INFORMATION SECURITY

Practical File

Course Code : INITC18

Submitted By –

Name: Rohit Kumar

Roll no.: 2020UIN3322

Branch: ITNS

Semester: 6th

Academic Year: 2022-23

INDEX

SNO.	NAME	DATE	SIGNATURE
1	Study of the features of firewall in providing network security and to set Firewall security in windows. Students should know the following: a) Know how to setup and configure a firewall on Operating System. b) Know about the Windows Firewall with Advanced Security. c) Know the Connection Security Rules d) Know How to Start & Use the Windows Firewall with Advanced Security		
2	Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures Using tool GnuPG. (Download GPG4Win Tool). Create your public and private keys using Kloepatra Certificate management software. Check encryption and decryption.		
3	Implement MD5 Algorithm in Python/C++/Java.		
4	Implement SHA-256 Algorithm. Consider all the Constants and Tables as given in the textbook.		
5	Plot an elliptic curve over a finite field. Check whether the points lie on the elliptic curve or not.		
6	Perform the Elliptic Curve operations like Addition and Multiplication of two points and find the Inverse of a point also.		
7	Implement RSA Digital Signature Scheme.		
8	Implement Elgamal Digital Signature Scheme.		
9	Implement Schnorr Digital Signature Scheme.		
10	Using Jcrypt tool (or any other equivalent) to demonstrate asymmetric, symmetric crypto algorithm.		
11	Implement the Identity-based Encryption (IBE). Use the email address of the recipient to generate the key for a destination.		
12	To study and work with KF SENSOR Intrusion Detection Tool. Setup a honeypot and monitor the honeypot on the network.		
13.	Configure Wireshark with a key to let you look inside encrypted SSL messages. You can read on the web how to do this. Once decrypted, you will be able to observe the HTTP protocol running on top of SSL, as well as the details of other SSL messages such as Alerts.		
14.	To build a Trojan and know the harmness of the trojan malwares in a computer system. When the trojan code executes, it will open MS-Paint, Notepad, Command Prompt,		

	Explorer, calculator, infinitely. Note: Use Vmware to perform this experiment		
15.	Implement a code to simulate buffer overflow attack.		
16.	Use the Nessus tool to scan the network for vulnerabilities.		

Practical-1

Study of the features of firewall in providing network security and to set Firewall security in windows. Students should know the following:

a) Know how to setup and configure a firewall on Operating System.

b) Know about the Windows Firewall with Advanced Security.

c) Know the Connection Security Rules

d) Know How to Start & Use the Windows Firewall with Advanced Security

The features of a firewall in providing network security are:

- Various protection levels
- Wireless network protection
- Internet and network access
- Blockage against unauthorized access
- Protection against malware

a) The following are the steps to set firewall security in windows:

Select the start button > Settings > Update and Security > Windows Security and then Firewall & network protection.

Select a network profile: Domain network, Private network or Public network

Under Microsoft Defender Firewall, switch the setting to On. If your device is connected to a network, network policy settings might prevent you from completing these steps. For more info, contact your administrator.

b) Windows Defender Firewall in Windows 8, Windows 7, Windows Vista, Windows Server 2012, Windows Server 2008, and Windows Server 2008 R2 is a stateful host firewall that helps secure the device by allowing you to create rules that determine which network traffic is permitted to enter the device from the network and which network traffic the device is allowed to send to the network. Windows Defender Firewall also supports Internet Protocol security (IPsec), which you can use to require authentication from any device that is attempting to communicate with your device. When authentication is required, devices that cannot be authenticated as a trusted device cannot communicate with your device. You can also use IPsec to require that certain network traffic is encrypted to prevent it from being read by network packet analyzers that could be attached to the network by a malicious user.

The Windows Defender Firewall with Advanced Security MMC snap-in is more flexible and provides much more functionality than the consumer-friendly Windows Defender Firewall interface found in the Control Panel. Both interfaces interact with the same underlying services but provide different levels of control over those services. While the Windows Defender Firewall Control Panel program can protect a single device in a home environment, it does not provide enough centralized management or security features to help secure more complex network traffic found in a typical business enterprise environment.

c) Connection security rules specify how and when Windows Firewall with Advanced Security uses IPsec to protect traffic passing between the local computer and other computers on the network. Connection security rules force two peer computers to authenticate before a connection can be established between them. Connection security rules can also ensure that communications between the computers is secure by encrypting all traffic passed between them

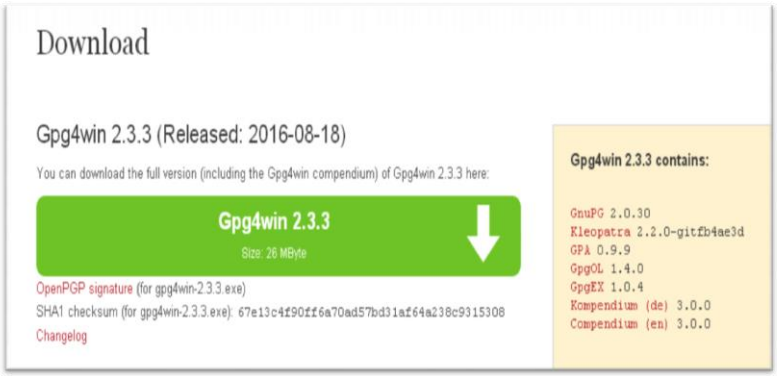
d) To access the Windows Defender Firewall with Advanced Security, the easiest method to open Windows Defender Firewall with Advanced Security in all three Windows versions is to search for its executable file. Type "wf.msc" in the Windows search box and click or tap on the result. In Control Panel you can access the Windows Defender Firewall with Advanced Security by going to "System and Security -> Windows Defender Firewall," and then by clicking or tapping Advanced settings. In Windows 10, you can find a shortcut for Windows Defender Firewall with Advanced Security in the Start Menu using the following path: "Start Menu → Windows Administrative Tools → Windows Defender Firewall with Advanced Security."

Practical-2

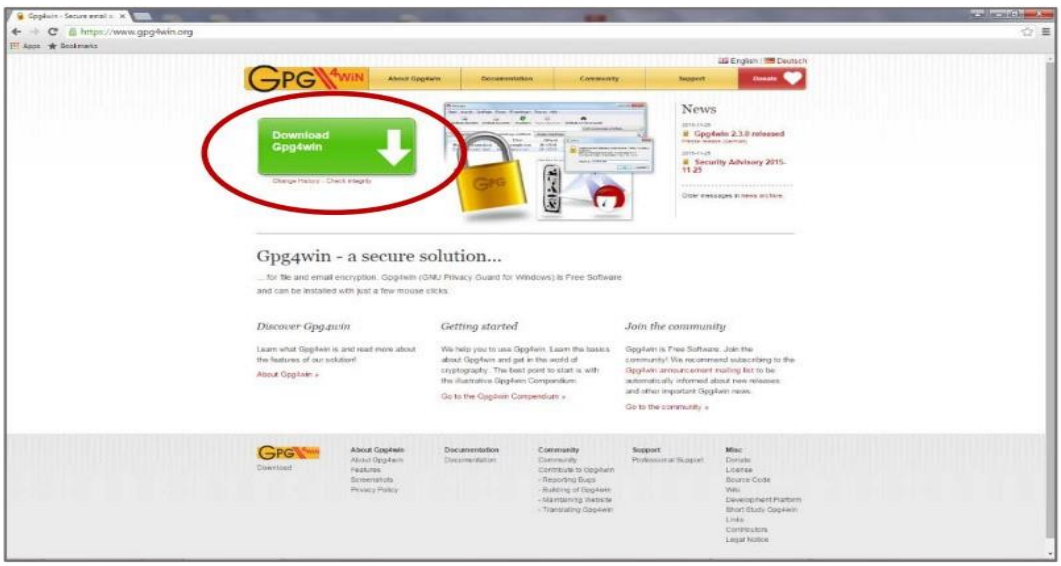
Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures Using tool GnuPG. (Download GPG4Win Tool). Create your public and private keys using Kloepatra Certificate management software. Check encryption and decryption.

INSTALLING THE SOFTWARE:

1. Visit www.gpg4win.org. Click on the "Gpg4win 2.3.0" button

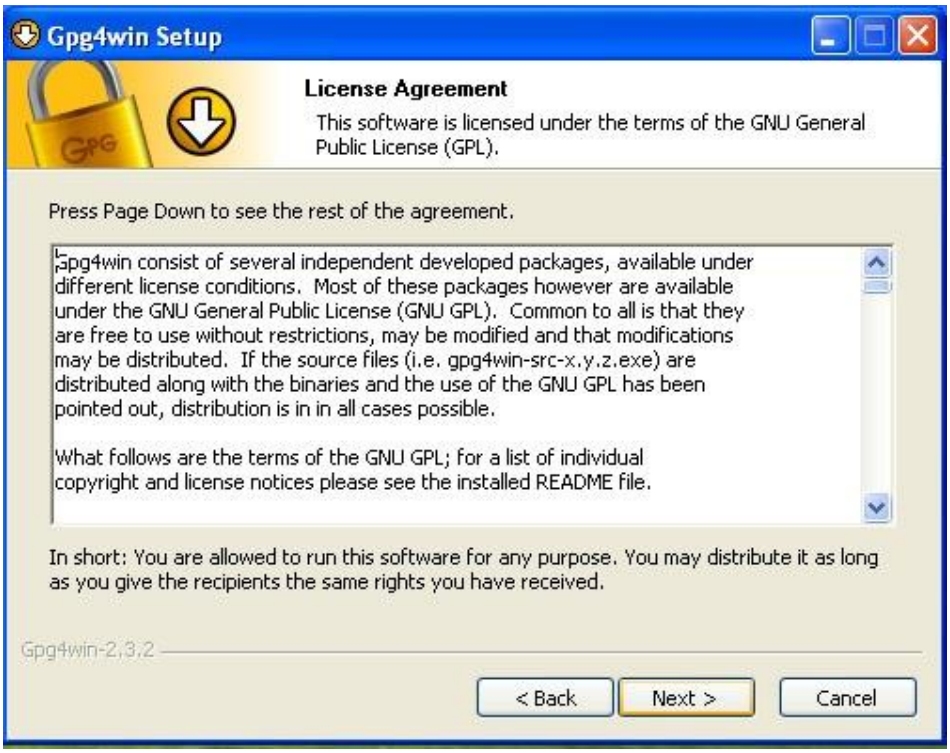


2. On the following screen, click the "Download Gpg4win" button.

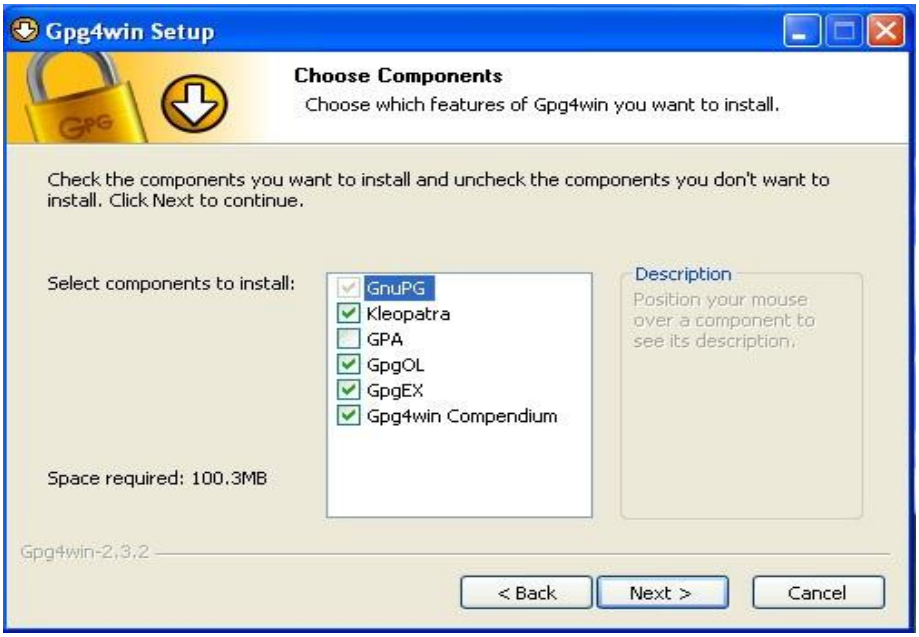


3. When the "Welcome" screen is displayed, click the "Next" button

4. When the “License Agreement” page is displayed, click the “Next” button



5. Set the check box values as specified below, then click the “Next” button

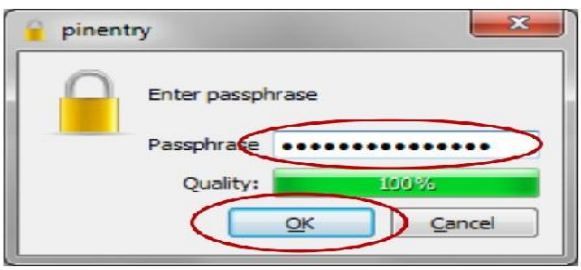


6. Set the location where you want the software to be installed. The default location is fine. Then, click the “Next” button.

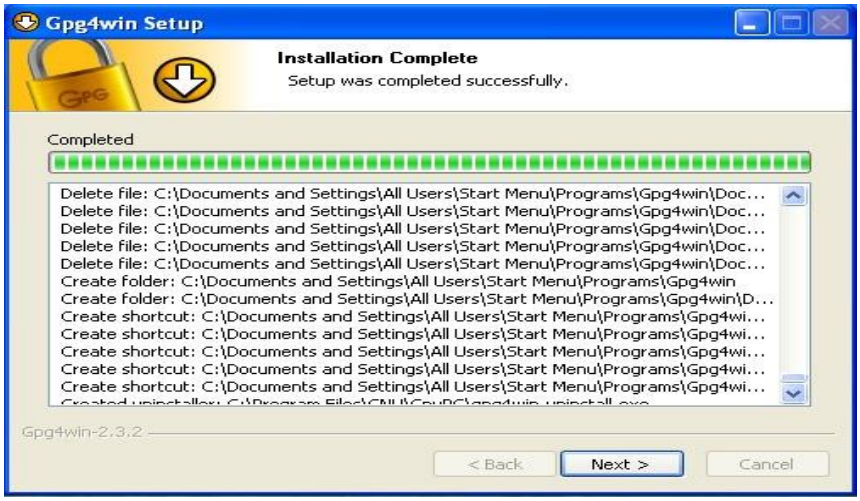
7. Specify where you want shortcuts to the software placed, then click the “Next” button.

8. If you selected to have a GPG shortcut in your Start Menu, specify the folder in which it will be placed. The default “Gpg4win” is OK. Click the “Install” button to continue

9. A warning will be displayed if you have Outlook or Explorer opened. If this occurs, click the “OK” button.

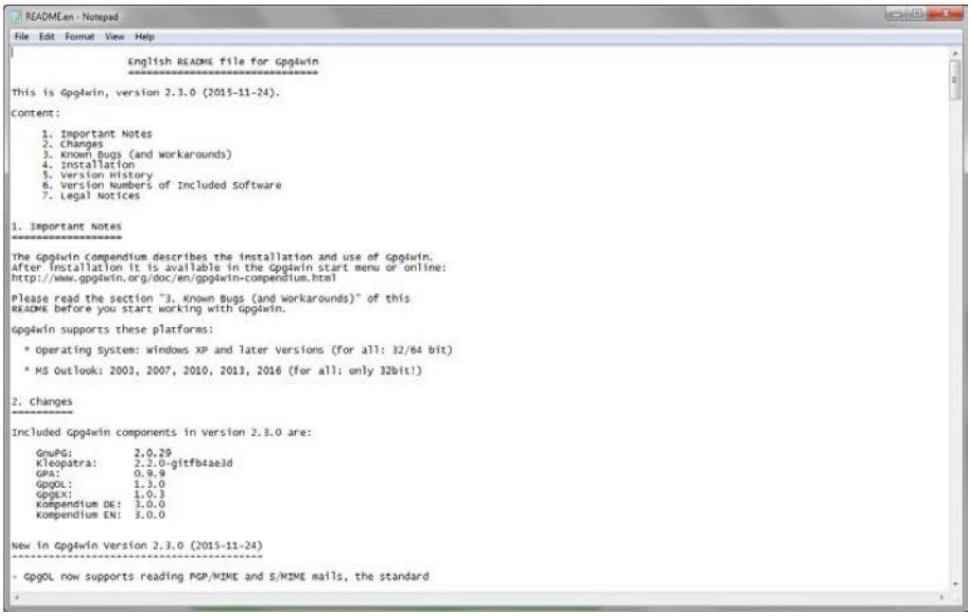


10. The installation process will tell you when it is complete. Click the “Next” button



11. Once the Gpg4win setup wizard is complete, the following screen will be displayed. Click the “Finish” button

12. If you do not uncheck the “Show the README file” check box, the README file will be displayed. The window can be closed after you’ve reviewed it.

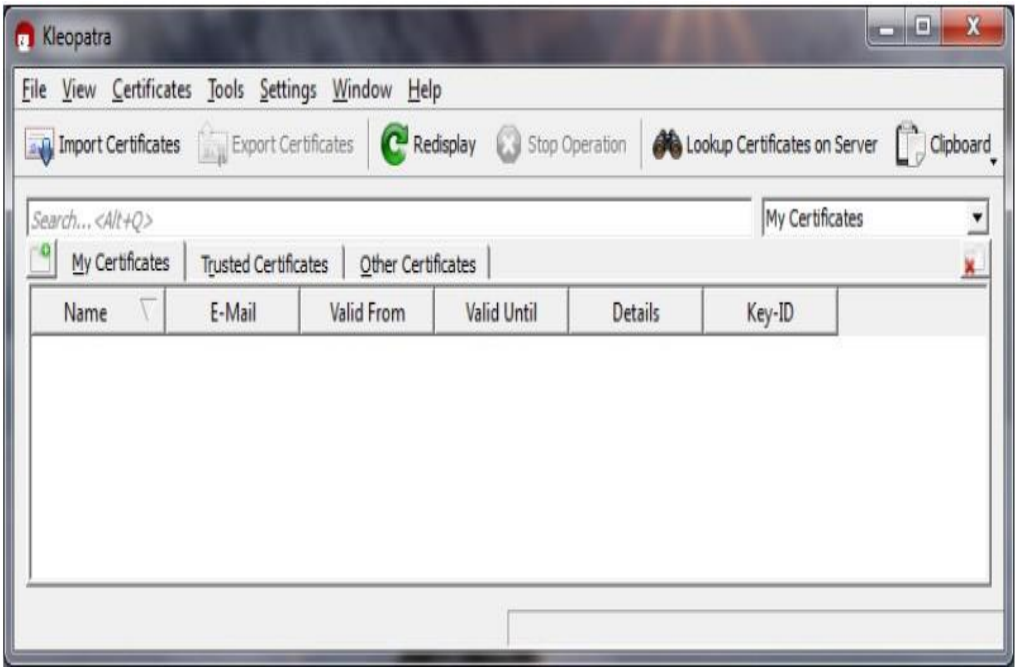


CREATING YOUR PUBLIC AND PRIVATE KEYS

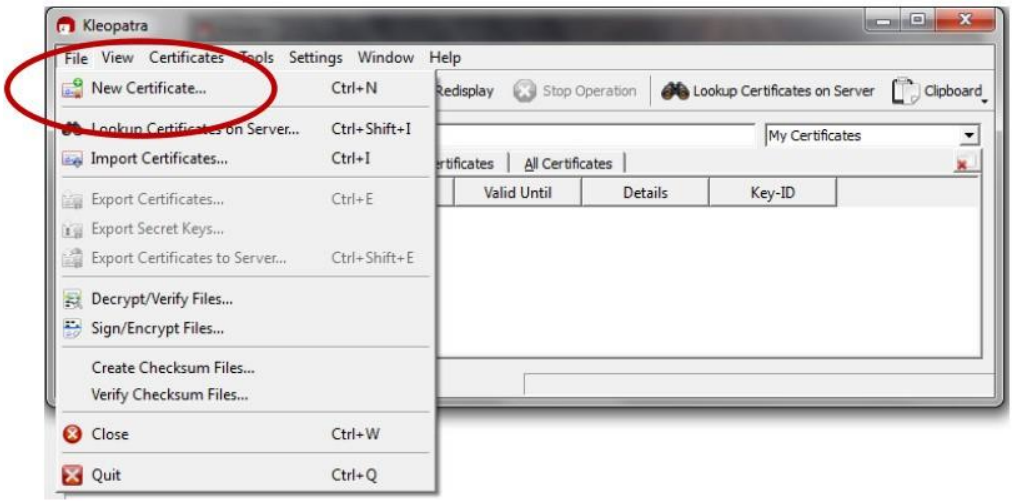
GPG encryption and decryption is based upon the keys of the person who will be receiving the encrypted file or message. Any individual who wants to send the person an encrypted file or message must possess the recipient's public key certificate to encrypt the message. The recipient must have the associated private key, which is different than the public key, to be able to decrypt the file. The public and private key pair for an individual is usually generated by the individual on his or her computer using the installed GPG program, called "Kleopatra" and the following procedure:

