA256Managed.Create();             byte[] bytes =
Encoding.UTF8.GetBytes(inputString);             byte[] hash =
sha256.ComputeHash(bytes);             return GetStringFromHash(hash);
        }
        public static string GenerateSHA512String(string inputString)         {
            SHA512 sha512 = SHA512Managed.Create();             byte[] bytes =
Encoding.UTF8.GetBytes(inputString);             byte[] hash =
sha512.ComputeHash(bytes);             return GetStringFromHash(hash);
        }
        private static string GetStringFromHash(byte[] hash)
        {
            StringBuilder result = new StringBuilder();             for (int i = 0; i <
hash.Length; i++)
            {
                result.Append(hash[i].ToString("X2"));
            }
            return result.ToString();
        }         private void BtnGenerate_Click(object sender, EventArgs e)
        {
            string str = txtString.Text;
            if (cbxusing.SelectedIndex == 0)
            {
                txthash.Text = GenerateSHA512String(str);
            }
            else if (cbxusing.SelectedIndex == 1)
            {
                txthash.Text = GenerateSHA256String(str);
            }            else
            {
                MessageBox.Show("Invalid selection... Please choose Correct option");
            }
        }
        private void BtnReset_Click(object sender, EventArgs e)
        {            txtString.Text = "";
txthash.Text = "";
        }
        private void BtnCopy_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(txthash.Text);
            MessageBox.Show("Hash code Copied");

        }
        private void Btnmatch_Click(object sender, EventArgs e)
        {
            if (txthash.Text == txtreceivedhash.Text)
            {
                MessageBox.Show("Your Connection is Secure..\n No spoofing attack is found", "Connection status",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }            else
            {
                MessageBox.Show("Your Connection is not Secure..\n someone is spoofing as Receiver", "Warning",
MessageBoxButtons.OK, MessageBoxIcon.Warning);


            }
        }
        private void btnback_Click(object sender, EventArgs e)
        {

            Form1.staticVar.Show();

        }

    }
}
```

AES.cs

```
using System;
using System.Collections.Generic; using
System.ComponentModel; using
System.Data; using System.Drawing;
using System.Linq; using System.Text;
using System.Threading.Tasks; using
```

```
System.Windows.Forms; using
AesEncDec;

namespace chatting_app
{
   public partial class AES : Form
   {
       public AES()
       {
           InitializeComponent();
       }
       private void BtnEncrypt_Click(object sender, EventArgs e)
       {           string plaintext = txtplainforencrypt.Text;
txtencrypt.Text = AesCryp.Encrypt(plaintext);
       }
       private void BtnDecrypt_Click(object sender, EventArgs e)
       {
           string encrypted = txtencrypt.Text;
           txtdecryptedplain.Text = AesCryp.Decrypt(encrypted);
       }
       private void btnClear_Click(object sender, EventArgs e)
       {           txtdecryptedplain.Text = "";
txtencrypt.Text = "";            txtplainforencrypt.Text =
"";
       }
       private void btnback_Click(object sender, EventArgs e)
       {
           Form1.staticVar2.Show();
       }
       private void btnCopyencryptedtext_Click(object sender, EventArgs e)        {
           Clipboard.SetText(txtencrypt.Text);
           MessageBox.Show("Encrypted Text Copied !");
       }        private void btnCopyDecryptedtext_Click(object sender, EventArgs e)        {
           Clipboard.SetText(txtdecryptedplain.Text);
           MessageBox.Show("Decrypted Text Copied !");
       }
    } }
```

APPENDIX-2:

ADVANTAGES:

- Since we are using SHA-512 it is not easy to perform any type of attacks.
- Authenticity and integrity are achieved
- GUI

DISADVANTAGES:

- If hacker knows that the used encryption algorithm then confidentiality will be violated.

- We used symmetric key for encryption and decryption, so if any one share their keys then it will be attacked by an attacker.