



NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

INFORMATION SECURITY

ABHISHEK KUMAR SAH
2019UIT3001
IT-1

INDEX

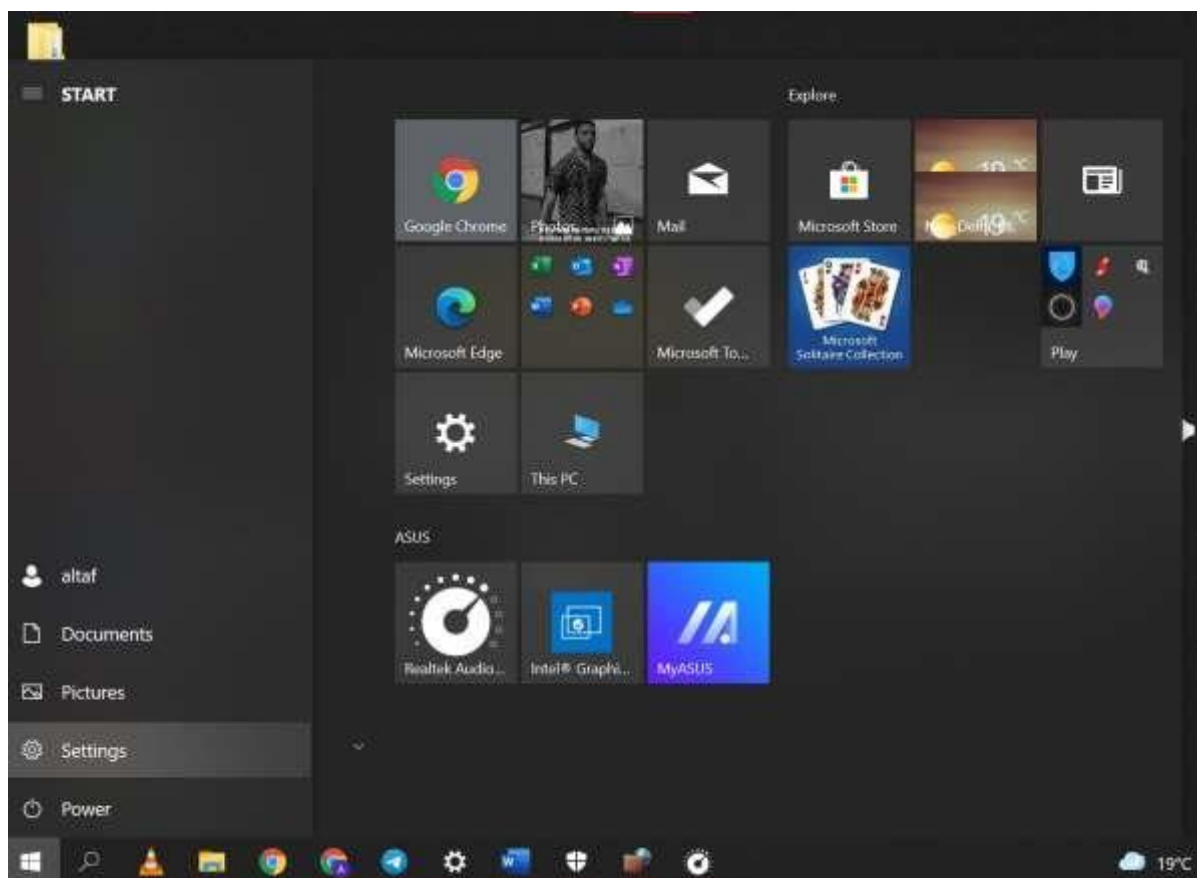
S. NO	PRACTICAL
1.	Study of the features of firewall in providing network security and to set Firewall security in windows. Students should know the following: a) Know how to setup and configure a firewall on Operating System. b) Know about the Windows Firewall with Advanced Security. c) Know the Connection Security Rules d) Know How to Start & Use the Windows Firewall with Advanced Security
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 Kloeopatra Certificate management software. Check encryption –decryption of an email sent to you.
3.	Implement MD5 Algorithm. Take all the constants from the Figure attached implement in Python
4.	Implement SHA-256 Algorithm. Consider all the Constants and Tables as given in the text book.
5.	Plot an elliptic curve over finite field. Check whether the points lies 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.	To work with Snort tool to demonstrate Intrusion Detection System. Download SNORT from snort.org.
16.	Implement a code to simulate buffer overflow attack.
17.	Use the Nessus tool to scan the network for vulnerabilities

STUDY OF THE FEATURES OF FIREWALL IN PROVIDING NETWORK SECURITY AND TO SET FIREWALL SECURITY IN WINDOWS.
STUDENTS SHOULD KNOW THE FOLLOWING:

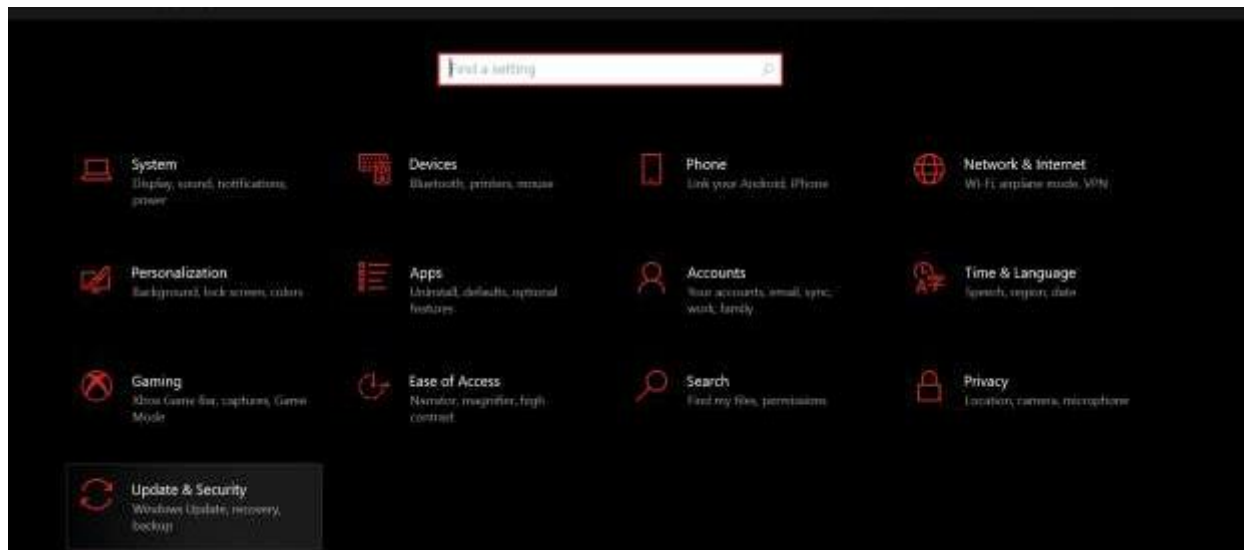
- a) Know how to setup and configure a firewall on Operating System.
- b) Know about the Windows Firewall with Advanced Security.
- c) Know the Connection Security Rules.
- d) Know How to Start & Use the Windows Firewall with Advanced Security.

- **How to step and configure a firewall.**

- 1) Click on start button in windows 10 and then open up settings.



2) Click on windows update and security.



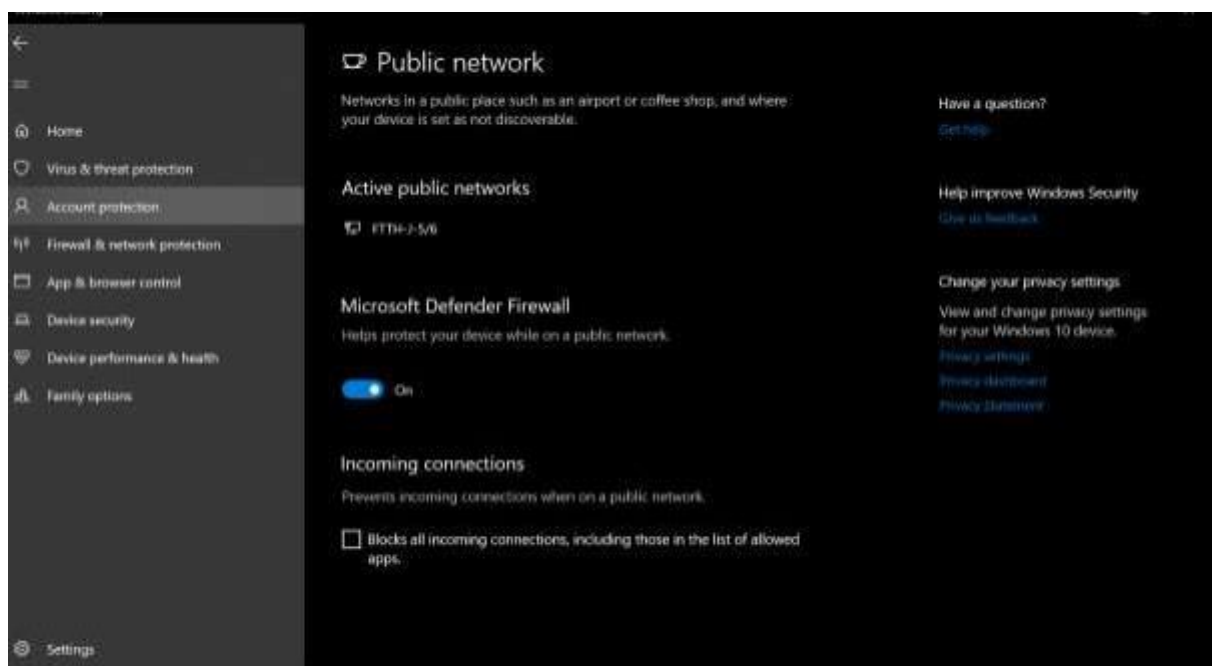
3) Click on windows security and then on Firewall and network protection.