1. From your start bar, select the "Kleopatra" icon to start the Kleopatra certificate management software

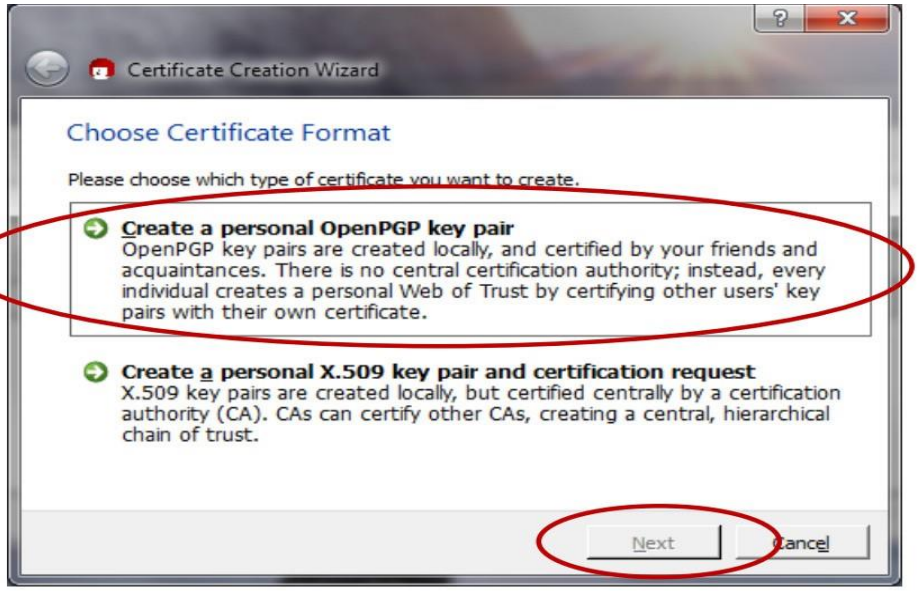
2. The following screen will be displayed



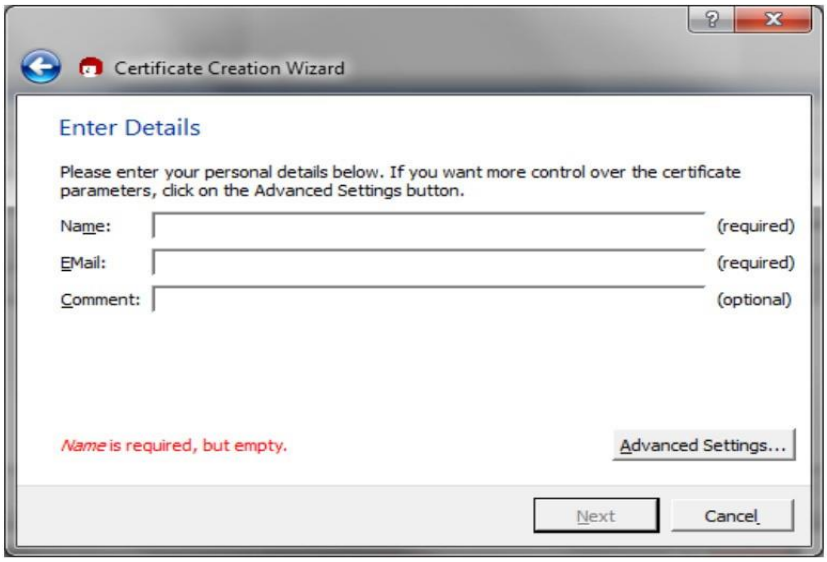
3. From the “File” dropdown, click on the “New Certificate” option



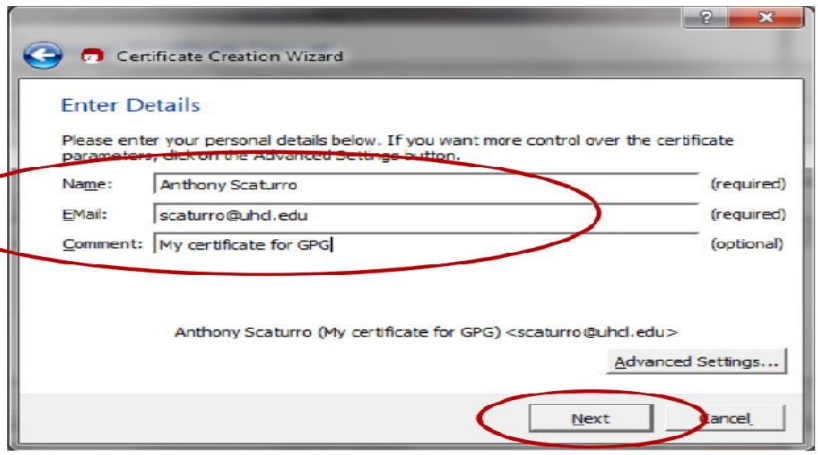
4. The following screen will be displayed. Click on “Create a personal OpenGPG key pair” and the “Next” button



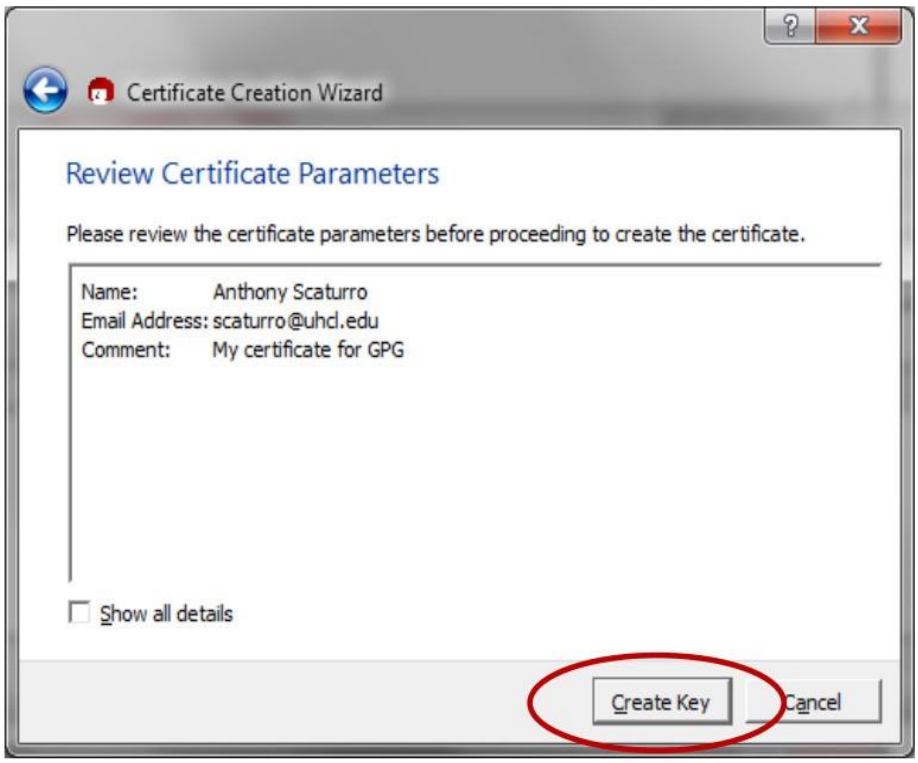
5. The Certificate Creation Wizard will start and display the following:



6. Enter your name and e-mail address. You may also enter an optional comment. Then, click the “Next” button



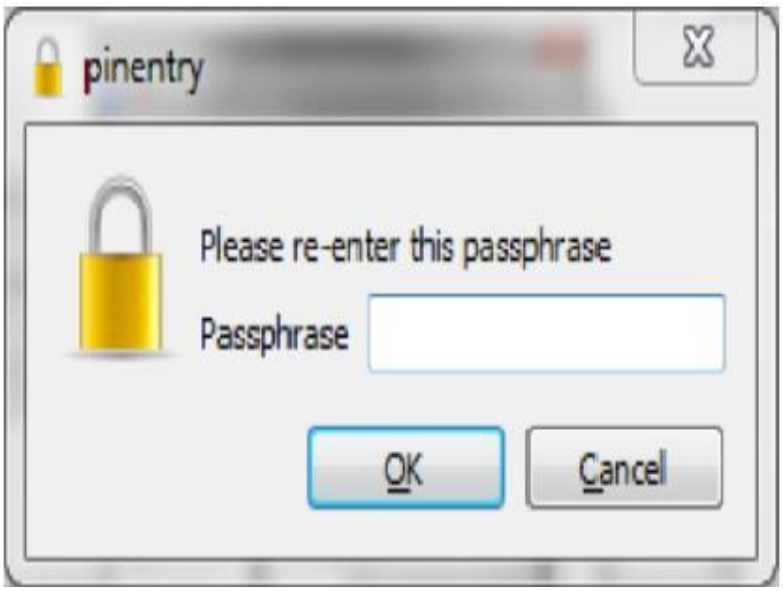
7. Review your entered values. If OK, click the “Create Key” button



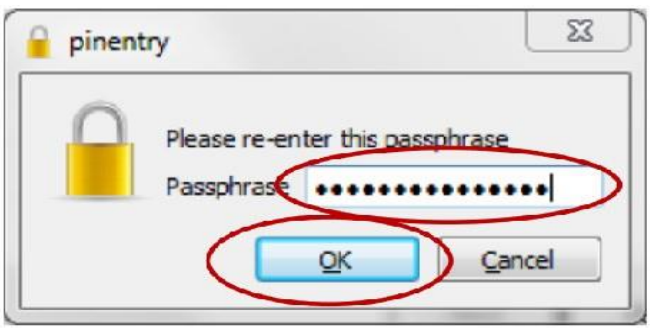
8. You will be asked to enter a passphrase

9. The passphrase should follow strong password standards. After you’ve entered your passphrase, click the “OK” button.

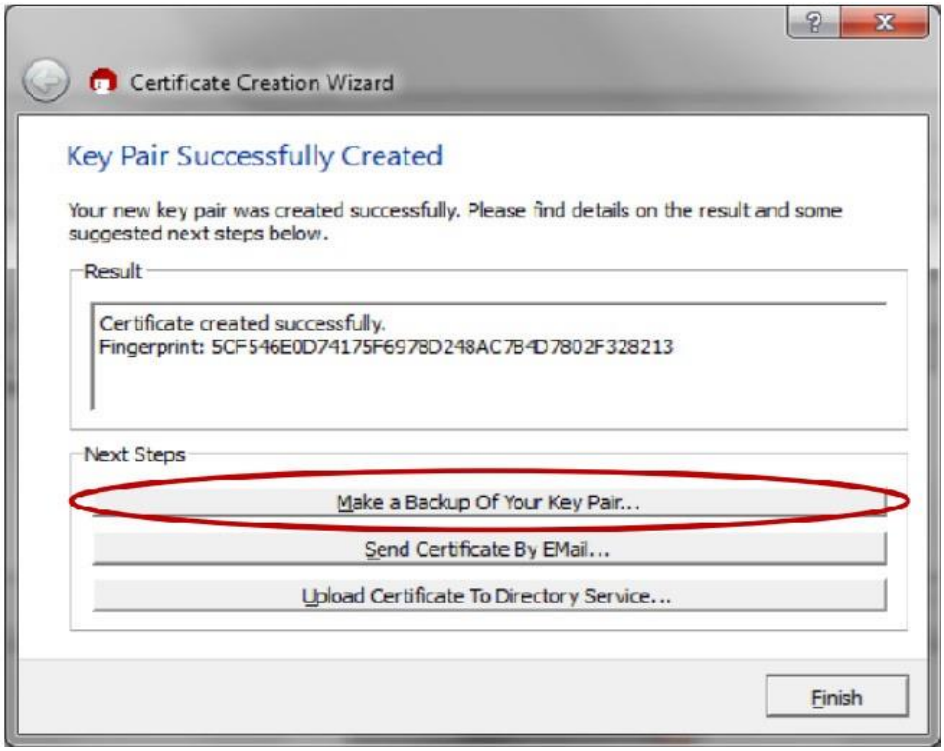
10. You will be asked to re-enter the passphrase



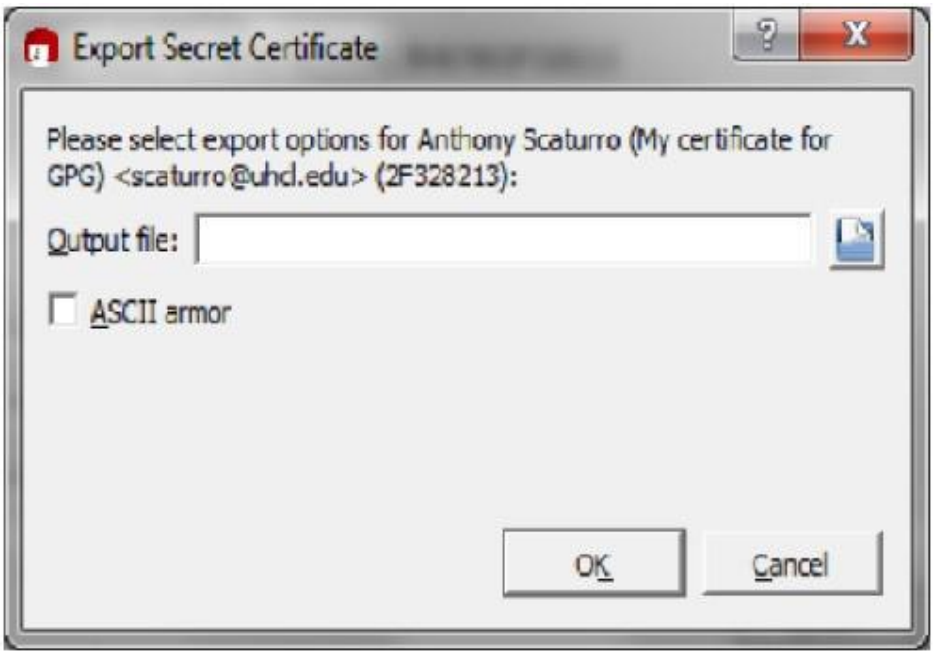
11. Re-enter the passphrase value. Then click the “OK” button. If the passphrases match, the certificate will be created.



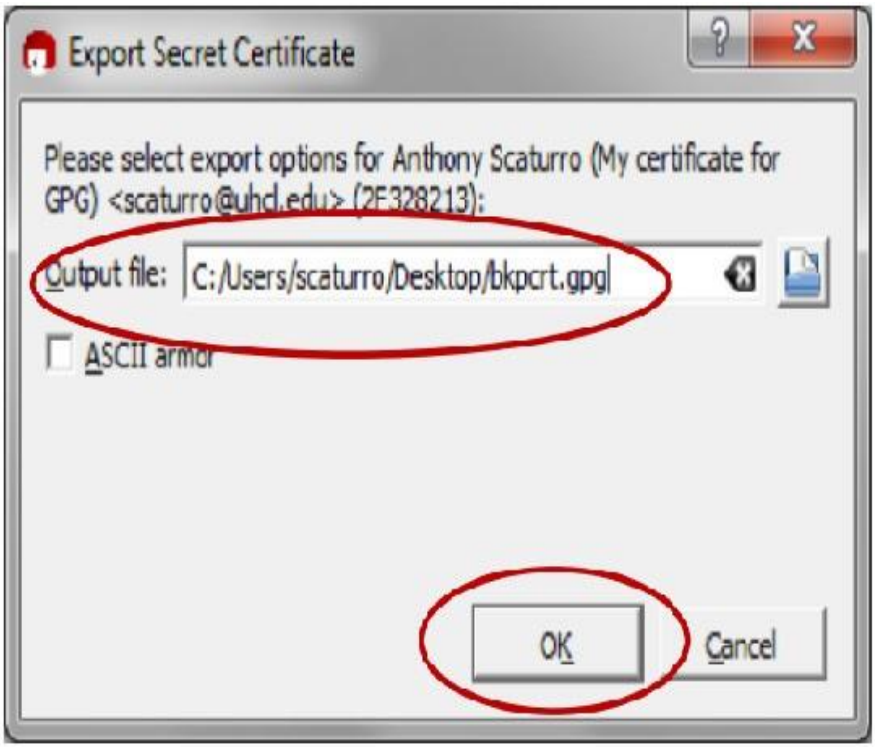
12. Once the certificate is created, the following screen will be displayed. You can save a backup of your public and private keys by clicking the “Make a backup Of Your Key Pair” button. This backup can be used to copy certificates onto other authorized computers.



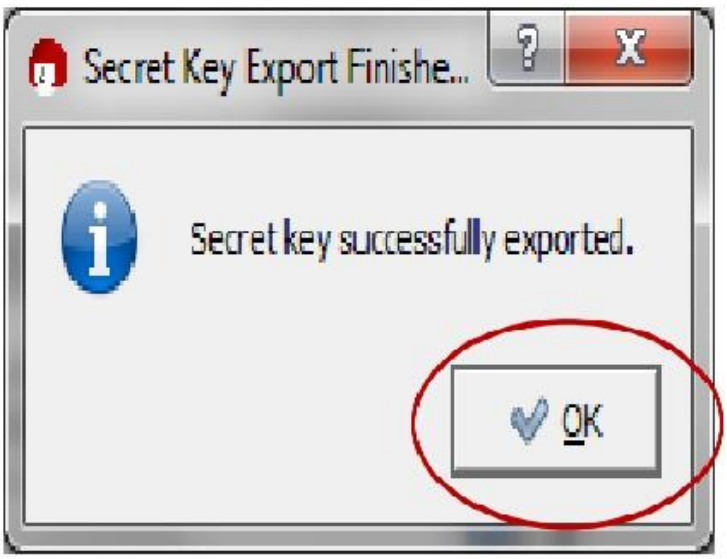
13. If you choose to backup your key pair, you will be presented with the following screen:



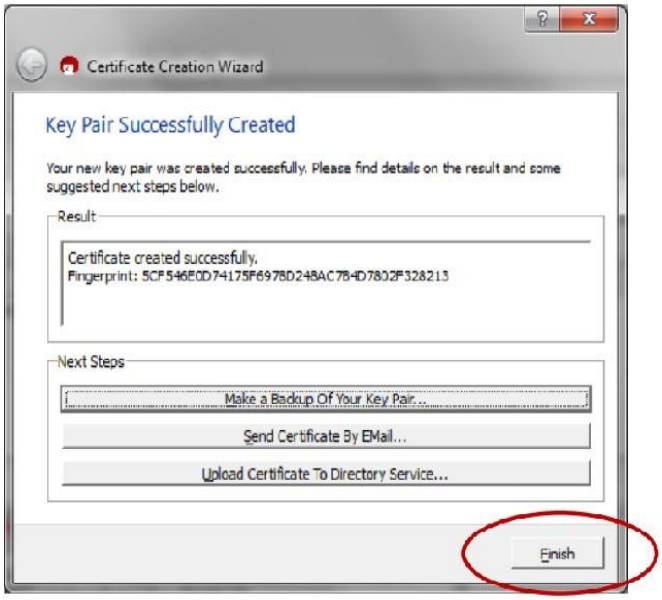
14. Specify the folder and name the file. Then click the “OK” button.



15. After the key is exported, the following will be displayed. Click the “OK” button.



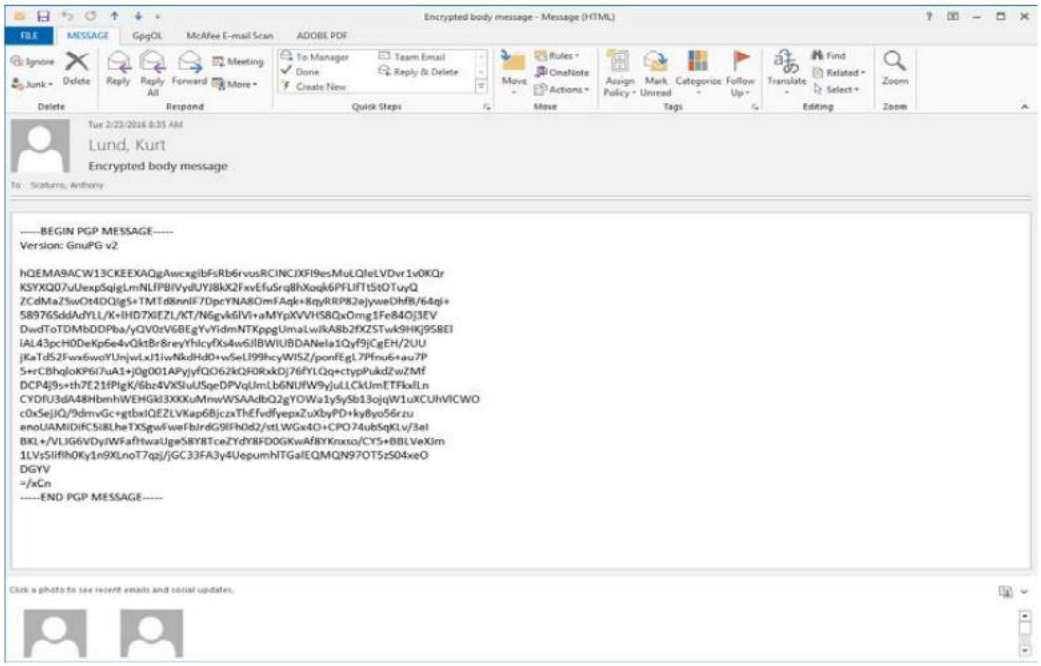
16. You will be returned to the “Key Pair Successfully Created” screen. Click the “Finish” button.



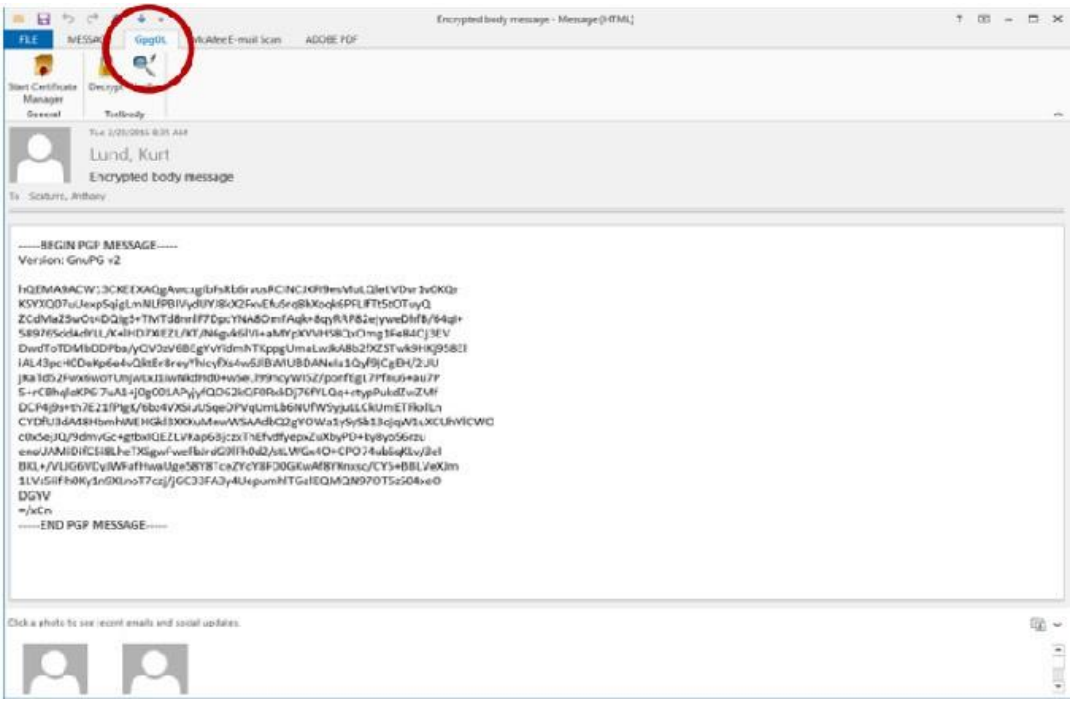
17. Before the program closes, you will need to confirm that you want to close the program by clicking on the "Quit Kleopatra" button

DECRYPTING AN ENCRYPTED E-MAIL THAT HAS BEEN SENT TO YOU:

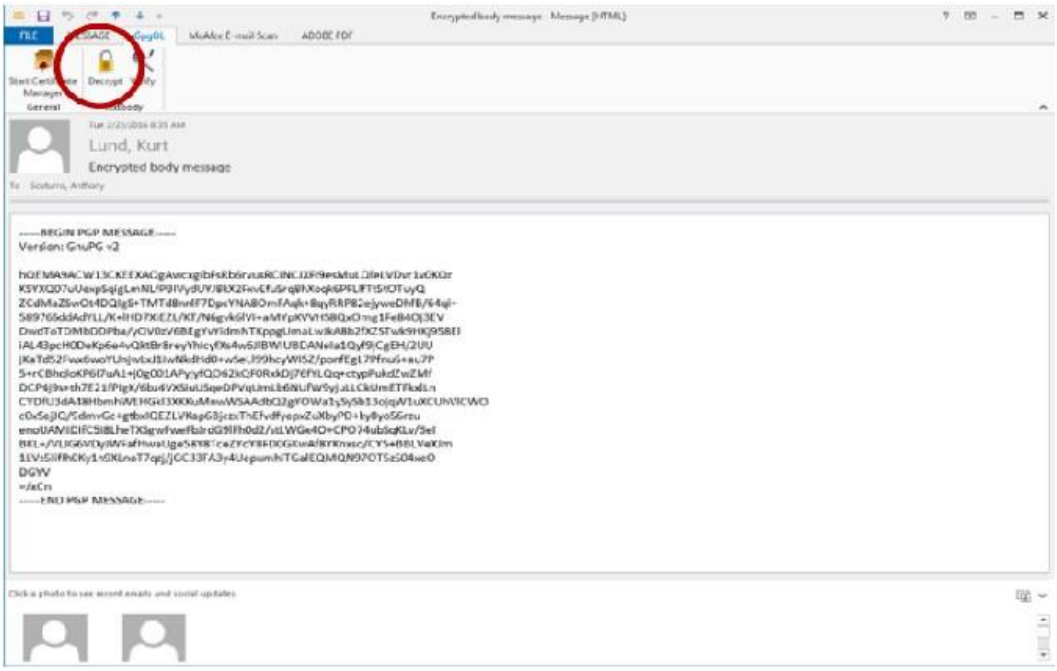
1. Open the e-mail message



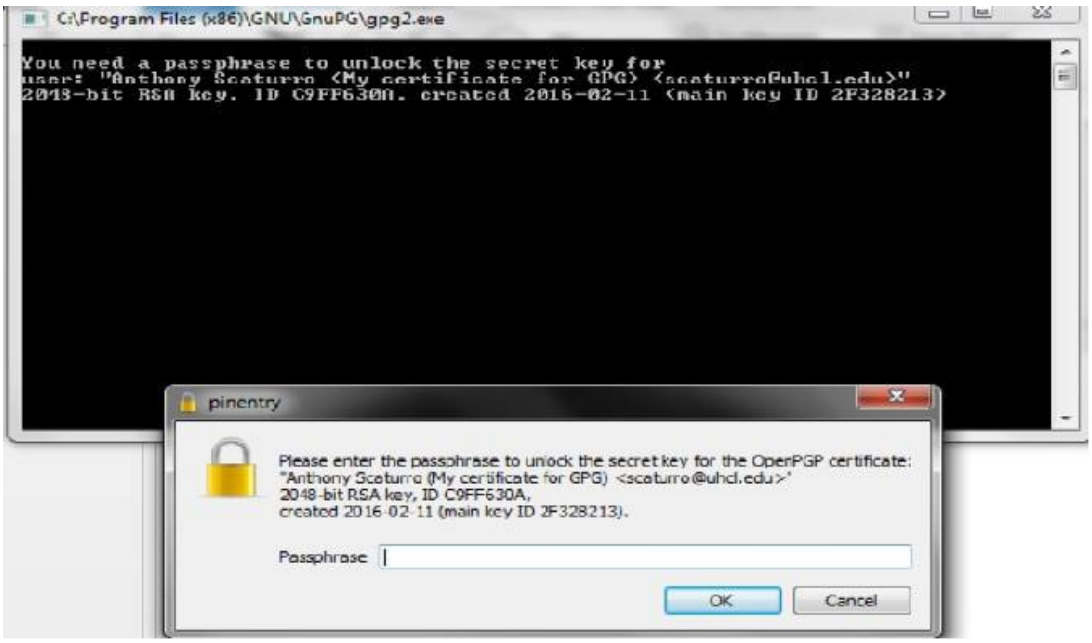
2. Select the GpgOL tab



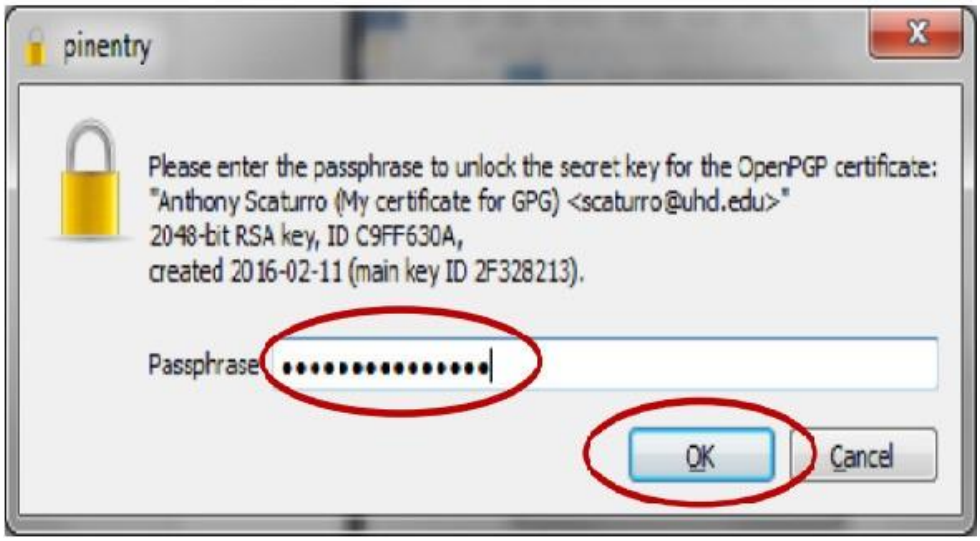
3. Click the “Decrypt” button



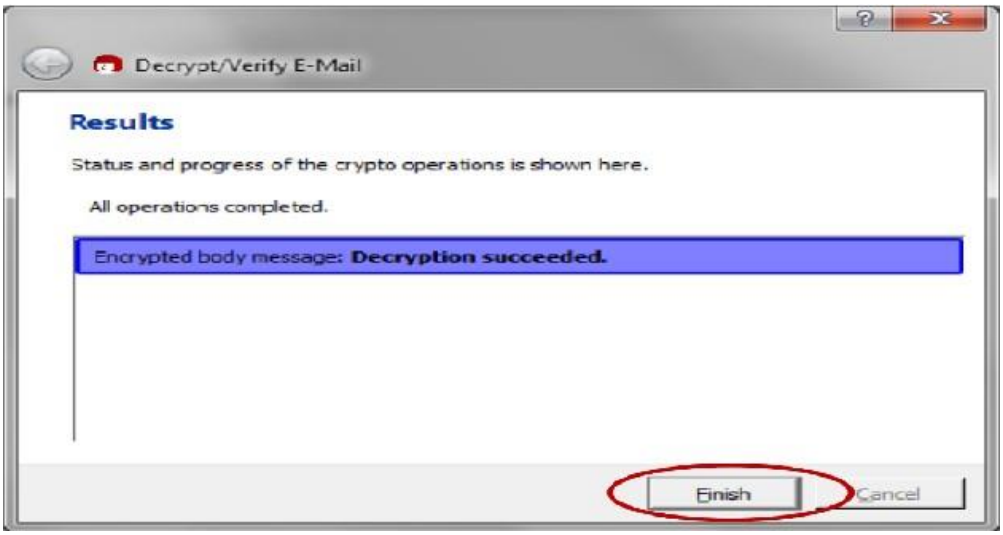
4. A command window will open along with a window that asks for the Passphrase to your private key that will be used to decrypt the incoming message.



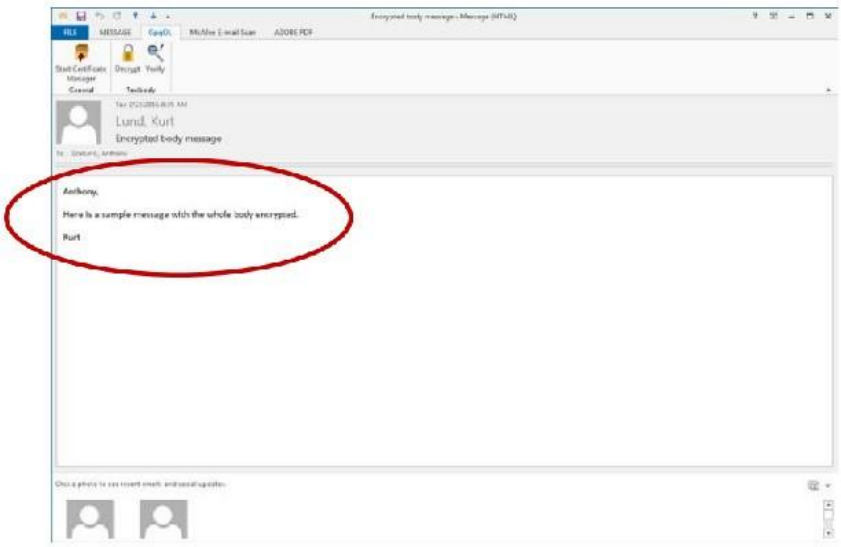
5. Enter your passphrase and click the “OK” button



6. The results window will tell you if the decryption succeeded. Click the "Finish" button top close the window



7. Your unencrypted e-mail message body will be displayed.



8. When you close the e-mail you will be asked if you want to save the e-mail message in its unencrypted form. For maximum security, click the “No” button. This will keep the message encrypted within the e-mail system and will require you to enter your passphrase each time you reopen the e-mail message

Practical-3

Implement MD5 Algorithm in Python/C++/Java.

```cpp
// Base code taken from
// https://github.com/mfontanini/Programs-Scripts/blob/master/constexpr_hashes/md5.h
// and expanded to include a main function

#ifndef CONSTEXPR_HASH_MD5_H
#define CONSTEXPR_HASH_MD5_H

#include <array>
#include <iostream>
#include <cstdint>

namespace ConstexprHashes
{
    // MD5 operations
    constexpr uint32_t f(uint32_t x, uint32_t y, uint32_t z)
    {
        return z ^ (x & (y ^ z));
    }

    constexpr uint32_t g(uint32_t x, uint32_t y, uint32_t z)
    {
        return y ^ (z & (x ^ y));
    }

    constexpr uint32_t h(uint32_t x, uint32_t y, uint32_t z)
    {
        return x ^ y ^ z;
    }

    constexpr uint32_t i(uint32_t x, uint32_t y, uint32_t z)
    {
        return y ^ (x | ~z);
    }
```

```
    constexpr uint32_t step_helper(uint32_t fun_val, uint32_t s, uint32_t
b)
    {
        return ((fun_val << s) | ((fun_val & 0xffffffff) >> (32 - s))) + b;
    }

    // Generic application of the "fun" function

    template <typename Functor>
    constexpr uint32_t step(Functor fun, uint32_t a, uint32_t b, uint32_t
c, uint32_t d, uint32_t x, uint32_t t, uint32_t s)
    {
        return step_helper(a + fun(b, c, d) + x + t, s, b);
    }

    // Retrieve the nth uint32_t in the buffer

    constexpr uint32_t data32(const char *data, size_t n)
    {
        return (static_cast<uint32_t>(data[n * 4]) & 0xff) |
               ((static_cast<uint32_t>(data[n * 4 + 1]) << 8) & 0xff00) |
               ((static_cast<uint32_t>(data[n * 4 + 2]) << 16) & 0xff0000)
|
               ((static_cast<uint32_t>(data[n * 4 + 3]) << 24) &
0xff000000);
    }

    // Constants

    constexpr std::array<uint32_t, 64> md5_constants = {{0xd76aa478,
0xe8c7b756, 0x242070db, 0xc1bdceee, 0xf57c0faf, 0x4787c62a,
                                                         0xa8304613,
0xfd469501, 0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,
                                                         0x6b901122,
0xfd987193, 0xa679438e, 0x49b40821, 0xf61e2562, 0xc040b340,
                                                         0x265e5a51,
0xe9b6c7aa, 0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,
                                                         0x21e1cde6,
0xc33707d6, 0xf4d50d87, 0x455a14ed, 0xa9e3e905, 0xfcefa3f8,
                                                         0x676f02d9,
0x8d2a4c8a, 0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,
                                                         0xa4beea44,
0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70, 0x289b7ec6, 0xeaa127fa,
                                                         0xd4ef3085,
0x04881d05, 0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,
                                                         0xf4292244,
0x432aff97, 0xab9423a7, 0xfc93a039, 0x655b59c3, 0x8f0ccc92,
                                                         0xffeff47d,
0x85845dd1, 0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,
                                                         0xf7537e82,
0xbd3af235, 0x2ad7d2bb, 0xeb86d391}};

    constexpr std::array<size_t, 64> md5_shift = {{7, 12, 17, 22, 7, 12,
17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 5, 9, 14, 20, 5, 9, 14, 20,
```

```
                                                    5, 9, 14, 20, 5, 9, 14,
20, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,
                                                    6, 10, 15, 21, 6, 10,
15, 21, 6, 10, 15, 21, 6, 10, 15, 21}};

    constexpr std::array<size_t, 64> md5_indexes = {{0, 1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 6, 11, 0, 5, 10, 15, 4,
                                                     9, 14, 3, 8, 13, 2, 7,
12, 5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2,
                                                     0, 7, 14, 5, 12, 3,
10, 1, 8, 15, 6, 13, 4, 11, 2, 9}};

    // Functions applied

    constexpr std::array<decltype(f) *, 4> md5_functions = {{f, g, h, i}};

    /******************** Initial buffer generators
***********************/

    // index_tuples to fill the initial buffer

    template <size_t... indexes>
    struct index_tuple
    {
    };

    template <size_t head, size_t... indexes>
    struct index_tuple<head, indexes...>
    {
        typedef typename index_tuple<head - 1, head - 1, indexes...>::type
type;
    };

    template <size_t... indexes>
    struct index_tuple<0, indexes...>
    {
        typedef index_tuple<indexes...> type;
    };

    template <typename... Args>
    struct index_tuple_maker
    {
        typedef typename index_tuple<sizeof...(Args)>::type type;
    };

    /* This builds the buffer.
     *
     * For indexes < string length: output the ith character in the string.
     * For indexes > string length: output 0.
     * If index == string length: output 0x80
     * If index == 56: output string length << 3
     *
     */

    template <size_t n, size_t i>
    struct buffer_builder
```

```
    {
        static constexpr char make_value(const char *data)
        {
            return (i <= n) ? data[i] : 0;
        }
    };

    template <size_t n>
    struct buffer_builder<n, n>
    {
        static constexpr char make_value(const char *)
        {
            return 0x80;
        }
    };

    template <size_t n>
    struct buffer_builder<n, 56>
    {
        static constexpr char make_value(const char *)
        {
            return n << 3;
        }
    };

    /*
     * Simple array implementation, which allows constexpr access to its
     * elements.
     */

    template <typename T, size_t n>
    struct constexpr_array
    {
        const T array[n];

        constexpr const T *data() const
        {
            return array;
        }
    };

    typedef constexpr_array<char, 64> buffer_type;

    template <size_t n, size_t... indexes>
    constexpr buffer_type make_buffer_helper(const char (&data)[n],
index_tuple<indexes...>)
    {
        return buffer_type{{buffer_builder<n - 1,
indexes>::make_value(data)...}};
    }

    // Creates the actual buffer

    template <size_t n>
    constexpr buffer_type make_buffer(const char (&data)[n])
    {
```

```
        return make_buffer_helper(data, index_tuple<64>::type());
    }

    /************************ MD5 impl ***************************/

    typedef std::array<char, 16> md5_type;

    /*
     * There are 64 steps. The ith step has the same structure as the ith +
4 step.
     * That means that we can repeat the same structure, and pick the
appropiate
     * constants and function to apply, depending on the step number.
     */

    template <size_t n, size_t rot>
    struct md5_step;

    /*
     * Nasty, but works. Convert the MD5 result(which is 4 uint32_t), to
     * a std::array<char, 16>.
     */

    constexpr md5_type make_md5_result(uint32_t a, uint32_t b, uint32_t c,
uint32_t d)
    {
        typedef md5_type::value_type value_type;
        return md5_type{{
            static_cast<value_type>(a & 0xff),
            static_cast<value_type>((a & 0xff00) >> 8),
            static_cast<value_type>((a & 0xff0000) >> 16),
            static_cast<value_type>((a & 0xff000000) >> 24),

            static_cast<value_type>(b & 0xff),
            static_cast<value_type>((b & 0xff00) >> 8),
            static_cast<value_type>((b & 0xff0000) >> 16),
            static_cast<value_type>((b & 0xff000000) >> 24),

            static_cast<value_type>(c & 0xff),
            static_cast<value_type>((c & 0xff00) >> 8),
            static_cast<value_type>((c & 0xff0000) >> 16),
            static_cast<value_type>((c & 0xff000000) >> 24),

            static_cast<value_type>(d & 0xff),
            static_cast<value_type>((d & 0xff00) >> 8),
            static_cast<value_type>((d & 0xff0000) >> 16),
            static_cast<value_type>((d & 0xff000000) >> 24),
        }};
    }

    template <>
    struct md5_step<64, 0>
    {
        static constexpr md5_type do_step(const char *, uint32_t a,
uint32_t b, uint32_t c, uint32_t d)
        {
```

```
            return make_md5_result(a + 0x67452301, b + 0xefcdab89, c +
0x98badcfe, d + 0x10325476);
        }
    };

    template <size_t n>
    struct md5_step<n, 3>
    {
        static constexpr md5_type do_step(const char *data, uint32_t a,
uint32_t b, uint32_t c, uint32_t d)
        {
            return md5_step<n + 1, (n + 1) % 4>::do_step(data, a,
step(md5_functions[n / 16], b, c, d, a, data32(data, md5_indexes[n]),
md5_constants[n], md5_shift[n]), c, d);
        }
    };

    template <size_t n>
    struct md5_step<n, 2>
    {
        static constexpr md5_type do_step(const char *data, uint32_t a,
uint32_t b, uint32_t c, uint32_t d)
        {
            return md5_step<n + 1, (n + 1) % 4>::do_step(data, a, b,
step(md5_functions[n / 16], c, d, a, b, data32(data, md5_indexes[n]),
md5_constants[n], md5_shift[n]), d);
        }
    };

    template <size_t n>
    struct md5_step<n, 1>
    {
        static constexpr md5_type do_step(const char *data, uint32_t a,
uint32_t b, uint32_t c, uint32_t d)
        {
            return md5_step<n + 1, (n + 1) % 4>::do_step(data, a, b, c,
step(md5_functions[n / 16], d, a, b, c, data32(data, md5_indexes[n]),
md5_constants[n], md5_shift[n]));
        }
    };

    template <size_t n>
    struct md5_step<n, 0>
    {
        static constexpr md5_type do_step(const char *data, uint32_t a,
uint32_t b, uint32_t c, uint32_t d)
        {
            return md5_step<n + 1, (n + 1) % 4>::do_step(data,
step(md5_functions[n / 16], a, b, c, d, data32(data, md5_indexes[n]),
md5_constants[n], md5_shift[n]), b, c, d);
        }
    };

    template <size_t n>
    constexpr md5_type md5(const char (&data)[n])
    {
```

```
        return md5_step<0, 0>::do_step(make_buffer(data).data(),
0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476);
    }

} // namespace ConstexprHashes
#endif // CONSTEXPR_HASH_MD5_H

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{

    auto hash = ConstexprHashes::md5("rohit");
    cout << hex;
    for (auto i : hash)
    {
        cout << (static_cast<int>(i) & 0xff);
    }
    cout << endl;
}
```