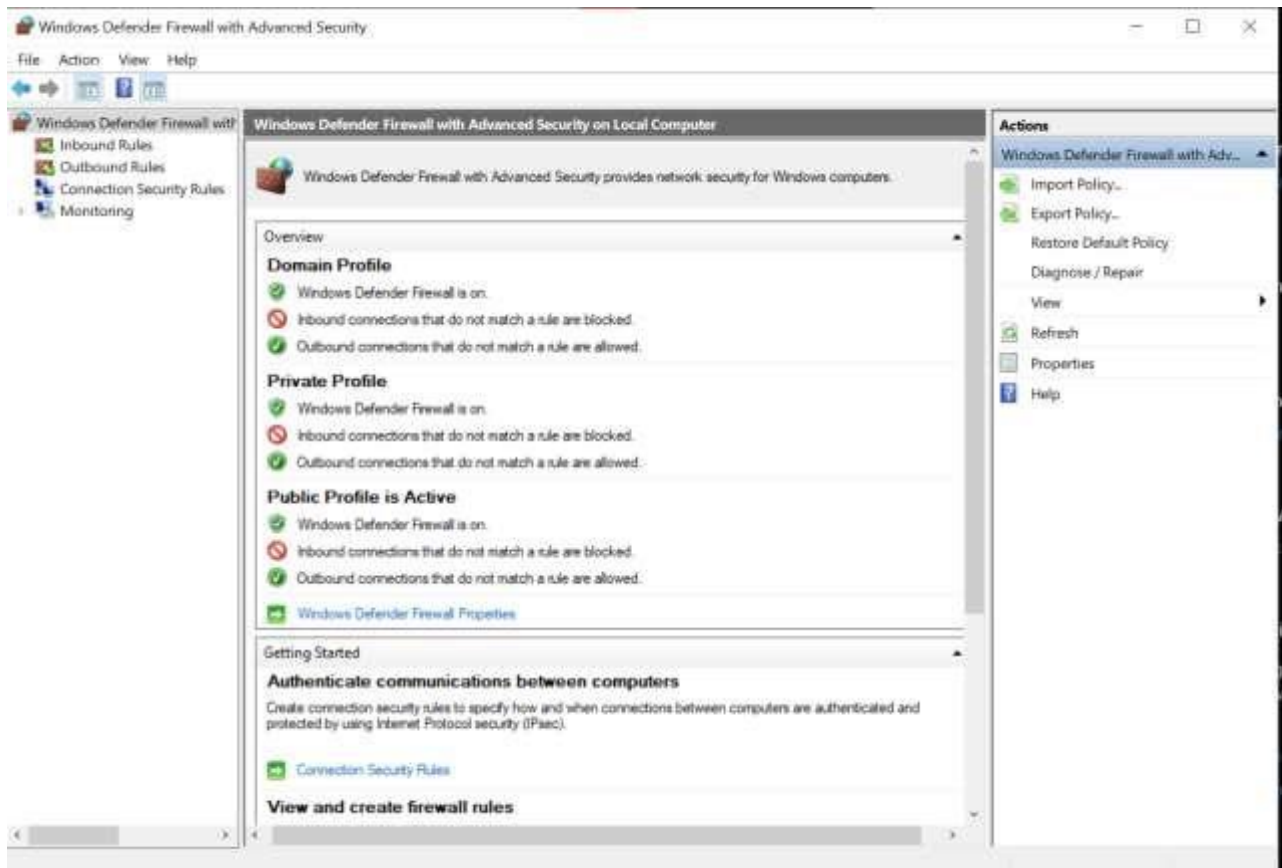
- 4) Open up firewall and network security and click on public network



- 5) Now to configure firewall and network: slide on the slider of Microsoft Defender Firewall.