output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS

PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals> .\md5.exe
2d235ace0a3ad85f59e321c89bb99
PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals>
```

Practical-4

Implement SHA-256 Algorithm. Consider all the Constants and Tables as given in the textbook.

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long int int64;
int64 Message[80];
const int64 Constants[80] = {0x428a2f98d728ae22, 0x7137449123ef65cd,
                             0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,
                             0x3956c25bf348b538, 0x59f111f1b605d019,
                             0x923f82a4af194f9b, 0xab1c5ed5da6d8118,
                             0xd807aa98a3030242, 0x12835b0145706fbe,
                             0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2,
                             0x72be5d74f27b896f, 0x80deb1fe3b1696b1,
                             0x9bdc06a725c71235, 0xc19bf174cf692694,
                             0xe49b69c19ef14ad2, 0xefbe4786384f25e3,
                             0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
                             0x2de92c6f592b0275, 0x4a7484aa6ea6e483,
                             0x5cb0a9dcbd41fbd4, 0x76f988da831153b5,
                             0x983e5152ee66dfab, 0xa831c66d2db43210,
                             0xb00327c898fb213f, 0xbf597fc7beef0ee4,
                             0xc6e00bf33da88fc2, 0xd5a79147930aa725,
                             0x06ca6351e003826f, 0x142929670a0e6e70,
                             0x27b70a8546d22ffc, 0x2e1b21385c26c926,
                             0x4d2c6dfc5ac42aed, 0x53380d139d95b3df,
                             0x650a73548baf63de, 0x766a0abb3c77b2a8,
                             0x81c2c92e47edaee6, 0x92722c851482353b,
                             0xa2bfe8a14cf10364, 0xa81a664bbc423001,
                             0xc24b8b70d0f89791, 0xc76c51a30654be30,
                             0xd192e819d6ef5218, 0xd69906245565a910,
                             0xf40e35855771202a, 0x106aa07032bbd1b8,
                             0x19a4c116b8d2d0c8, 0x1e376c085141ab53,
                             0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8,
                             0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb,
                             0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,
                             0x748f82ee5defb2fc, 0x78a5636f43172f60,
                             0x84c87814a1f0ab72, 0x8cc702081a6439ec,
                             0x90befffa23631e28, 0xa4506cebde82bde9,
                             0xbef9a3f7b2c67915, 0xc67178f2e372532b,
                             0xca273eceea26619c, 0xd186b8c721c0c207,
                             0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,
                             0x06f067aa72176fba, 0x0a637dc5a2c898a6,
                             0x113f9804bef90dae, 0x1b710b35131c471b,
                             0x28db77f523047d84, 0x32caab7b40c72493,
                             0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c,
                             0x4cc5d4becb3e42b6, 0x597f299cfc657e2a,
                             0x5fcb6fab3ad6faec,
0x6c44198c4a475817};
```

```
string gethex(string bin)
{
    if (bin == "0000")
        return "0";
    if (bin == "0001")
        return "1";
    if (bin == "0010")
        return "2";
    if (bin == "0011")
        return "3";
    if (bin == "0100")
        return "4";
    if (bin == "0101")
        return "5";
    if (bin == "0110")
        return "6";
    if (bin == "0111")
        return "7";
    if (bin == "1000")
        return "8";
    if (bin == "1001")
        return "9";
    if (bin == "1010")
        return "a";
    if (bin == "1011")
        return "b";
    if (bin == "1100")
        return "c";
    if (bin == "1101")
        return "d";
    if (bin == "1110")
        return "e";
    // if (bin == "1111")
        return "f";
}

string decimaltohex(int64 deci)
{
    string EQBIN = bitset<64>(deci).to_string();
    string hexstring = "";
    string temp;
    for (unsigned int i = 0;
         i < EQBIN.length(); i += 4)
    {
        temp = EQBIN.substr(i, 4);
        hexstring += gethex(temp);
    }
    return hexstring;
}
int64 BintoDec(string bin)
```

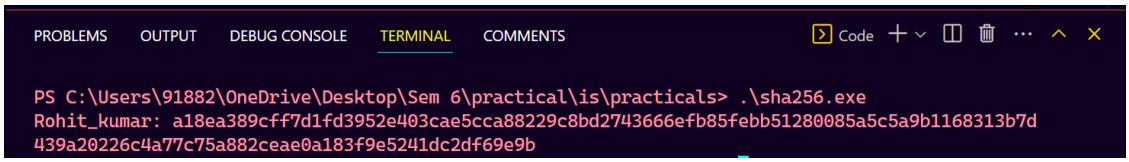
```
{
    int64 value = bitset<64>(bin).to_ullong();
    return value;
}
int64 rotate_right(int64 x, int n)
{
    return (x >> n) | (x << (64 - n));
}
int64 shift_right(int64 x, int n)
{
    return (x >> n);
}
void separator(string getBlock)
{
    int chunknum = 0;
    for (unsigned int i = 0;
         i < getBlock.length();
         i += 64, ++chunknum)
    {
        Message[chunknum] = BintoDec(getBlock.substr(i, 64));
    }
    for (int g = 16; g < 80; ++g)
    {
        int64 WordA = rotate_right(Message[g - 2], 19) ^
rotate_right(Message[g - 2], 61) ^ shift_right(Message[g - 2], 6);
        int64 WordB = Message[g - 7];
        int64 WordC = rotate_right(Message[g - 15], 1) ^
rotate_right(Message[g - 15], 8) ^ shift_right(Message[g - 15], 7);
        int64 WordD = Message[g - 16];
        int64 T = WordA + WordB + WordC + WordD;
        Message[g] = T;
    }
}
int64 maj(int64 a, int64 b, int64 c)
{
    return (a & b) ^ (b & c) ^ (c & a);
}
int64 Ch(int64 e, int64 f, int64 g)
{
    return (e & f) ^ (~e & g);
}
int64 sigmaE(int64 e)
{
    return rotate_right(e, 14) ^ rotate_right(e, 18) ^
rotate_right(e, 41);
}
int64 sigmaA(int64 a)
{
    return rotate_right(a, 28) ^ rotate_right(a, 34) ^
rotate_right(a, 39);
```

```
}
void Func(int64 a, int64 b, int64 c,
          int64 &d, int64 e, int64 f,
          int64 g, int64 &h, int K)
{
    int64 T1 = h + Ch(e, f, g) + sigmaE(e) + Message[K] +
Constants[K];
    int64 T2 = sigmaA(a) + maj(a, b, c);
    d = d + T1;
    h = T1 + T2;
}
string SHA512(string myString)
{
    int64 A = 0x6a09e667f3bcc908;
    int64 B = 0xbb67ae8584caa73b;
    int64 C = 0x3c6ef372fe94f82b;
    int64 D = 0xa54ff53a5f1d36f1;
    int64 E = 0x510e527fade682d1;
    int64 F = 0x9b05688c2b3e6c1f;
    int64 G = 0x1f83d9abfb41bd6b;
    int64 H = 0x5be0cd19137e2179;
    int64 AA, BB, CC, DD, EE, FF, GG, HH;
    stringstream fixedstream;
    for (int i = 0;
         i < myString.size(); ++i)
    {
        fixedstream << bitset<8>(myString[i]);
    }
    string s1024;
    s1024 = fixedstream.str();
    int orilen = s1024.length();
    int tobeadded;
    int modded = s1024.length() % 1024;
    if (1024 - modded >= 128)
    {
        tobeadded = 1024 - modded;
    }
    else if (1024 - modded < 128)
    {
        tobeadded = 2048 - modded;
    }
    s1024 += "1";
    for (int y = 0; y < tobeadded - 129; y++)
    {
        s1024 += "0";
    }
    string lengthbits = std::bitset<128>(orilen).to_string();
    s1024 += lengthbits;
    int blocksnumber = s1024.length() / 1024;
    int chunknum = 0;
```

```
    string Blocks[blocksnumber];
    for (int i = 0; i < s1024.length();
         i += 1024, ++chunknum)
    {
        Blocks[chunknum] = s1024.substr(i, 1024);
    }
    for (int letsgo = 0;
         letsgo < blocksnumber;
         ++letsgo)
    {
        separator(Blocks[letsgo]);
        AA = A;
        BB = B;
        CC = C;
        DD = D;
        EE = E;
        FF = F;
        GG = G;
        HH = H;
        int count = 0;
        for (int i = 0; i < 10; i++)
        {
            Func(A, B, C, D, E, F, G, H, count);
            count++;
            Func(H, A, B, C, D, E, F, G, count);
            count++;
            Func(G, H, A, B, C, D, E, F, count);
            count++;
            Func(F, G, H, A, B, C, D, E, count);
            count++;
            Func(E, F, G, H, A, B, C, D, count);
            count++;
            Func(D, E, F, G, H, A, B, C, count);
            count++;
            Func(C, D, E, F, G, H, A, B, count);
            count++;
            Func(B, C, D, E, F, G, H, A, count);
            count++;
        }
        A += AA;
        B += BB;
        C += CC;
        D += DD;
        E += EE;
        F += FF;
        G += GG;
        H += HH;
    }
    stringstream output;
    output << decimaltohex(A);
```

```
    output << decimaltohex(B);
    output << decimaltohex(C);
    output << decimaltohex(D);
    output << decimaltohex(E);
    output << decimaltohex(F);
    output << decimaltohex(G);
    output << decimaltohex(H);
    return output.str();
}
int main()
{
    string S = "Rohit_kumar";
    cout << S << ": " << SHA512(S);
    return 0;
}
```

OUTPUT

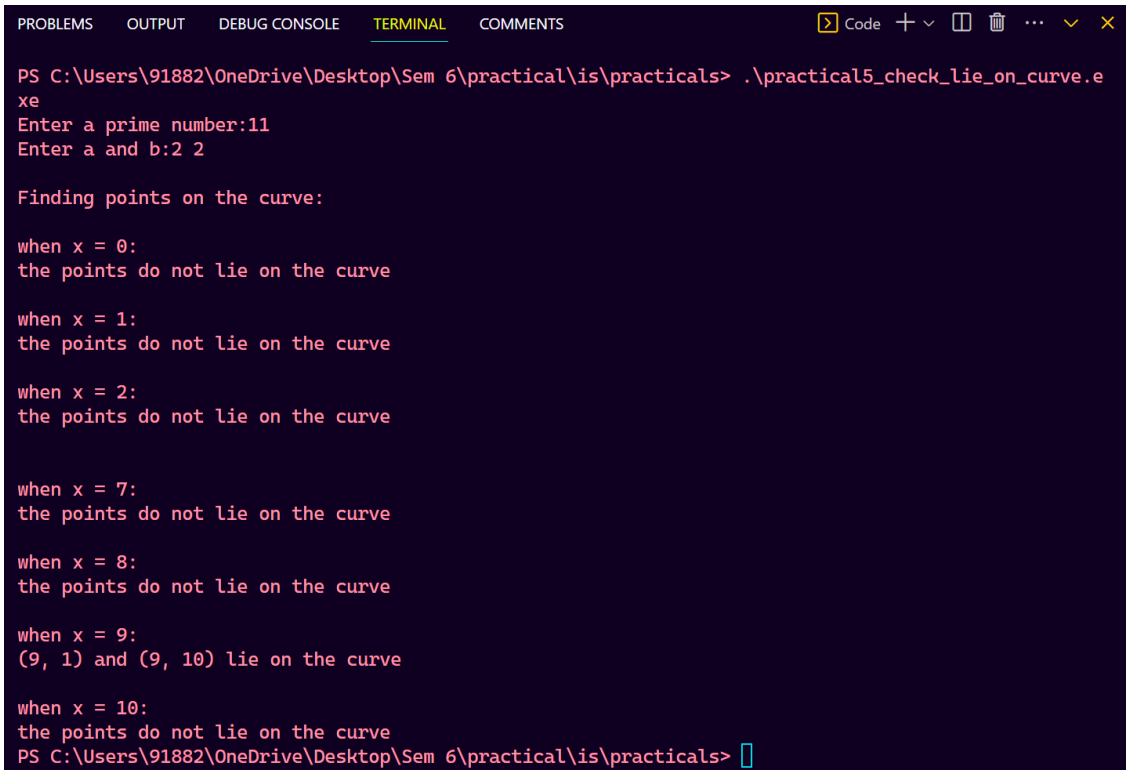


Practical-5

Plot an elliptic curve over a finite field. Check whether the points lie on the elliptic curve or not.

```cpp
#include <bits/stdc++.h>
using namespace std;
void ellipticCurve_points(int p, int a, int b)
{
    int x = 0, w;
    while (x < p)
    {
        cout << "\nwhen x = " << x << ":\n";
        w = (pow(x, 3) + (a * x) + b);
        w = w % p;
        if (floor(sqrt(w)) == sqrt(w) || ceil(sqrt(w)) ==
                                            sqrt(w))
        {
            int y = sqrt(w);
            cout << "(" << x << ", " << y << ") and (" << x
                 << ", " << (p - y) % p << ") lie on the curve\n";
        }
        else
        {
            cout << "the points do not lie on the curve\n";
        }
        ++x;
    }
}
int main()
{
    int p, a, b;
    cout << "Enter a prime number:";
    cin >> p;
    cout << "Enter a and b:";
    cin >> a >> b;
    cout << "\nFinding points on the curve:\n";
    ellipticCurve_points(p, a, b);
    return 0;
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS
Code
PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals> .\practical5_check_lie_on_curve.exe
Enter a prime number:11
Enter a and b:2 2
Finding points on the curve:
when x = 0:
the points do not lie on the curve
when x = 1:
the points do not lie on the curve
when x = 2:
the points do not lie on the curve
when x = 7:
the points do not lie on the curve
when x = 8:
the points do not lie on the curve
when x = 9:
(9, 1) and (9, 10) lie on the curve
when x = 10:
the points do not lie on the curve
PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals>

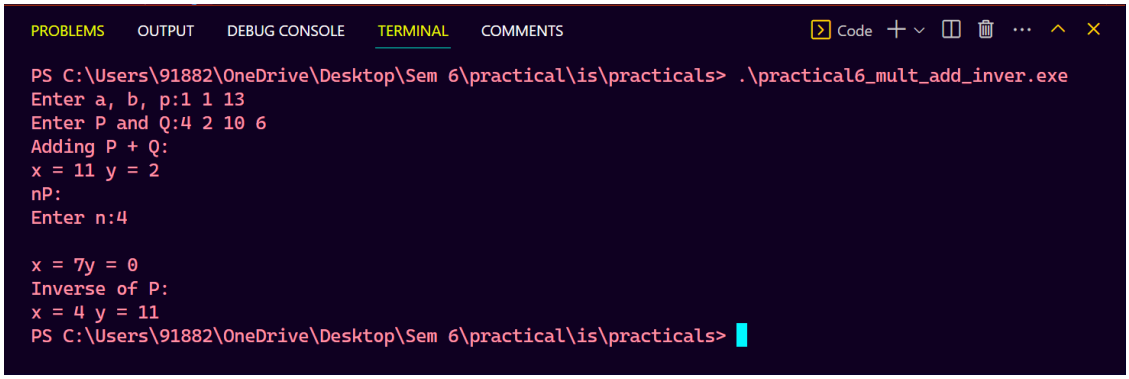
Practical-6

Perform the Elliptic Curve operations like Addition and Multiplication of two points and find the Inverse of a point also.

```
#include <bits/stdc++.h>
using namespace std;
vector<int> xcord, ycord;
void ellipticCurve_points(int p, int a, int b)
{
    int x = 0, w;
    while (x < p)
    {
        w = (pow(x, 3) + (a * x) + b);
        w = w % p;
        if (floor(sqrt(w)) == sqrt(w) || ceil(sqrt(w)) ==
                                              sqrt(w))
        {
            int y = sqrt(w);
            if (y < (p - y))
            {
                xcord.push_back(x);
                ycord.push_back(y);
                xcord.push_back(x);
                ycord.push_back(p - y);
            }
            else
            {
                xcord.push_back(x);
                ycord.push_back(p - y);
                xcord.push_back(x);
                ycord.push_back(y);
            }
        }
        ++x;
    }
}
int modInverse(int a, int m)
{
    for (int x = 1; x < m; x++)
        if (((a % m) * (x % m)) % m == 1)
            return x;
    return 0;
}
int main()
{
    int a, b, p, x1, y1, x2, y2, x3, y3, m, n;
    cout << "Enter a, b, p:";
    cin >> a >> b >> p;
    cout << "Enter P and Q:";
    cin >> x1 >> y1 >> x2 >> y2;
```

```
    cout << "Adding P + Q:";
    m = ((y2 - y1) * modInverse((x2 - x1), p)) % p;
    x3 = ((m * m) - x1 - x2) % p;
    y3 = (m * (x1 - x3) - y1) % p;
    if (y3 < 0)
    {
        cout << "\nx = " << x3 << " y = " << p + y3;
    }
    else
    {
        cout << "\nx = " << x3 << " y = " << y3;
    }
    cout << "\nnP:";
    cout << "\nEnter n:";
    cin >> n;
    ellipticCurve_points(p, a, b);
    int point;
    for (int i = 0; i < xcord.size(); ++i)
    {
        if (xcord[i] == x1 && ycord[i] == y1)
        {
            point = i;
        }
    }
    cout << "\nx = " << xcord[(point + n) % xcord.size()] << "y = "
<< ycord[(point + n) % xcord.size()];
    cout << "\nInverse of P:";
    cout << "\nx = " << x1 << " y = " << p - y1;
    return 0;
}
```

OUTPUT



Practical-7

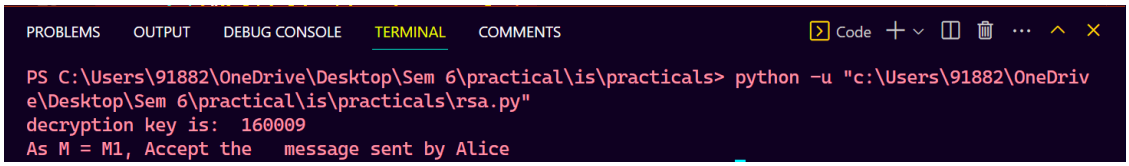
Implement RSA Digital Signature Scheme.

```
# Function to find gcd
# of two numbers
def euclid(m, n):

    if n == 0:
        return m
    else:
        r = m % n
        return euclid(n, r)


# Program to find
# Multiplicative inverse
def exteuclid(a, b):

    r1 = a
    r2 = b
    s1 = int(1)
    s2 = int(0)
    t1 = int(0)
    t2 = int(1)

    while r2 > 0:

        q = r1//r2
        r = r1-q * r2
        r1 = r2
        r2 = r
        s = s1-q * s2
        s1 = s2
        s2 = s
        t = t1-q * t2
        t1 = t2
        t2 = t

    if t1 < 0:
        t1 = t1 % a

    return (r1, t1)


# Enter two large prime
# numbers p and q
p = 823
q = 953
n = p * q
```

```
Pn = (p-1)*(q-1)

# Generate encryption key
# in range 1<e<Pn
key = []

for i in range(2, Pn):

    gcd = euclid(Pn, i)

    if gcd == 1:
        key.append(i)


# Select an encryption key
# from the above list
e = int(313)

# Obtain inverse of
# encryption key in Z_Pn
r, d = exteuclid(Pn, e)
if r == 1:
    d = int(d)
    print("decryption key is: ", d)

else:
    print("Multiplicative inverse for\
    the given encryption key does not \
    exist. Choose a different encryption key ")


# Enter the message to be sent
M = 19070

# Signature is created by Alice
S = (M**d) % n

# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
# and product n.
M1 = (S**e) % n

# If M = M1 only then Bob accepts
# the message sent by Alice.

if M == M1:
    print("As M = M1, Accept the\
    message sent by Alice")
else:
```

```
    print("As M not equal to M1,\
    Do not accept the message\
    sent by Alice ")
```

OUTPUT



Practical-8

Implement Elgamal Digital Signature Scheme.