- **Window Defender Firewall with Advanced settings.**

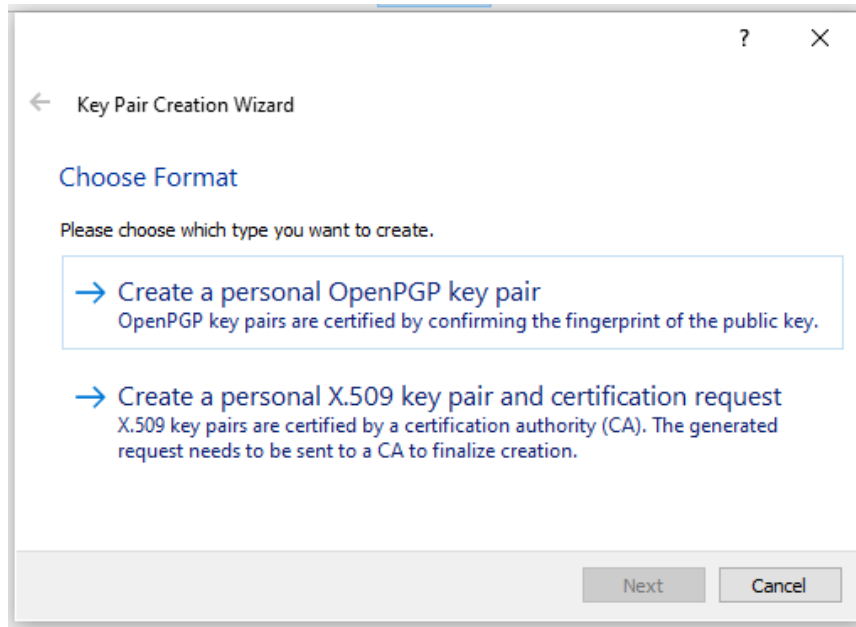


- **Connection Security Rules**

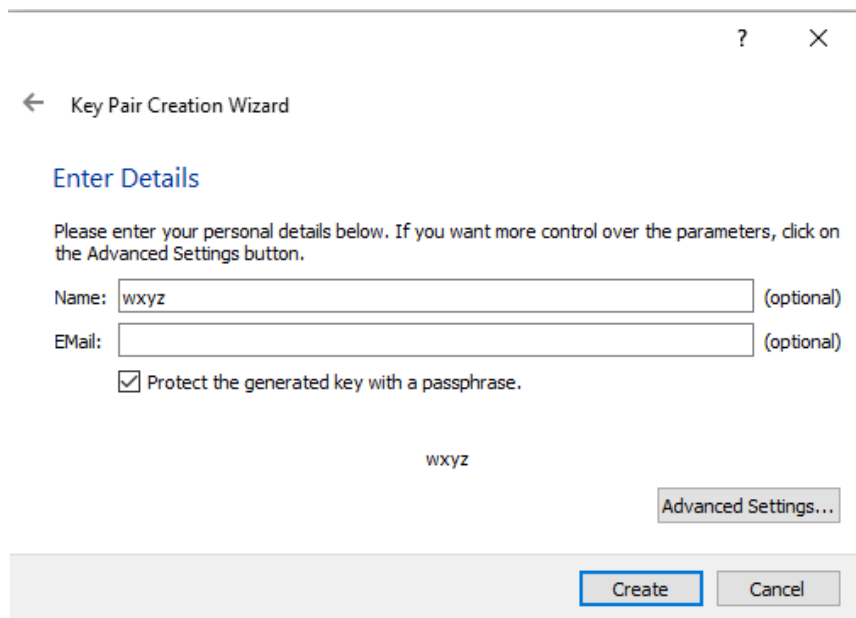
- Connection security involves the authentication of two computers before they begin communications and the securing of information sent between two computers. Windows Firewall with Advanced Security uses Internet Protocol security (IPsec) to achieve connection security by using key exchange, authentication, data integrity, and, optionally, data encryption.
- Connection security rules use IPsec to secure traffic while it crosses the network. We use connection security rules to specify that connections between two computers must be authenticated or encrypted. we have to create a firewall rule to allow network traffic protected by a connection security rule.

DEMONSTRATE HOW TO PROVIDE SECURE DATA STORAGE, SECURE DATA TRANSMISSION AND FOR CREATING DIGITAL SIGNATURES USING TOOL GNUPG.

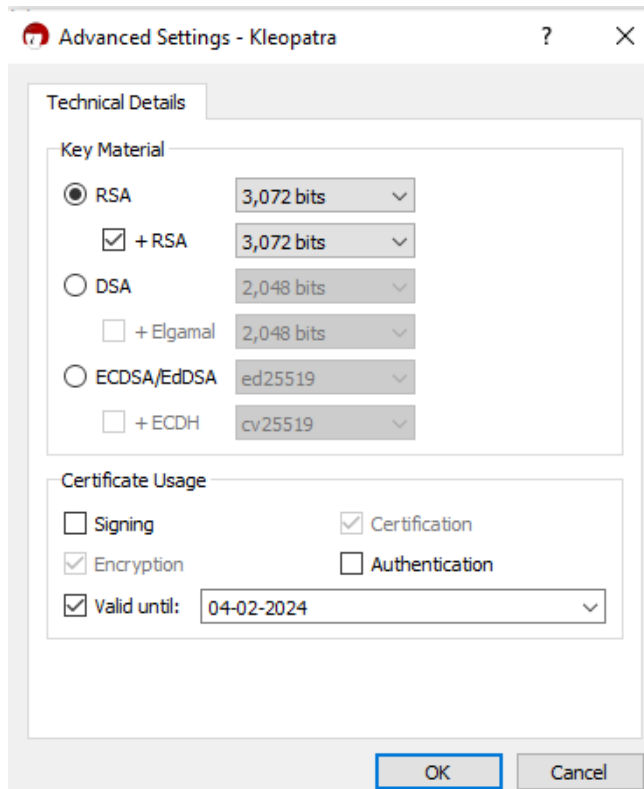
1. Click File->New Certificate:



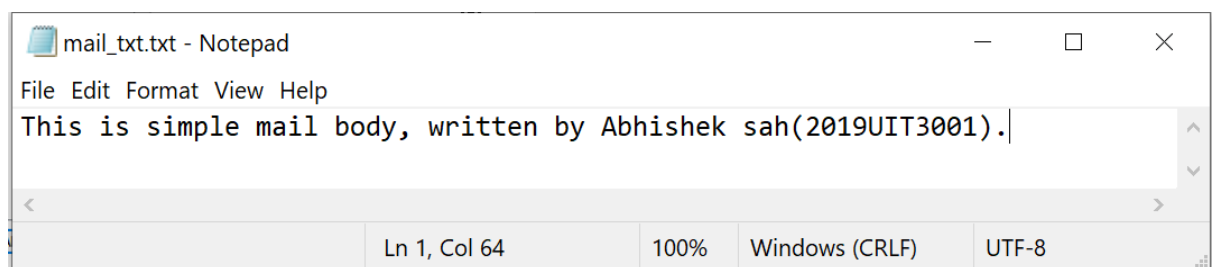
2. Choose and create a personal OpenPGP key pair. It will prompt to enter name and email.



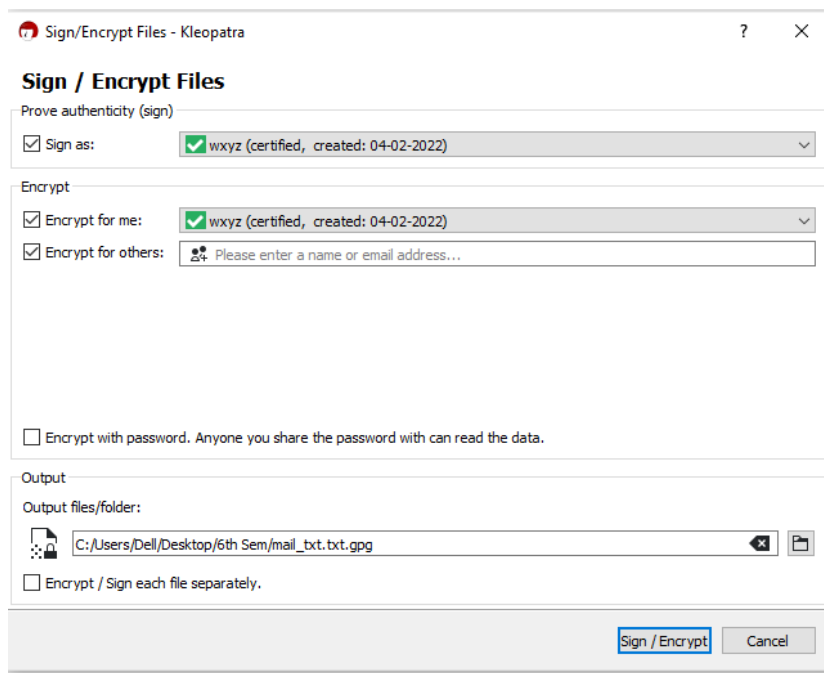
3. Click Advanced Settings and choose appropriate key and key size. After this step key will be created.



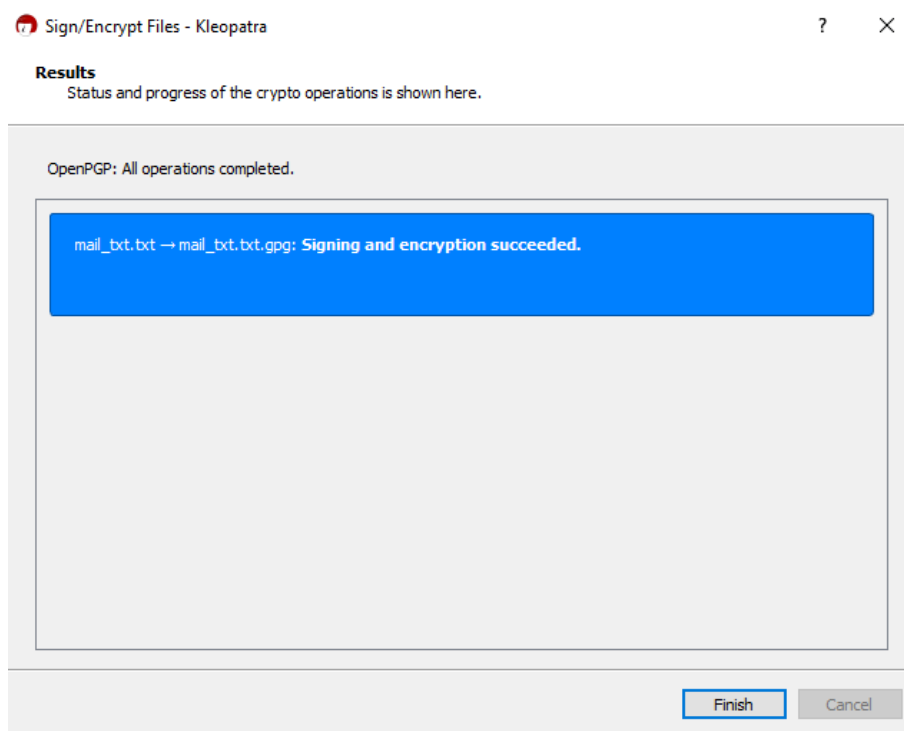
4. Create a text file mail_txt and add some content to it.