```
#include <bits/stdc++.h>
using namespace std;
int modInverse(int a, int m)
{
    for (int x = 1; x < m; x++)
        if (((a % m) * (x % m)) % m == 1)
            return x;
    return 0;
}
int main()
{
    int p, e1, e2, d, r, M;
    cout << "\nEnter the values of p, e1, d, and r:\n";
    cin >> p >> e1 >> d >> r;
    cout << "\nEnter the message:\n";
    cin >> M;
    cout << "\nSigning:\n";
    int S1, S2;
    e2 = e1;
    for (int i = 1; i < d; ++i)
    {
        e2 = (e2 * e1) % p;
    }
    S1 = e1;
    for (int i = 1; i < r; ++i)
    {
        S1 = (S1 * e1);
        S1 = S1 % p;
    }
    int x = (d * S1) % (p - 1);
    S2 = M - x;
    if (S2 < 0)
    {
        S2 = (p - 1) + S2;
    }
    S2 = S2 * modInverse(r, p - 1);
    S2 = S2 % (p - 1);
```

```
    cout << "\nThe signatures are:\n"
         << "S1 = " << S1 << ",S2 = " << S2 << endl;
    cout << "\nVerifying:\n";
    int V1, V2;
    V1 = e1;
    for (int i = 1; i < M; ++i)
    {
        V1 = V1 * e1;
        V1 = V1 % p;
    }
    cout << "\nV1 = " << V1;
    int a = d, b = S1;
    for (int i = 1; i < S1; ++i)
    {
        a = a * d;
        if (a > p)
        {
            a = a % p;
        }
    }
    for (int i = 1; i < S2; ++i)
    {
        b = b * S1;
        if (b > p)
        {
            b = b % p;
        }
    }
    V2 = (a * b) % p;
    cout << "\nV2 = " << V2;
    if (V1 != V2)
    {
        cout << "\nSince V1 and V2 are not congruent, the receiver
rejects the message";
    }
    else
    {
        cout << "\nMessage Accepted";
    }
    return 0;
}
```

Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS                    Code

PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals> .\elgamal.exe

Enter the values of p, e1, d, and r:
3119 2 127 307

Enter the message:
320

Signing:

The signatures are:
S1 = 2083,S2 = 2105

Verifying:

V1 = 3006
V2 = 1723
Since V1 and V2 are not congruent, the receiver rejects the message
PS C:\Users\91882\OneDrive\Desktop\Sem 6\practical\is\practicals>
```

Practical-9

Implement Schnorr Digital Signature Scheme.

```cpp
#include <bits/stdc++.h>
using namespace std;
int modInverse(int a, int m)
{
    for (int x = 1; x < m; x++)
        if (((a % m) * (x % m)) % m == 1)
            return x;
    return 0;
}
int main()
{
    int p = 2267, q = 103, e0 = 2, e1, e2, d = 30, r = 11, M = 1000;
    /*cout << "\nEnter the values of p, e1, d, and r:\n";
    cin >> p >> e1 >> d >> r;
    cout << "\nEnter the message:\n";
    cin >> M;*/
    cout << "\nSigning:\n";
    int S1, S2, exp;
    exp = (p - 1) / q;
    e1 = e0;
    for (int i = 0; i < exp; ++i)
    {
        e1 = e1 * e0;
        e1 = e1 % p;
    }
    e2 = e1;
    for (int i = 1; i < r; ++i)
```

```
    {
        e2 = (e2 * e1) % p;
    }
    S1 = stoi(to_string(M) + to_string(e2));
    S1 = rand() % S1;
    S2 = (r + (d * S1)) % p;
    cout << "\nThe signatures are:\n"
         << "S1 = " << S1 << ",S2 = " << S2 << endl;
    cout << "\nVerifying:\n";
    int V, a, b;
    a = e1;
    for (int i = 0; i < S2; ++i)
    {
        a = a * e1;
        a = a % p;
    }
    b = e2;
    for (int i = 0; i < S1; ++i)
    {
        b = b * e2;
        b = b % p;
    }
    b = modInverse(b, p);
    int c = (a * b) % p;
    V = stoi(to_string(M) + to_string(c));
    cout << "\nV = " << V;
    cout << "\nIf h(V) is congruent to h(S1) then the message is
accepted, otherwise rejected";
    return 0;
}
```

Output

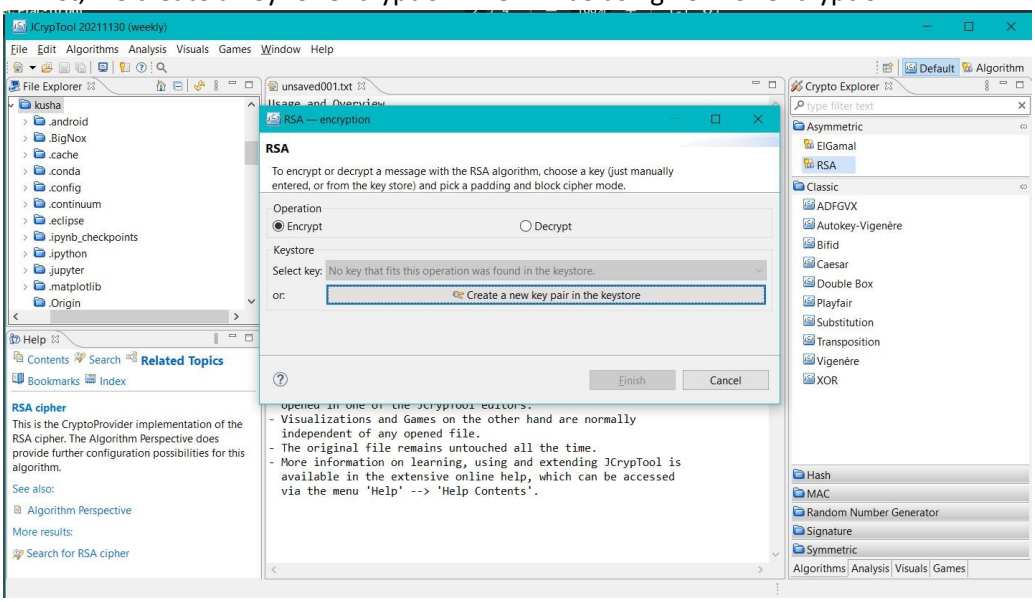
```
Signing:


The signatures are:
S1 = 41,S2 = 1241

Verifying:

V = 10001990
If h(V) is congruent to h(S1) then the message is accepted, otherwise rejected
```

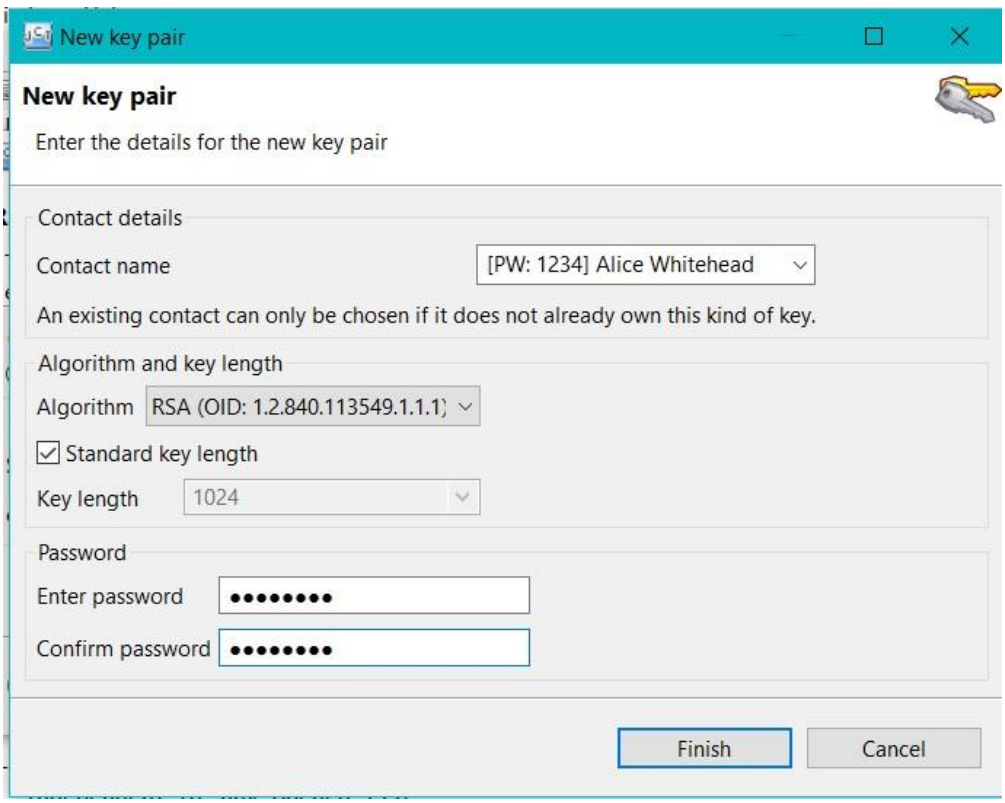
Practical-10

Using the Jcrypt tool (or any other equivalent) to demonstrate an asymmetric, symmetric crypto algorithm

1. Asymmetric cryptographic algorithm

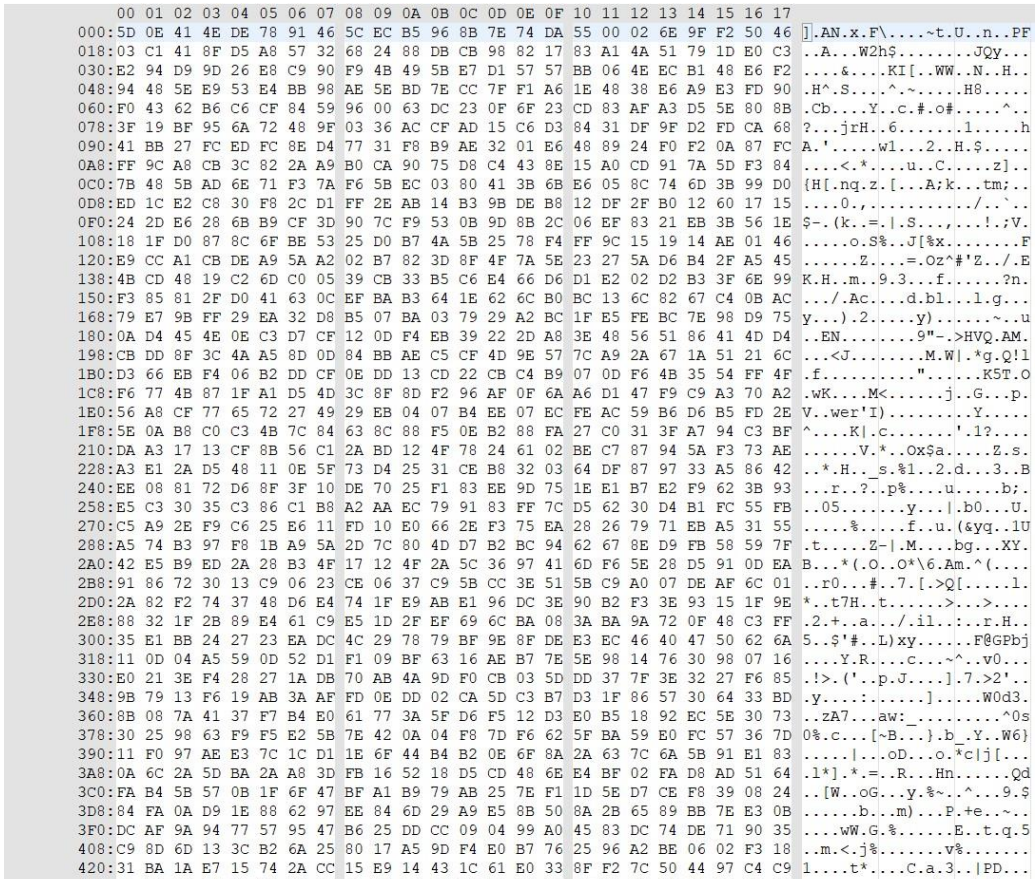
   First, we create a key for encryption. We will be using RSA for encryption





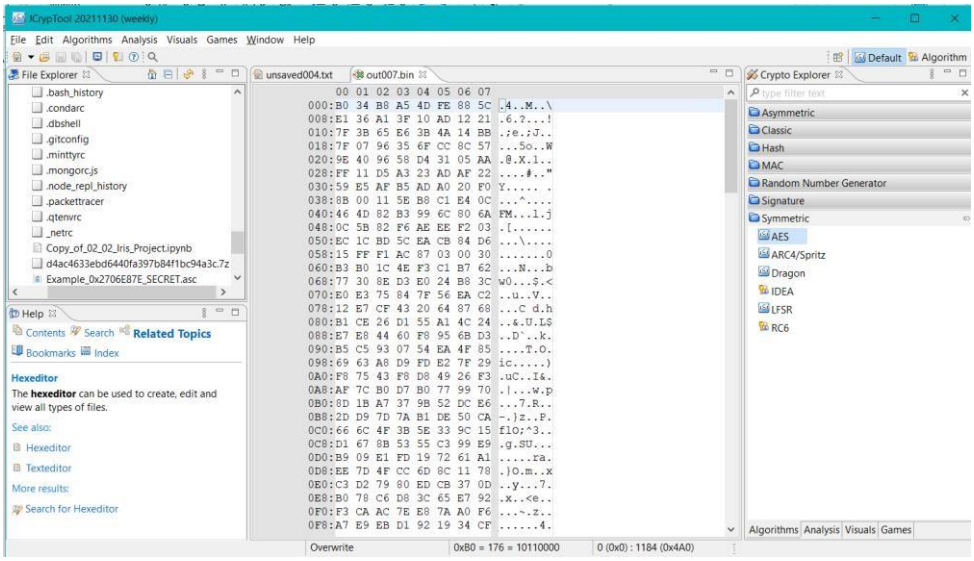
Using this key, we will encrypt the sample text file

```
Usage and Overview
==================

Use this sample file for a quick start with JCrypTool, the
platform-independent cryptography e-learning platform.
For instance,
- encrypt or digitally sign it using the 'Algorithms' menu,
- apply one of the analysis methods in the 'Analysis' menu.

The fast search helps to find 'actions' like particular algorithms,
analyses, visualizations or games). Open it with the magnifying
glass icon in the toolbar, or by pressing "Ctrl+3".

Details
=======
- The 'Crypto Explorer' on the right side lists all the items
  which are also in the menus 'Algorithms', 'Analysis', 'Visuals'
  and 'Games'.
- A double click on an entry in the 'Algorithms' tab launches
  a wizard which guides you step by step through the process.
- Most algorithms in Algorithms and Analysis require a file
  opened in one of the JCrypTool editors.
- Visualizations and Games on the other hand are normally
  independent of any opened file.
- The original file remains untouched all the time.
- More information on learning, using and extending JCrypTool is
  available in the extensive online help, which can be accessed
  via the menu 'Help' --> 'Help Contents'.
```



```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
000:5D 0E 41 4E DE 78 91 46 5C EC B5 96 8B 7E 74 DA 55 00 02 6E 9F F2 50 46 ].AN.x.F\....~t.U..n..PF
018:03 C1 41 8F D5 A8 57 32 68 24 88 DB CB 98 82 17 83 A1 4A 51 79 1D E0 C3 ..A...W2h$.........JQy...
030:E2 94 D9 9D 26 E8 C9 90 F9 4B 49 5B E7 D1 57 57 BB 06 4E EC B1 48 E6 F2 ....&....KI[..WW..N..H..
048:94 48 5E E9 53 E4 BB 98 AE 5E BD 7E CC 7F F1 A6 1E 48 38 E6 A9 E3 FD 90 .H^.S....^.~.....H8.....
060:F0 43 62 B6 C6 CF 84 59 96 00 63 DC 23 0F 6F 23 CD 83 AF A3 D5 5E 80 8B .Cb....Y..c.#.o#.....^..
078:3F 19 BF 95 6A 72 48 9F 03 36 AC CF AD 15 C6 D3 84 31 DF 9F D2 FD CA 68 ?...jrH..6.......1.....h
090:41 BB 27 FC ED FC 8E D4 77 31 F8 B9 AE 32 01 E6 48 89 24 F0 F2 0A 87 FC A.'.....w1...2..H.$.....
0A8:FF 9C A8 CB 3C 82 2A A9 B0 CA 90 75 D8 C4 43 8E 15 A0 CD 91 7A 5D F3 84 ....<.*....u..C.....z]..
0C0:7B 48 5B AD 6E 71 F3 7A F6 5B EC 03 80 41 3B 6B E6 05 8C 74 6D 3B 99 D0 {H[.nq.z.[...A;k...tm;..
0D8:ED 1C E2 C8 30 F8 2C D1 FF 2E AB 14 B3 9B DE B8 12 DF 2F B0 12 60 17 15 ....0.,........../..`..
0F0:24 2D E6 28 6B B9 CF 3D 90 7C F9 53 0B 9D 8B 2C 06 EF 83 21 EB 3B 56 1E $-.(k..=.|.S..,...!.;V.
108:18 1F D0 87 8C 6F BE 53 25 D0 B7 4A 5B 25 78 F4 FF 9C 15 19 14 AE 01 46 .....o.S%..J[%x........F
120:E9 CC A1 CB DE A9 5A A2 02 B7 82 3D 8F 4F 7A 5E 23 27 5A D6 B4 2F A5 45 ......Z....=.Oz^#'Z../.E
138:4B CD 48 19 C2 6D C0 05 39 CB 33 B5 C6 E4 66 D6 D1 E2 02 D2 B3 3F 6E 99 K.H..m..9.3...f......?n.
150:F3 85 81 2F D0 41 63 0C EF BA B3 64 1E 62 6C B0 BC 13 6C 82 67 C4 0B AC .../.Ac....d.bl...l.g....
168:79 E7 9B FF 29 EA 32 D8 B5 07 BA 03 79 29 A2 BC 1F E5 FE BC 7E 98 D9 75 y...).2.....y).......~..u
180:0A D4 45 4E 0E C3 D7 CF 12 0D F4 EB 39 22 2D A8 3E 48 56 51 86 41 4D D4 ..EN.........9"-.>HVQ.AM.
198:CB DD 8F 3C 4A A5 8D 0D 84 BB AE C5 CF 4D 9E 57 7C A9 2A 67 1A 51 21 6C ...<J.........M.W|.*g.Q!l
1B0:D3 66 EB F4 06 B2 DD CF 0E DD 13 CD 22 CB C4 B9 07 0D F6 4B 35 54 FF 4F .f..........".......K5T.O
1C8:F6 77 4B 87 1F A1 D5 4D 3C 8F 8D F2 96 AF 0F 6A A6 D1 47 F9 C9 A3 70 A2 .wK....M<.......j..G...p.
1E0:56 A8 CF 77 65 72 27 49 29 EB 04 07 B4 EE 07 EC FE AC 59 B6 D6 B5 FD 2E V..wer'I).........Y.....
1F8:5E 0A B8 C0 C3 4B 7C 84 63 8C 88 F5 0E B2 88 FA 27 C0 31 3F A7 94 C3 BF ^....K|.c.......'.1?....
210:DA A3 17 13 CF 8B 56 C1 2A BD 12 4F 78 24 61 02 BE C7 87 94 5A F3 73 AE ......V.*..Ox$a.....Z.s.
228:A3 E1 2A D5 48 11 0E 5F 73 D4 25 31 CE B8 32 03 64 DF 87 97 33 A5 86 42 ..*.H.._s.%1..2.d...3..B
240:EE 08 81 72 D6 8F 3F 10 DE 70 25 F1 83 EE 9D 75 1E E1 B7 E2 F9 62 3B 93 ...r..?..p%....u.....b;.
258:E5 C3 30 35 C3 86 C1 B8 A2 AA EC 79 91 83 FF 7C D5 62 30 D4 B1 FC 55 FB ..05.......y...|.b0...U.
270:C5 A9 2E F9 C6 25 E6 11 FD 10 E0 66 2E F3 75 EA 28 26 79 71 EB A5 31 55 .....%.....f..u.(&yq..1U
288:A5 74 B3 97 F8 1B A9 5A 2D 7C 80 4D D7 B2 BC 94 62 67 8E D9 FB 58 59 7F .t.....Z-|.M....bg...XY.
2A0:42 E5 B9 ED 2A 28 B3 4F 17 12 4F 2A 5C 36 97 41 6D F6 5E 28 D5 91 0D EA B...*(.O..O*\6.Am.^(....
2B8:91 86 72 30 13 C9 06 23 CE 06 37 C9 5B CC 3E 51 5B C9 A0 07 DE AF 6C 01 ..r0...#..7.[.>Q[.....l.
2D0:2A 82 F2 74 37 48 D6 E4 74 1F E9 AB E1 96 DC 3E 90 B2 F3 3E 93 15 1F 9E *..t7H..t......>...>....
2E8:88 32 1F 2B 89 E4 61 C9 E5 1D 2F EF 69 6C BA 08 3A BA 9A 72 0F 48 C3 FF .2.+..a.../.il..:..r.H..
300:35 E1 BB 24 27 23 EA DC 4C 29 78 79 BF 9E 8F DE E3 EC 46 40 47 50 62 6A 5..$'#..L)xy......F@GPbj
318:11 0D 04 A5 59 0D 52 D1 F1 09 BF 63 16 AE B7 7E 5E 98 14 76 30 98 07 16 ....Y.R....c...~^..v0...
330:E0 21 3E F4 28 27 1A DB 70 AB 4A 9D F0 CB 03 5D DD 37 7F 3E 32 27 F6 85 .!>.('..p.J....].7.>2'..
348:9B 79 13 F6 19 AB 3A AF FD 0E DD 02 CA 5D C3 B7 D3 1F 86 57 30 64 33 BD .y....:......]......W0d3.
360:8B 08 7A 41 37 F7 B4 E0 61 77 3A 5F D6 F5 12 D3 E0 B5 18 92 EC 5E 30 73 ..zA7...aw:_.........^0s
378:30 25 98 63 F9 F5 E2 5B 7E 42 0A 04 F8 7D F6 62 5F BA 59 E0 FC 57 36 7D 0%.c...[~B...}.b_.Y..W6}
390:11 F0 97 AE E3 7C 1C D1 1E 6F 44 B4 B2 0E 6F 8A 2A 63 7C 6A 5B 91 E1 83 .....|...oD...o.*c|j[...
3A8:0A 6C 2A 5D BA 2A A8 3D FB 16 52 18 D5 CD 48 6E E4 BF 02 FA D8 AD 51 64 .l*].*.=..R...Hn......Qd
3C0:FA B4 5B 57 0B 1F 6F 47 BF A1 B9 79 AB 25 7E F1 1D 5E D7 CE F8 39 08 24 ..[W..oG...y.%~..^...9.$
3D8:84 FA 0A D9 1E 88 62 97 EE 84 6D 29 A9 E5 8B 50 8A 2B 65 89 BB 7E E3 0B ......b...m)...P.+e..~..
3F0:DC AF 9A 94 77 57 95 47 B6 25 DD CC 09 04 99 A0 45 83 DC 74 DE 71 90 35 ....wW.G.%......E..t.q.5
408:C9 8D 6D 13 3C B2 6A 25 80 17 A5 9D F4 E0 B7 76 25 96 A2 BE 06 02 F3 18 ..m.<.j%.......v%.......
420:31 BA 1A E7 15 74 2A CC 15 E9 14 43 1C 61 E0 33 8F F2 7C 50 44 97 C4 C9 1....t*....C.a.3..|PD...
438:15 A6 09 A4 C4 C2 19 03 1A BC A4 D0 FB A3 6C AF 99 03 5C 66 23 D4 DD B8 ..............l...\f#...
```

2. Symmetric cryptographic algorithm
   We will use the same key to encrypt the sample text using the AES algorithm


JCrypTool 20211130 (weekly)
File Edit Algorithms Analysis Visuals Games Window Help
Default Algorithm
File Explorer
.bash_history
.condarc
.dbshell
.gitconfig
.minttyrc
.mongorc.js
.node_repl_history
.packettracer
.qtenvrc
_netrc
Copy_of_02_02_Iris_Project.ipynb
d4ac4633ebd6440fa397b84f1bc94a3c.7z
Example_0x2706E87E_SECRET.asc
Help
Contents Search Related Topics
Bookmarks Index
Hexeditor
The hexeditor can be used to create, edit and view all types of files.
See also:
Hexeditor
Texteditor
More results:
Search for Hexeditor
unsaved004.txt
out007.bin
00 01 02 03 04 05 06 07
000:B0 34 B8 A5 4D FE 88 5C .4..M..\
008:E1 36 A1 3F 10 AD 12 21 .6.?...!
010:7F 3B 65 E6 3B 4A 14 BB .;e.;J..
018:7F 07 96 35 6F CC 8C 57 ...5o..W
020:9E 40 96 58 D4 31 05 AA .@.X.1..
028:FF 11 D5 A3 23 AD AF 22 ....#..”
030:59 E5 AF B5 AD A0 20 F0 Y..... .
038:8B 00 11 5E B8 C1 E4 0C ...^....
040:46 4D 82 B3 99 6C 80 6A FM...l.j
048:0C 5B 82 F6 AE EE F2 03 .[......
050:EC 1C BD 5C EA CB 84 D6 ...\....
058:15 FF F1 AC 87 03 00 30 0
060:B3 B0 1C 4E F3 C1 B7 62 ...N...b
068:77 30 8E D3 E0 24 B8 3C w0...$.<
070:E0 E3 75 84 7F 56 EA C2 ..u..V..
078:12 E7 CF 43 20 64 87 68 ...C d.h
080:B1 CE 26 D1 55 A1 4C 24 ..&.U.L$
088:E7 E8 44 60 F8 95 6B D3 ..D`..k.
090:B5 C5 93 07 54 EA 4F 85 ....T.O.
098:69 63 A8 D9 FD E2 7F 29 ic.....)
0A0:F8 75 43 F8 D8 49 26 F3 .uC..I&.
0A8:AF 7C B0 D7 B0 77 99 70 .|...w.p
0B0:8D 1B A7 37 9B 52 DC E6 ...7.R..
0B8:2D D9 7D 7A B1 DE 50 CA -.}z..P.
0C0:66 6C 4F 3B 5E 33 9C 15 flO;^3..
0C8:D1 67 8B 53 55 C3 99 E9 .g.SU...
0D0:B9 09 E1 FD 19 72 61 A1 .....ra.
0D8:EE 7D 4F CC 6D 8C 11 78 .}O.m..x
0E0:C3 D2 79 80 ED CB 37 0D ..y...7.
0E8:B0 78 C6 D8 3C 65 E7 92 .x..<e..
0F0:F3 CA AC 7E E8 7A A0 F6 ...~.z..
0F8:A7 E9 EB D1 92 19 34 CF ......4.
Overwrite
0xB0 = 176 = 10110000
0 (0x0) : 1184 (0x4A0)
Crypto Explorer
type filter text
Asymmetric
Classic
Hash
MAC
Random Number Generator
Signature
Symmetric
AES
ARC4/Spritz
Dragon
IDEA
LFSR
RC6
Algorithms Analysis Visuals Games

Practical-11

Implement the Identity-based Encryption (IBE). Use the email address of the recipient to generate the key for a destination.

Identity-based Encryption (IBE) is an alternative to Public key infrastructure and involves generating the encryption key from a piece of identity for the recipient. For example, we could use the email address of the recipient to generate the key for a destination.

For this we have some shared parameters with a trust center that both Bob and Alice trust. If Alice wants to send Bob an email, she takes the parameters from the trust center, and then uses Bob's email address to generate his public key.

Code for IBE is given below:

```
package Client;
import java.math.BigInteger;
public class Client {
public static void main(String args[]){
String message,id_sender="alice@home",temp,id_recipient="bob@home";
id_recipient = args[1];
id_sender = args[0];
message=args[2];
System.out.println("ID (sender): "+id_sender+","+" ID (recipient):"+id_recipient);
System.out.println("Message: "+message);
message = id_recipient+" has sent this message:\n"+message;
byte[] m = message.getBytes();
PKG pkg = new PKG();
BigInteger Public_key = pkg.get_public_key(id_recipient);
Public_key = pkg.get_public_key(id_sender);
BigInteger n = pkg.getn();
System.out.println("Recipient Public Key: " + Public_key);
System.out.println("\nEncrypted message:");
System.out.println(bytesToString((new BigInteger(m)).modPow(Public_key,
n).toByteArray()));
System.out.println("\n==== Server");
System.out.println("id_sender (recipient): "+id_recipient);
BigInteger Private_key =pkg.get_private_key(id_sender);
```

Practical-12

To study and work with KF SENSOR Intrusion Detection Tool. Setup a honeypot and monitor the honeypot on the network.

INTRODUCTION:

HONEY POT: A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. Generally, it consists of a computer, applications, and data that simulate the behavior of a real system that appears to be part of a network but is actually isolated and closely monitored. All communications with a honeypot are considered hostile, as there's no reason for legitimate users to access a honeypot. Viewing and logging this activity can provide an insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value. Honeypots can be classified based on their deployment (use/action) and based on their level of involvement.

Based on deployment, honeypots may be classified as:

1. Production honeypots
2. Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.

KF SENSOR: KFSensor is a Windows based honeypot Intrusion Detection System (IDS). It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and trojans. By acting as a decoy server it can divert attacks from critical systems and provide a higher level of information than can be achieved by using firewalls and NIDS alone. KFSensor is a system installed in a network in order to divert and study an attacker's behavior. This is a new technique that is very effective in detecting attacks.

The main feature of KFSensor is that every connection it receives is a suspect hence it results in very few false alerts. At the heart of KFSensor sits a powerful internet daemon service that is built to handle multiple ports and IP addresses. It is written to resist denial of service and buffer overflow attacks. Building on this flexibility KFSensor can respond to connections in a variety of ways, from simple port listening and basic services (such as echo), to complex simulations of standard system services. For the HTTP protocol KFSensor accurately simulates the way Microsoft's web server (IIS) responds to both valid and invalid requests. As well as being able to host a website it also handles complexities such as range requests and client side cache negotiations. This makes it extremely difficult for an attacker to fingerprint, or identify KFSensor as a honeypot.

PROCEDURE:

STEP-1: Download KF Sensor Evaluation Setup File from KF Sensor Website.

STEP-2: Install with License Agreement and appropriate directory path.

STEP-3: Reboot the Computer now. The KF Sensor automatically starts during windows boot.

STEP-4: Click Next to setup wizard.

STEP-5: Select all port classes to include and Click Next.

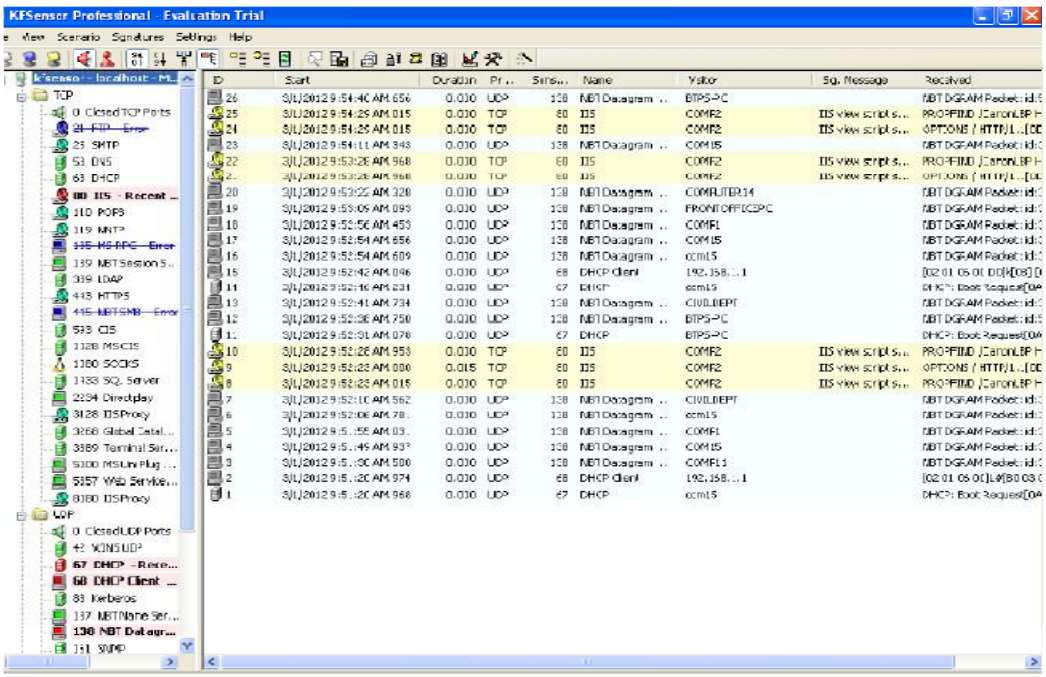
STEP-6: "Send the email and Send from email", enter the ID and Click Next.

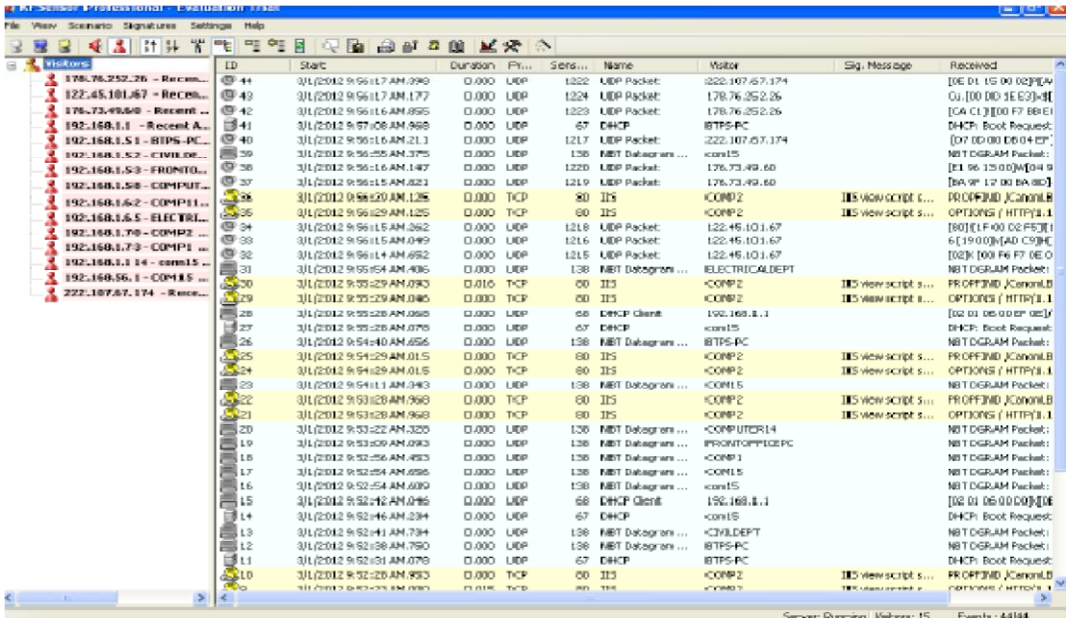
STEP-7: Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.

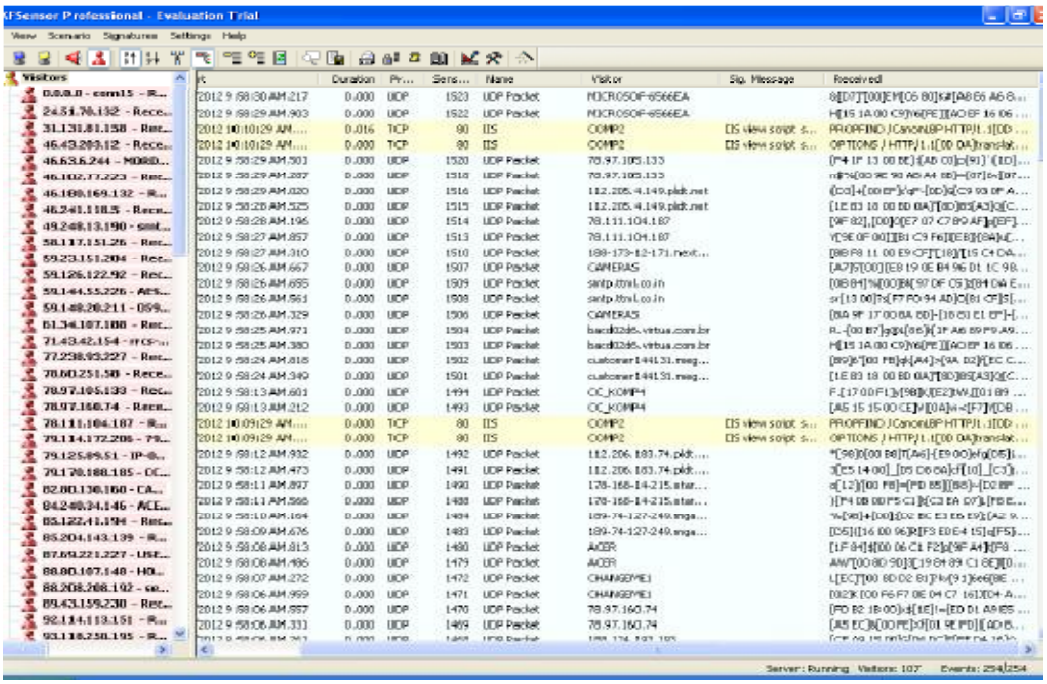
STEP-8: Select Install as System service and Click Next.

STEP-9: Click finish.

SCREENSHOTS:





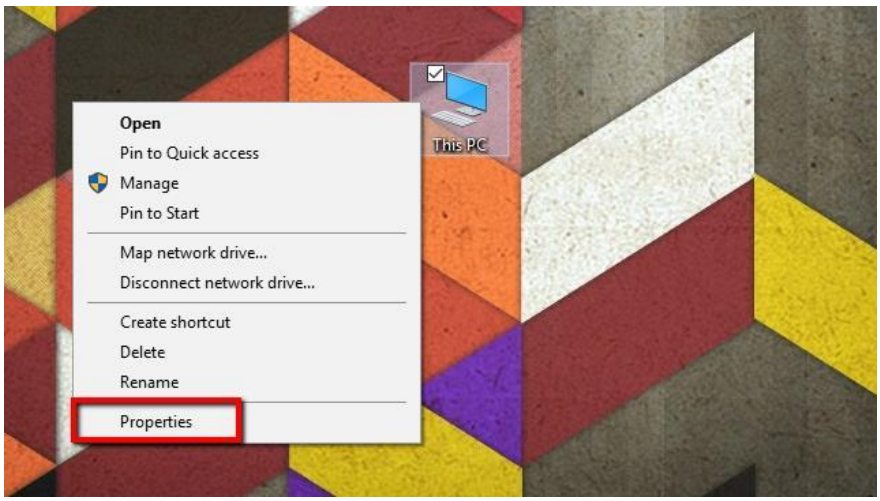


Practical – 13

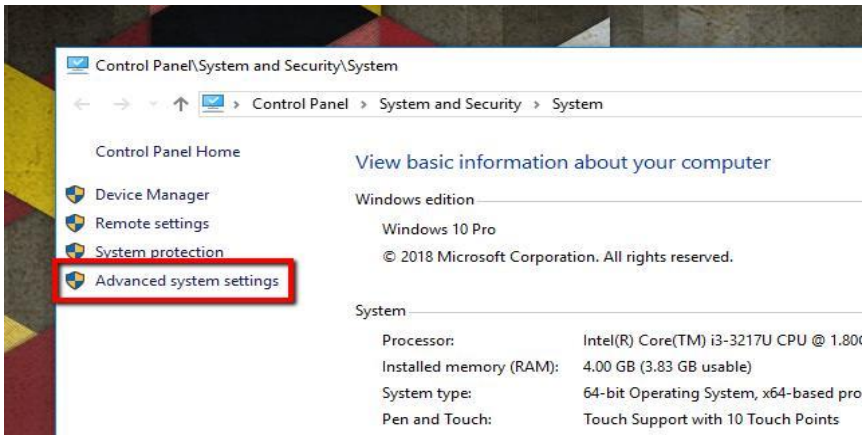
Configure Wireshark with a key to let you look inside encrypted SSL messages. You can read on the web how to do this. Once decrypted, you will be able to observe the HTTP protocol running on top of SSL, as well as the details of other SSL messages such as Alerts.

Steps:

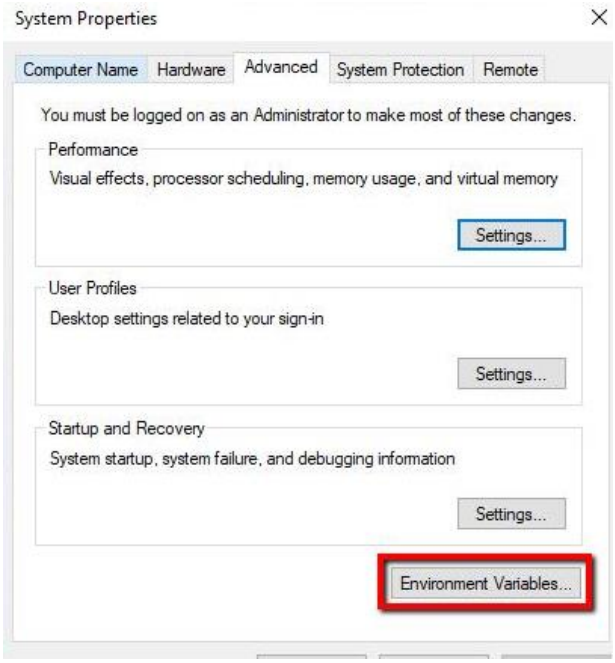
1. In **Windows systems**, you'll need to set an environment variable using the **Advanced system settings** utility. This variable, named **SSLKEYLOGFILE**, contains a path where the pre-master secret keys are stored.



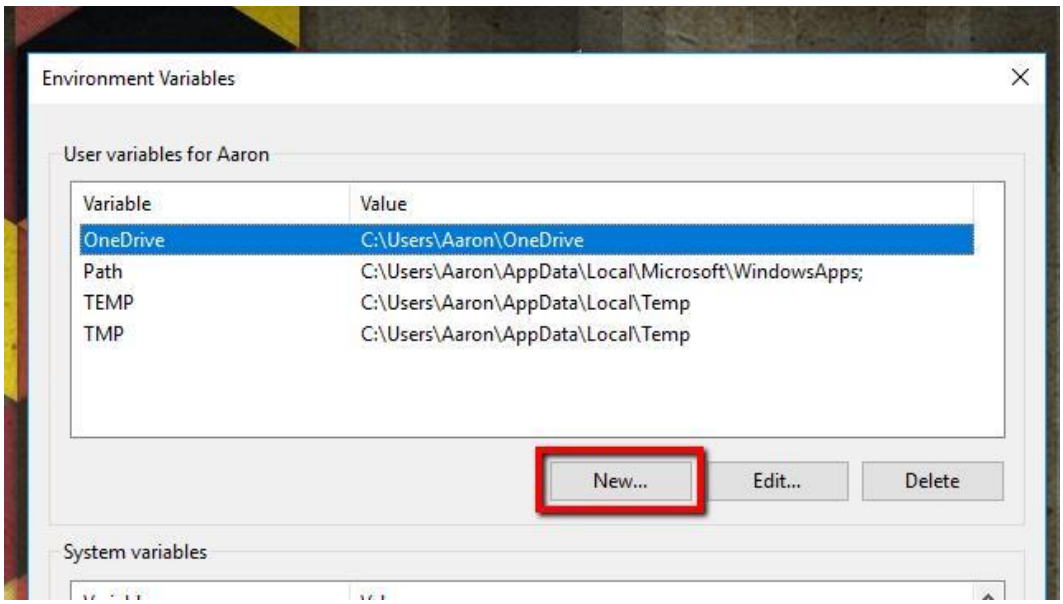
2. Start by right-clicking on My Computer and selecting Properties from the menu. The System menu will open.



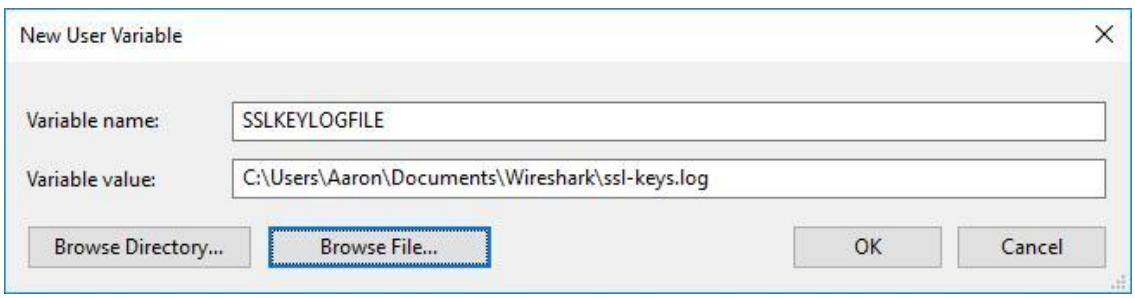
3. Next, click **Advanced system settings** on the list to the left. The **System Properties** window will open.