- Click on Sign/Encrypt Files and select the text file created in the step above. Select the keys and the path to store the encrypted file.



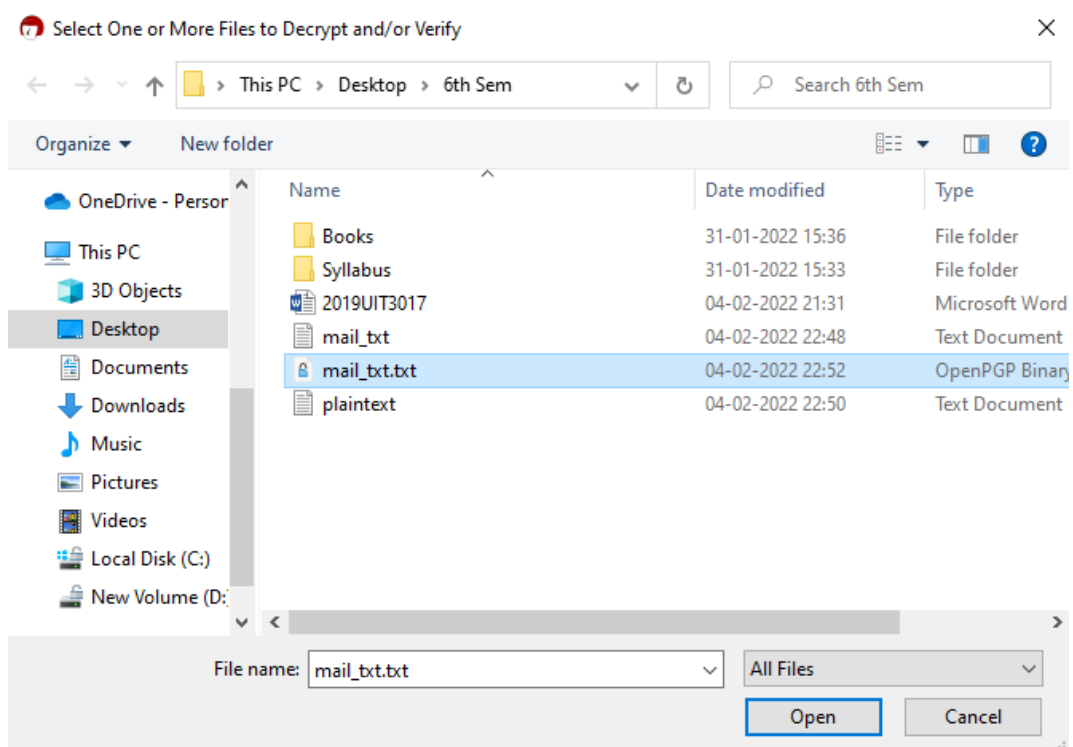
- Click on Sign/Encrypt and click on finish.



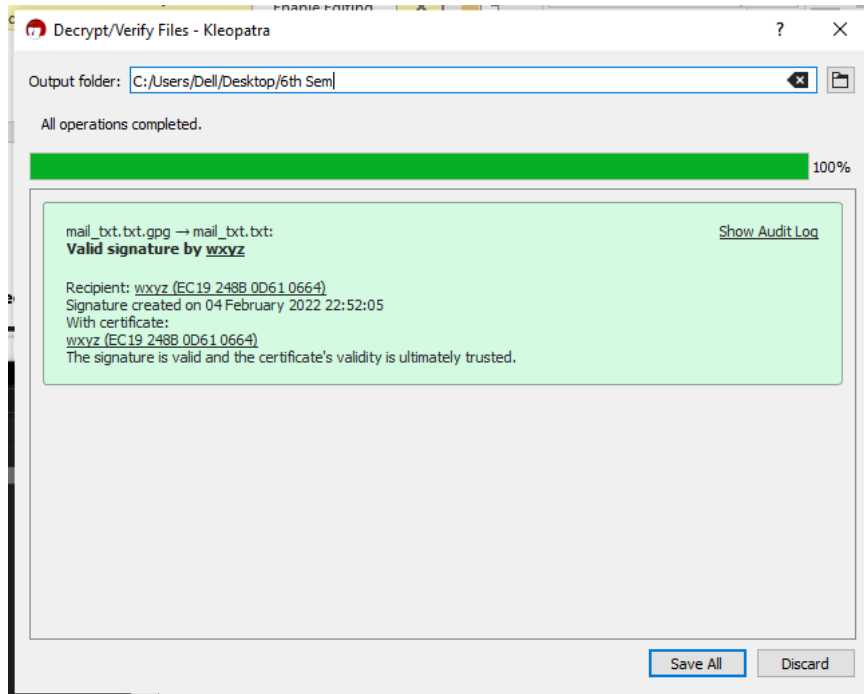
7. Now open the encrypted file to see the encrypted message.