4. On the **Advanced** tab, click the **Environment Variables** button.



5. Click the **New…** button under **User variables**. You can also create the variable under **System variables** if you'd like to log SSL keys for every user on the system, but I prefer to keep it confined to my profile.



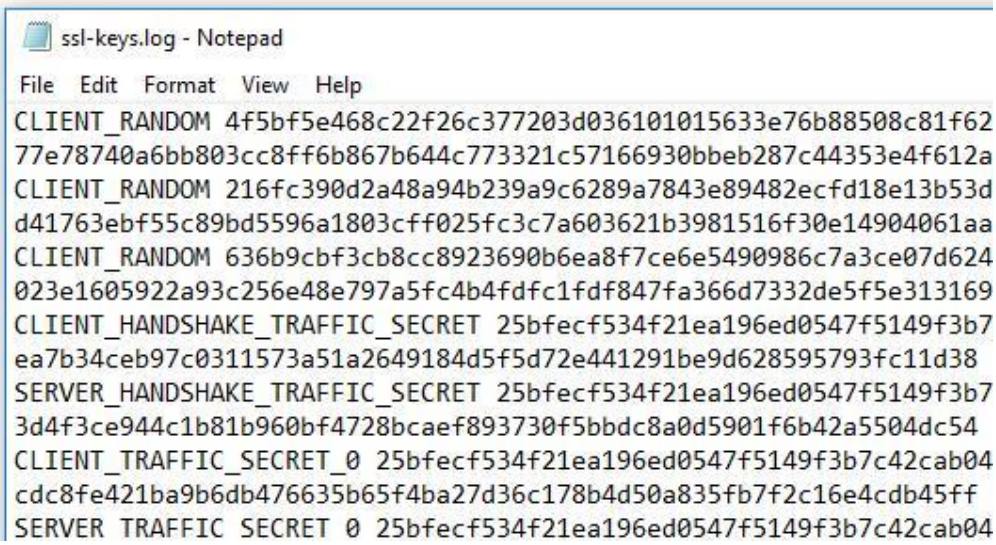
6. Under **Variable name**, type the following:

```
SSLKEYLOGFILE
```

In the **Variable value** field, type a path to the log file. You can also click the **Browse file…** button and specify the path using the file picker.

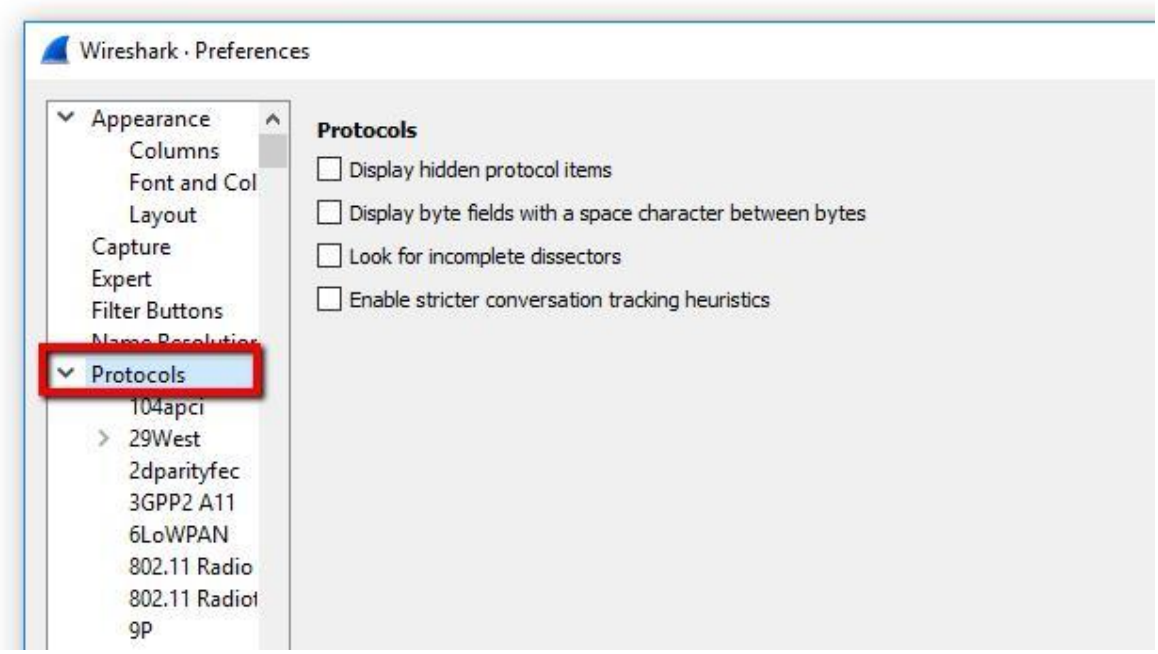
7. Before you launch Wireshark and configure it to decrypt SSL using a pre-master key, you should start your browser and confirm that the log file is being used.

8. To populate the log, it's important that you visit a site that has SSL enabled. One of the biggest benefits of using a pre-master shared key is **you don't need access to the server** to decrypt SSL.

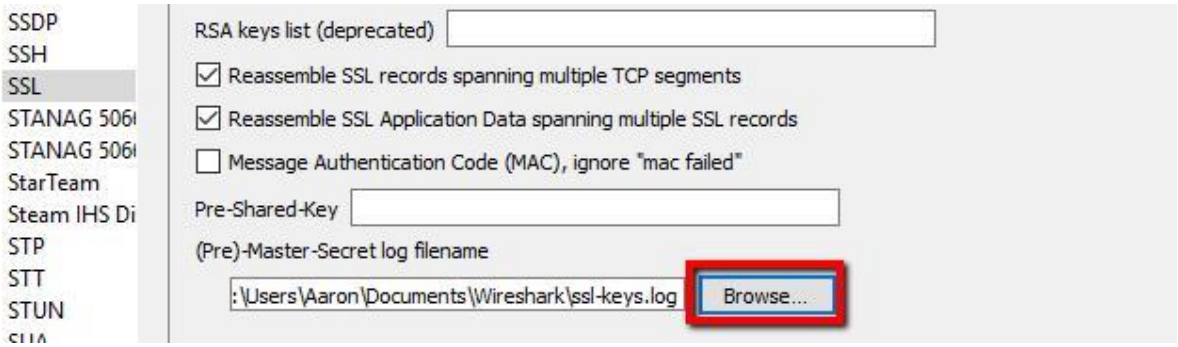


9. After you've visited a SSL-enabled website, check the file for data. In **Windows**, you can use **Notepad**. After you've confirmed that your browser is logging pre-master keys in the location you selected, you can configure Wireshark to use those keys to decrypt SSL.

10. Once your browser is logging pre-master keys, it's time to configure Wireshark to use those logs to decrypt SSL.



11. Open Wireshark and click **Edit**, then **Preferences**. The **Preferences** dialog will open, and on the left, you'll see a list of items. Expand **Protocols**, scroll down, then click **SSL**.

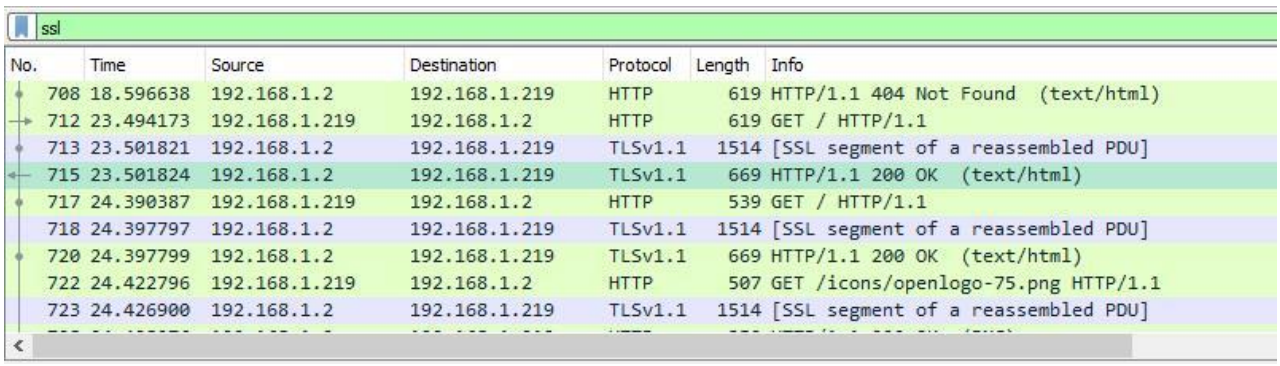


12. In the list of options for the SSL protocol, you'll see an entry for **(Pre)-Master-Secret log filename**. Browse to the log file you set up in the previous step, or just paste the path. When you've finished setting the **(Pre)-Master-Secret log filename**, click **OK** and return to Wireshark.

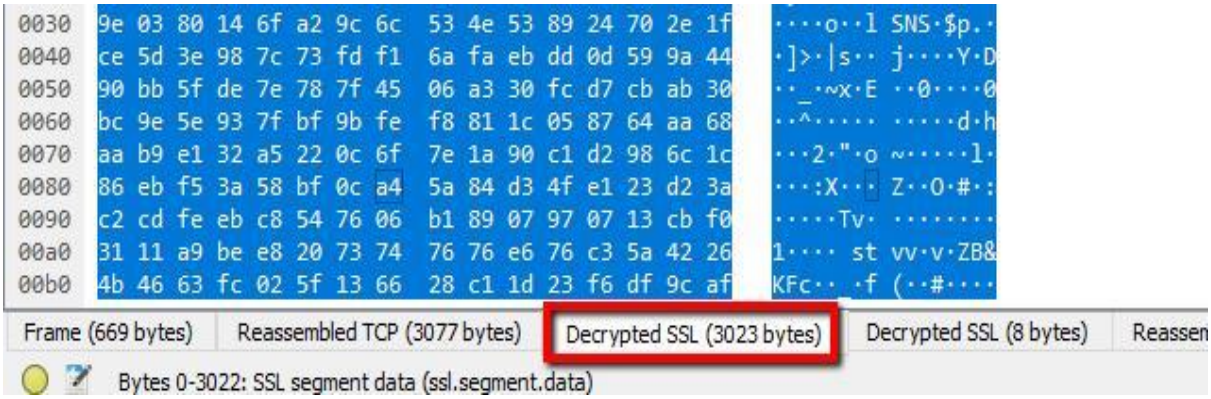
13. The final step is to capture a test session and make sure that Wireshark decrypts SSL successfully.

    - Start an unfiltered capture session, minimize it, and open your browser.
    - Visit a secure site to generate data, and optionally set a display filter of 'ssl' to minimize the session noise.
    - Click on any frame containing encrypted data.

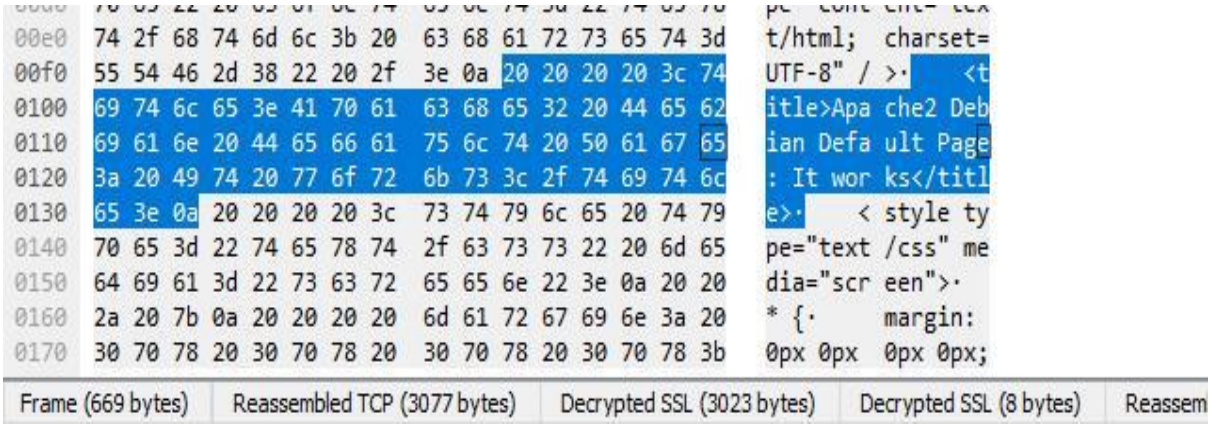
Here, we will select one that contains HTTP traffic with text/HTML encoding, since we would like to see the source code the web server is sending to the browser. But any encrypted transmissions that use a pre-master secret or private key will work with this method. That includes all data utilizing Perfect Forward Encryption (PFE) through Diffie-Hellman or comparable key exchanges.



14. Once you've selected an encrypted data frame, look at the **Packet byte view**, and specifically the tabs underneath the view. You should see an entry for **Decrypted SSL** data, among others.



15. You'll notice that the session still looks like it's full of garbage, and no HTML is visible. That's because the web server (and most Apache servers) use GZIP compression by default.



16. When you click the **Uncompressed entity body** tab, which only shows up in this case with SSL decryption enabled, you can view the source code of the site. For instance, here's the title element of the default Apache page in plaintext.

Practical – 14

To build a Trojan and know the harmness of the trojan malwares in a computer system. When the trojan code executes, it will open MS-Paint, Notepad, Command Prompt, Explorer, calculator, infinitely. Note: Use VMware to perform this experiment.

Theory:

- ✓ It is a code that is malicious in nature and has the capacity to take control of the computer. It is designed to steal, damage, or do some harmful actions on the computer. It tries to deceive the user to load and execute the files on the device. After it executes, this allows cybercriminals to perform many actions on the user's computer like deleting data from files, modifying data from files, and more. Now like many viruses or worms, Trojan Horse does not have the ability to replicate itself.

- ✓ **For example:**
  There was a Trojan that disguised itself as a game. Many users have downloaded this game and that secretly turned into a self-replicating virus.

- ✓ The game was a simple theme-based game, but it started to back up all the files on the drive where the user would access them.

- ✓ The Trojan turned out to be harmless, and it was easy for them to fix. So this was identified as Trojan because it did not disclose the virus.

- ✓ Many people have been infected by Trojans without realizing it. This type of Trojans is called Direct-Action-Trojans. It can't spread to any user because when a virus infects the system show some indications that it has been affected by the virus.

- ✓ For example: there is a direct action Trojan name Js. ExitW. It can be downloaded from many malicious sites. The effect of the Js. ExitW is to make the computer fall into a never-ending loop of start and shutdown. The Trojan does not do any damage which could be considered dangerous. But we should be aware that there are many Trojans that are far more dangerous.

Some features of the Trojan horse are as follows :

- It steals information like a password and more.
- It can be used to allow remote access to a computer.
- It can be used to delete data and more on the user's computers.

How Trojans are used?

There are many ways that it can be used :

1. **Spy** –
   Some Trojans act as spyware. It is designed to take the data from the victim like social networking(username and passwords), credit card details, and more.
2. **Creating backdoors** –
   The Trojan makes some changes in the system or the device of the victim, So this

is done to let other malware or any cyber criminals get into your device or the system.

3. **Zombie** – There are many times that the hacker is not at all interested in the victim's computer, but they want to use it under their control.

Now there are many Trojans which is designed to perform specific functions. Some of them are: –

1. **Trojan-Banker** – It is designed to steal the account data for online banking, credit and debit cards, etc.
2. **Trojan_Downloader** – It is designed to download many malicious files like the new versions of Trojan and Adware into the computer of the victims.
3. **Trojan-Dropper** – It is designed to prevent the detection of malicious files in the system. It can be used by hackers for installing Trojans or viruses on the victim's computers.
4. **Trojan-GameThief** – It is designed to steal data from Online Gamers.
5. **Trojan-I's** – It is designed to steal the data of login and passwords like: -a. skype b. yahoo pager and more.

Other Trojans can also be used like: -Trojan-notifier, Trojan-clicker, and more.

Indications that the system has been affected by the virus:

- First, the system or the device where it has been affected will be slow.
- The user will experience the files to be opening much slower.
- The user can also experience a direct shutdown of the pc.

Disadvantages of the Trojan Horse:

- It can't manifest by itself. It requires the implementation of the .exe files.
- It remains undetected and starts its execution when the user is doing any online transaction activity.

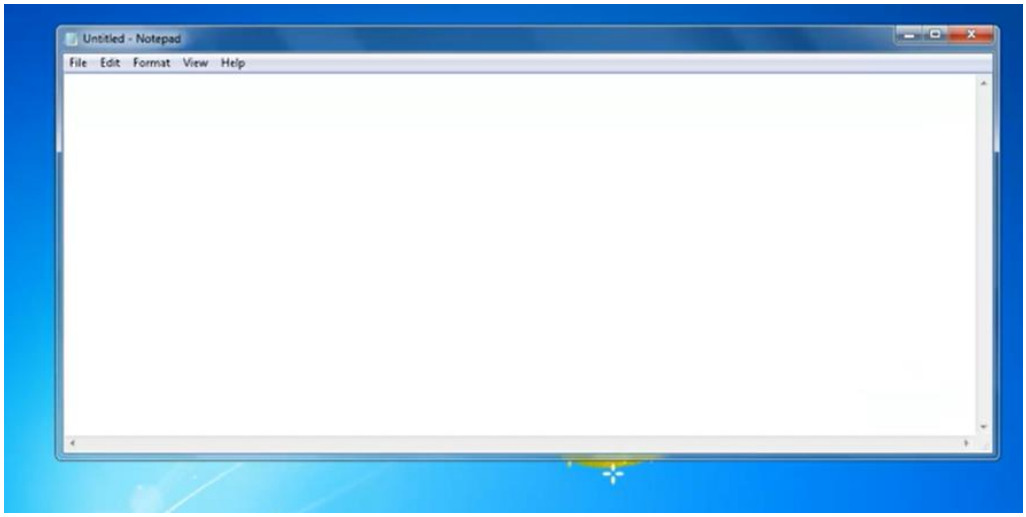
1. Do not open any attachment that has been sent from an unknown use.

The most common method:

The user must install the anti-virus program. This anti-virus program has the capacity to detect those files which are affected by a virus.

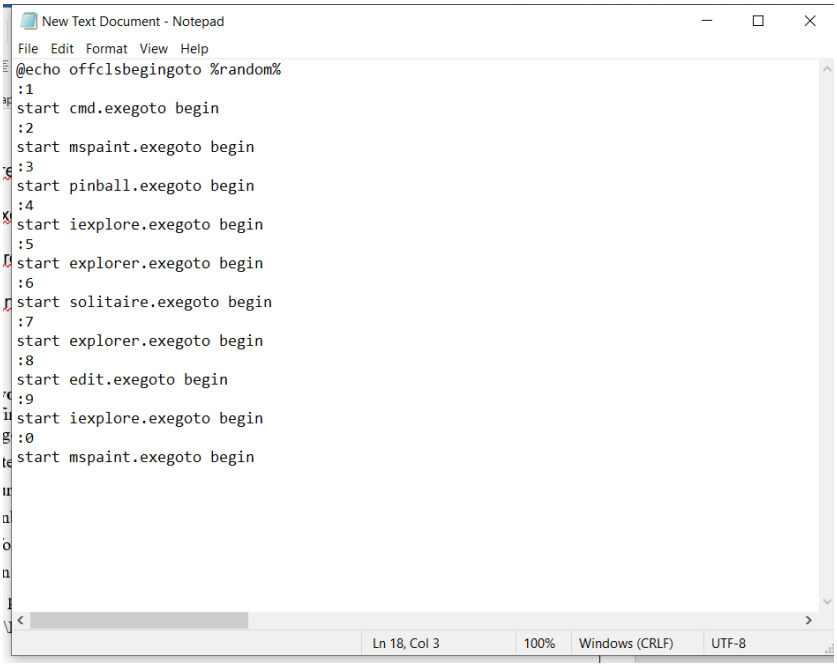
Making a Trojan

- **Open Notepad.** This trojan uses Notepad commands to cause the computer to randomly open programs until the batch file is disabled or the computer crashes. To make this E-bomb, you just need to copy and paste the pre-written commands in this section. However, note that this may not work on all computers.

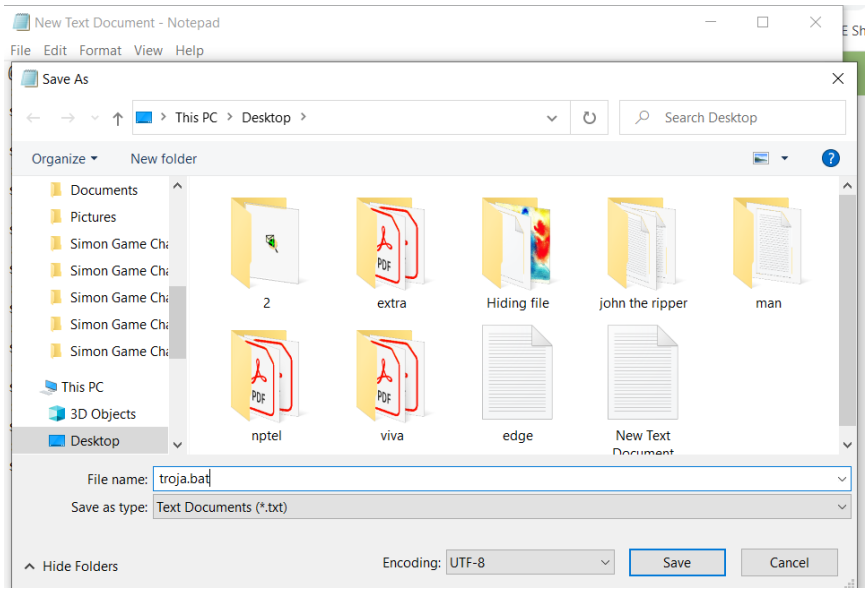


- **Copy and paste the following commands:**

```
@echo offclsbegingoto %random%
:1
start cmd.exegoto begin
:2
start mspaint.exegoto begin
:3
start pinball.exegoto begin
:4
start iexplore.exegoto begin
:5
start explorer.exegoto begin
:6
start solitaire.exegoto begin
:7
start explorer.exegoto begin
:8
start edit.exegoto begin
:9
start iexplore.exegoto begin
:0
start mspaint.exegoto begin
```



- **Modify as you see fit.** This program randomly opens the programs listed after each "start" indefinitely. You may notice some of these programs have been repeated. Feel free to change the programs listed to whichever ones you would like.
    - Note that some of the program names listed above may not be accurate for your specific machine - for instance, some computers won't contain "pinball.exe" You may want to double-check to see if each program exists before executing your E-bomb.
    - If in doubt about the name of a specific program, remember that its precise file pathway is also valid. For instance, "iexplore.exe" can also be written as "C:\Program Files\Internet Explorer".



- **Save as a batch file, then run the file (if you dare).** Save the file using the ".bat" extension (being sure to select "All files" from the "Save as type:" menu) When you trick the user into opening it, it will begin opening seemingly randomly programs on the computer without stopping.
    - For a bit of fun, try replacing one of the 'start *.exe' commands with the path of a notepad file on the person's computer or something like that. Use the following command:edit (path to file)goto beginThis will open their file in a DOS-based text editor, making it look like a hacker is reading their personal documents. Try it!

- **Learn the meaning of commands to be able to modify your Trojan.** Even if you do not understand how they work, trojan can be great pranks, but you'll be able to get even more enjoyment out of them if you know exactly what's happening when one is run. As a bonus, once you understand how E-bombs work, you can start writing your own! Here is a list of the commands we used in this article, along with a brief description of each:
    - @echo off - Turns command prompt comments off
    - cls - Clears the command prompt screen. This just makes the command prompt appear neater.
    - goto - Go to whatever *flag* you specify immediately after "goto".
    - %random% - A windows variable that randomly generates a number between (and including) 0-9.
    - :(number, letter, or word) - A flag. "Goto" commands send the program to whatever flag they specify.
    - Note: in our example above, we have 10 flags. If we leave a number out, the program should close when %random% generates that number.

Practical-15

Implement a code to simulate buffer overflow attack.

THEORY :

A buffer is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

In a buffer-overflow attack, the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user; for example, the data could trigger a response that damages files, changes data or unveils private information.

Attacker would use a buffer-overflow exploit to take advantage of a program that is waiting on a user's input. There are two types of buffer overflows: stack-based and heap-based. Heap-based, which are difficult to execute and the least common of the two, attack an application by flooding the memory space reserved for a program. Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack memory space used to store user input.

Let us study some real program examples that show the danger of such situations based on the C.

In the examples, we do not implement any malicious code injection but just to show that the buffer can be overflow. Modern compilers normally provide overflow checking option during the compile/link time but during the run time it is quite difficult to check this problem without any extra protection mechanism such as using exception handling.

CODE :

```c
// A C program to demonstrate buffer overflow
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{

    // Reserve 5 byte of buffer plus the terminating NULL.
    // should allocate 8 bytes = 2 double words,
    // To overflow, need more than 8 bytes...
    char buffer[5]; // If more than 8 characters input
                    // by user, there will be access
                    // violation, segmentation fault

    // a prompt how to execute the program...
    if (argc < 2)
    {
            printf("strcpy() NOT executed....\n");
            printf("Syntax: %s <characters>\n", argv[0]);
```

```
            exit(0);
    }

    // copy the user input to mybuffer, without any
    // bound checking a secure version is strcpy_s()
    strcpy(buffer, argv[1]);
    printf("buffer content= %s\n", buffer);

    // you may want to try strcpy_s()
    printf("strcpy() executed...\n");

    return 0;
}
```

Compile this program in Linux and for output use command output_file

INPUT

```
Input  : 12345678 (8 bytes), the program run smoothly.
```

```
Input : 123456789 (9 bytes)
"Segmentation fault" message will be displayed and the program
terminates.
```

The vulnerability exists because the buffer could be overflowed if the user input (argv[1]) bigger than 8 bytes. Why 8 bytes? For 32 bit (4 bytes) system, we must fill up a double word (32 bits) memory. Character (char) size is 1 byte, so if we request buffer with 5 bytes, the system will allocate 2 double words (8 bytes). That is why when you input more than 8 bytes; the mybuffer will be over flowed

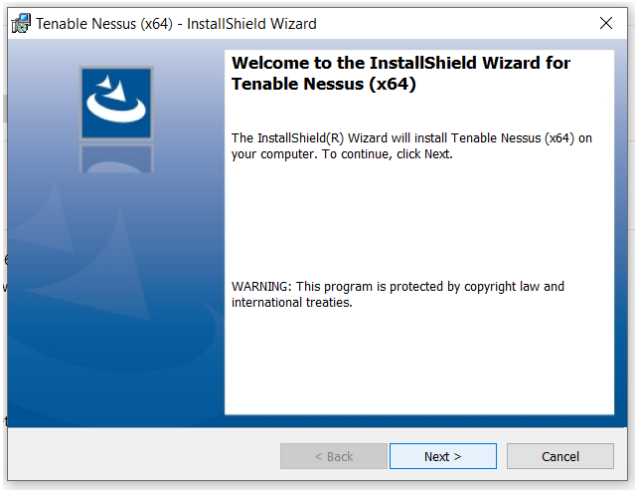
Similar standard functions that are technically less vulnerable, such as strncpy(), strncat(), and memcpy(), do exist. But the problem with these functions is that it is the programmer responsibility to assert the size of the buffer, not the compiler.

Every C/C++ coder or programmer must know the buffer overflow problem before they do the coding. A lot of bugs generated, in most cases can be exploited as a result of buffer overflow.

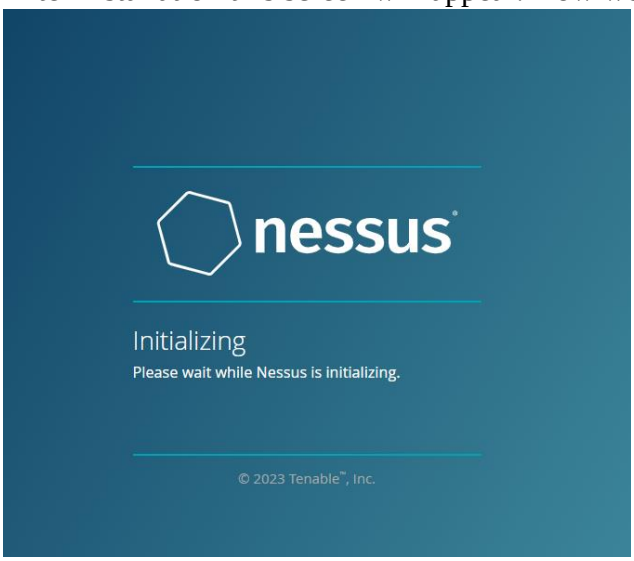
PRACTICAL 16

Use the Nessus tool to scan the network for vulnerabilities.

- Install the Nessus tool wizard.

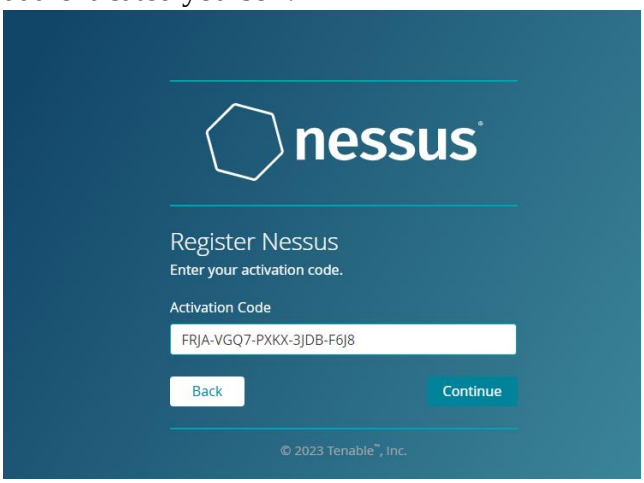


- After installation this screen will appear. Now we must enter details.

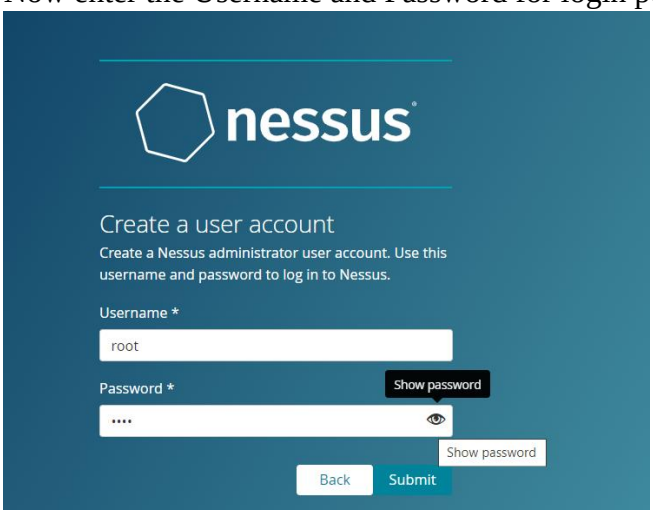


- Here select the essentials version. Essential version has limited functionality and it is limited to 16 IP addresses only.

- Enter the activation code which was sent to your mail ID when you authenticated yourself.

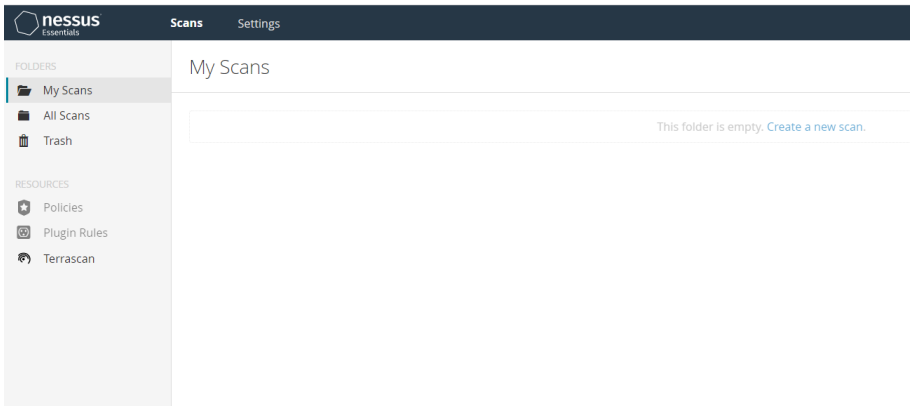


- Now enter the Username and Password for login purpose.



- The final process for downloading plugins start now.

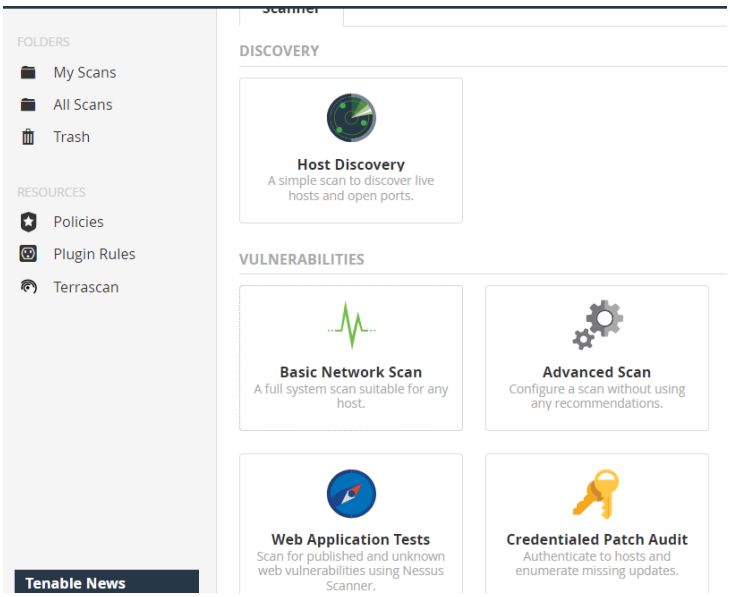
- After you have downloaded all the plugins this kind of screen appears.



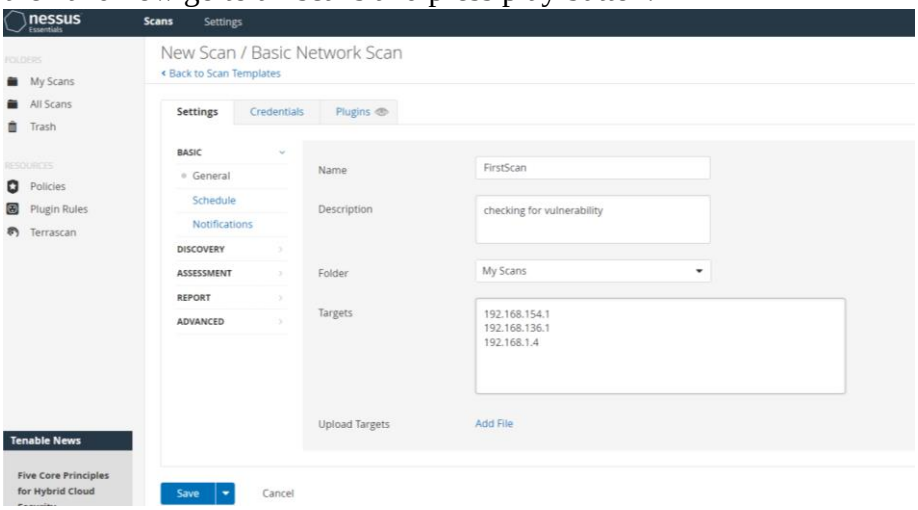
- Go to command prompt and type ipconfig. It will provide you with all the IP addresses you need in this process.

```
Ethernet adapter VMware Network Adapter VMnet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::6439:6f42:2b4d:b875%19
   IPv4 Address. . . . . . . . . . . : 192.168.154.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter VMware Network Adapter VMnet8:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::9a1e:6612:b003:da53%3
   IPv4 Address. . . . . . . . . . . : 192.168.136.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::8c28:7d91:186a:4c1f%5
   IPv4 Address. . . . . . . . . . . : 192.168.1.4
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%5
                                       192.168.1.1
```

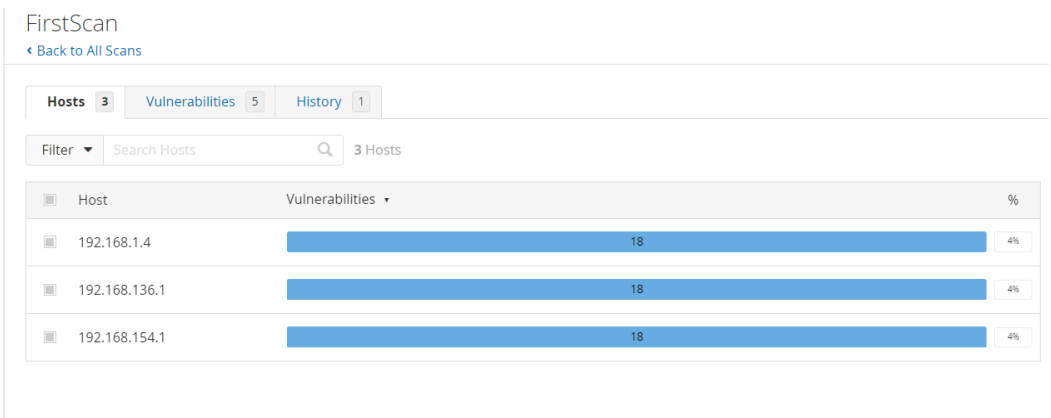
- Now click on new scan and select Basic Network Scan option.



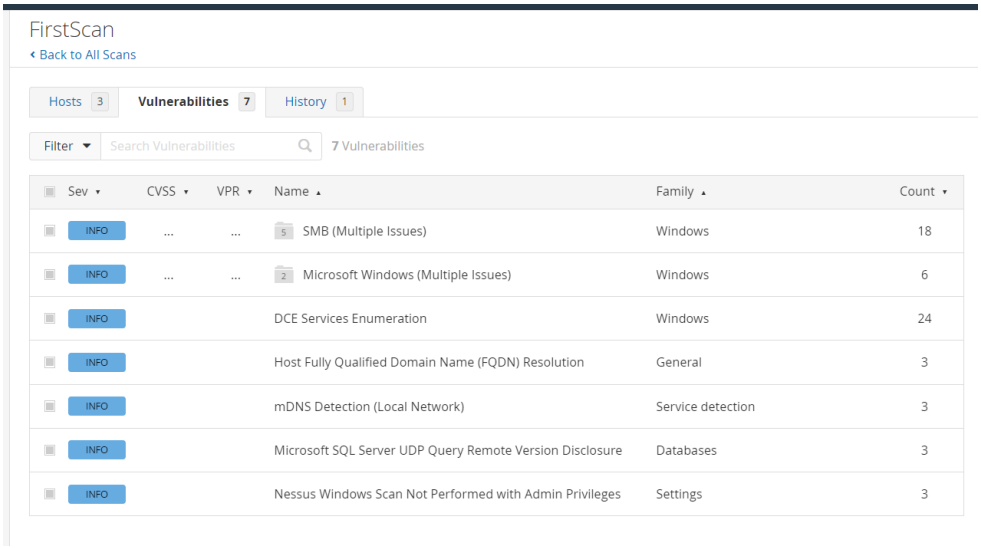
- Enter the Scan related details and the IP addresses you wish to scan. Click save then and now go to all scans and press play button.



- Now you can view all the hosts and corresponding vulnerabilities.



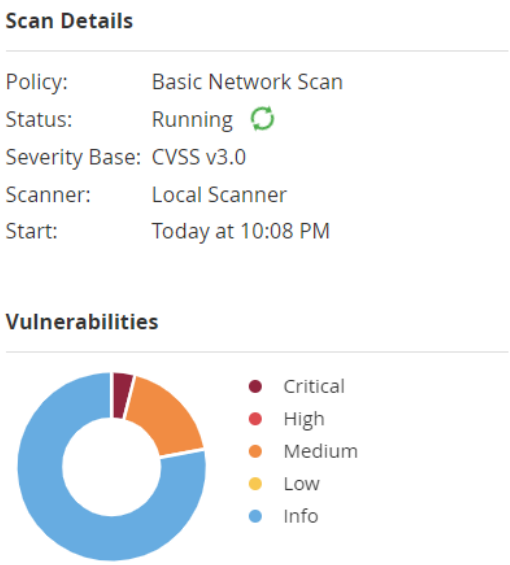
- Click on any vulnerability to view it.



- After some time, you can now notice CVSS values assigned to the vulnerabilities on the scale of 1 to 10.

Sev	CVSS	VPR	Name	Family	Count
CRITICAL	10.0 *		Microsoft Windows/Exchange SMTP DNS Lookup Overflo...	SMTP problems	3
MEDIUM	6.5		Echo Service Detection	Service detection	6
MEDIUM	6.5		Unencrypted Telnet Server	Misc.	3
MEDIUM	5.3		SMB Signing not required	Misc.	3
MEDIUM	5.0 *		Chargen UDP Service Remote DoS	Denial of Service	3
MEDIUM	5.0 *		MS10-024: Vulnerabilities in Microsoft Exchange and Wind...	SMTP problems	3
INFO	...	...	SMB (Multiple Issues)	Windows	21
INFO	...	...	Microsoft Windows (Multiple Issues)	Windows	6
INFO			DCE Services Enumeration	Windows	24
INFO			Authenticated Check : OS Name and Installed Package En...	Settings	3

- To the right of the screen you can see every detail regarding this.



➢ Click on the vulnerability and view its CVSS value the what has caused that.

MEDIUM Unencrypted Telnet Server

Description

The remote host is running a Telnet server over an unencrypted channel.

Using Telnet over an unencrypted channel is not recommended as logins, passwords, and commands are transferred in cleartext. This allows a remote, man-in-the-middle attacker to eavesdrop on a Telnet session to obtain credentials or other sensitive information and to modify traffic exchanged between a client and server.

SSH is preferred over Telnet since it protects credentials from eavesdropping and can tunnel additional data streams such as an X11 session.

Solution

Disable the Telnet service and use SSH instead.

Output

```
Nessus collected the following banner from the remote Telnet server :

------------------------------ snip ------------------------------
Welcome to Microsoft Telnet Service

login:
------------------------------ snip ------------------------------
```

To see debug logs, please visit individual host

Port ▲	Hosts
23 / tcp / telnet	192.168.1.4 192.168.136.1 192.168.154.1

➢ A 10 CVSS vulnerability can be dangerous. Try to eliminate all such vulnerability.

CRITICAL Microsoft Windows/Exchange SMTP DNS Lookup Overflow (885881)

Description

The remote host is running a version of Microsoft SMTP server which fails to validate DNS response data. An attacker can exploit this flaw to execute arbitrary code subject to the priviliges of the SMTP application server process.

Solution

Apply the bulletin referenced above.

See Also

https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2004/ms04-035

Output

```
No output recorded.
```

To see debug logs, please visit individual host

Port ▲	Hosts
25 / tcp	192.168.1.4 192.168.136.1 192.168.154.1