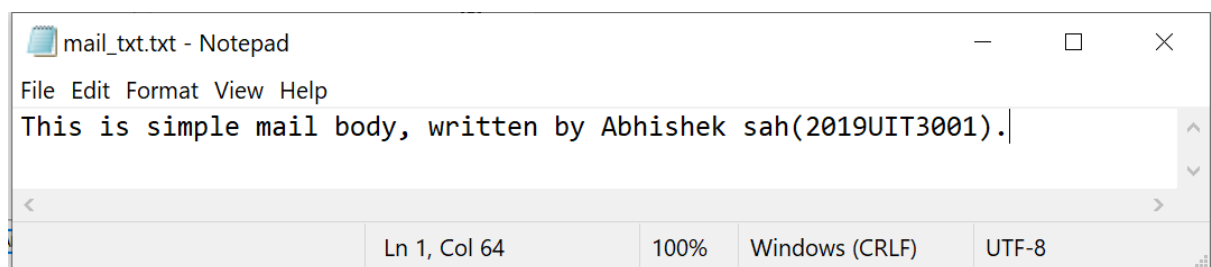
8. Now click on Decrypt/verify and select the encrypted file.



9. Select the path to store the decrypted file.



10. Open the decrypted will and it will have same content as the original file. So Encryption and Decryption is successful.



IMPLEMENT MD5 ALGORITHM

MD5.h

```
#ifndef md5_h
#define md5_h

#include <cstring>
#include <iostream>

class MD5
{
public:

    typedef unsigned int size_type; // must be 32bit

    MD5();
    MD5(const std::string& text);
    void update(const unsigned char *buf, size_type length);
    void update(const char *buf, size_type length);
    MD5& finalize();

    std::string hexdigest() const;
    friend std::ostream& operator<<(std::ostream&, MD5 md5);

private:

    void init();
    typedef unsigned char uint1; // 8bit
    typedef unsigned int uint4; // 32bit
    enum {blocksize = 64}; // VC6 won't eat a const static int here

    void transform(const uint1 block[blocksize]);
    static void decode(uint4 output[], const uint1 input[], size_type len);
    static void encode(uint1 output[], const uint4 input[], size_type len);

    bool finalized;

    uint1 buffer[blocksize];
    uint4 count[2]; // 64bit counter for number of bits (lo, hi)
    uint4 state[4]; // digest so far
    uint1 digest[16]; // the result
```

```

// low level logic operations
static inline uint4 F(uint4 x, uint4 y, uint4 z);
static inline uint4 G(uint4 x, uint4 y, uint4 z);
static inline uint4 H(uint4 x, uint4 y, uint4 z);
static inline uint4 I(uint4 x, uint4 y, uint4 z);
static inline uint4 rotate_left(uint4 x, int n);
static inline void FF(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);
static inline void GG(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);
static inline void HH(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);
static inline void II(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

};

std::string md5(const std::string str);

#endif /* md5_h */

```

Main.cpp

```

/* interface header */
#include "md5.h"

/* system implementation headers */
#include <cstdio>

// Constants for MD5Transform routine.
#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21

////////////////////////////////////

```

```

// F, G, H and I are basic MD5 functions.
inline MD5::uint4 MD5::F(uint4 x, uint4 y, uint4 z) {
    return x&y | ~x&z;
}

inline MD5::uint4 MD5::G(uint4 x, uint4 y, uint4 z) {
    return x&z | y&~z;
}

inline MD5::uint4 MD5::H(uint4 x, uint4 y, uint4 z) {
    return x^y^z;
}

inline MD5::uint4 MD5::I(uint4 x, uint4 y, uint4 z) {
    return y ^ (x | ~z);
}

// rotate_left rotates x left n bits.
inline MD5::uint4 MD5::rotate_left(uint4 x, int n) {
    return (x << n) | (x >> (32-n));
}

// FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.
// Rotation is separate from addition to prevent recomputation.
inline void MD5::FF(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {
    a = rotate_left(a+ F(b,c,d) + x + ac, s) + b;
}

inline void MD5::GG(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {
    a = rotate_left(a + G(b,c,d) + x + ac, s) + b;
}

inline void MD5::HH(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {
    a = rotate_left(a + H(b,c,d) + x + ac, s) + b;
}

inline void MD5::II(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {
    a = rotate_left(a + I(b,c,d) + x + ac, s) + b;
}

////////////////////////////////////

// default ctor, just initailize
MD5::MD5()
{
    init();
}

```

```

// nifty shortcut ctor, compute MD5 for string and finalize it right away
MD5::MD5(const std::string &text)
{
    init();
    update(text.c_str(), text.length());
    finalize();
}

////////////////////////////////////

void MD5::init()
{
    finalized=false;

    count[0] = 0;
    count[1] = 0;

    // load magic initialization constants.
    state[0] = 0x67452301;
    state[1] = 0xefcdab89;
    state[2] = 0x98badcfe;
    state[3] = 0x10325476;
}

////////////////////////////////////

// decodes input (unsigned char) into output (uint4). Assumes len is a multiple of 4.
void MD5::decode(uint4 output[], const uint1 input[], size_type len)
{
    for (unsigned int i = 0, j = 0; j < len; i++, j += 4)
        output[i] = ((uint4)input[j]) | (((uint4)input[j+1]) << 8) |
            (((uint4)input[j+2]) << 16) | (((uint4)input[j+3]) << 24);
}

////////////////////////////////////

// encodes input (uint4) into output (unsigned char). Assumes len is
// a multiple of 4.
void MD5::encode(uint1 output[], const uint4 input[], size_type len)
{
    for (size_type i = 0, j = 0; j < len; i++, j += 4) {
        output[j] = input[i] & 0xff;
        output[j+1] = (input[i] >> 8) & 0xff;
        output[j+2] = (input[i] >> 16) & 0xff;
        output[j+3] = (input[i] >> 24) & 0xff;
    }
}

```

```
////////////////////////////////////
```

```
// apply MD5 algo on a block
```

```
void MD5::transform(const uint1 block[blocksize])
```

```
{
```

```
    uint4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];
```

```
    decode (x, block, blocksize);
```

```
    /* Round 1 */
```

```
    FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
```

```
    FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
```

```
    FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
```

```
    FF (b, c, d, a, x[ 3], S14, 0xc1bdcee); /* 4 */
```

```
    FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
```

```
    FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
```

```
    FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
```

```
    FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
```

```
    FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
```

```
    FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
```

```
    FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
```

```
    FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
```

```
    FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
```

```
    FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
```

```
    FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
```

```
    FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */
```

```
    /* Round 2 */
```

```
    GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
```

```
    GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
```

```
    GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
```

```
    GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
```

```
    GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
```

```
    GG (d, a, b, c, x[10], S22, 0x2441453); /* 22 */
```

```
    GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
```

```
    GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
```

```
    GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
```

```
    GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
```

```
    GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
```

```
    GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
```

```
    GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
```

```
    GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
```

```
    GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
```

```
    GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */
```

```
    /* Round 3 */
```

```
    HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
```

```
    HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
```

```
    HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
```

```
    HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
```



```

HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
HH (c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
HH (b, c, d, a, x[10], S34, 0xbebfb7c0); /* 40 */
HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
HH (d, a, b, c, x[ 0], S32, 0xea127fa); /* 42 */
HH (c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
HH (b, c, d, a, x[ 6], S34, 0x4881d05); /* 44 */
HH (a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
HH (b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */

```

```

/* Round 4 */

```

```

II (a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
II (d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
II (b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
II (d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
II (c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
II (b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
II (a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
II (c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
II (a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
II (c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
II (b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */

```

```

state[0] += a;
state[1] += b;
state[2] += c;
state[3] += d;

```

```

// Zeroize sensitive information.

```

```

memset(x, 0, sizeof x);
}

```

```

////////////////////////////////////

```

```

// MD5 block update operation. Continues an MD5 message-digest
// operation, processing another message block

```

```

void MD5::update(const unsigned char input[], size_type length)
{

```

```

    // compute number of bytes mod 64
    size_type index = count[0] / 8 % blocksize;

```



```

};

if (!finalized) {

    // Save number of bits
    unsigned char bits[8];
    encode(bits, count, 8);

    // pad out to 56 mod 64.
    size_type index = count[0] / 8 % 64;
    size_type padLen = (index < 56) ? (56 - index) : (120 - index);
    update(padding, padLen);

    // Append length (before padding)
    update(bits, 8);

    // Store state in digest
    encode(digest, state, 16);

    // Zeroize sensitive information.
    memset(buffer, 0, sizeof buffer);
    memset(count, 0, sizeof count);

    finalized=true;
}

return *this;
}

////////////////////////////////////

// return hex representation of digest as string
std::string MD5::hexdigest() const
{
    if (!finalized)
        return "";

    char buf[33];
    for (int i=0; i<16; i++)
        sprintf(buf+i*2, "%02x", digest[i]);
    buf[32]=0;

    return std::string(buf);
}

std::ostream& operator<<(std::ostream& out, MD5 md5)
{
    return out << md5.hexdigest();
}

```

```
}


std::string md5(const std::string str)
{
    MD5 md5 = MD5(str);

    return md5.hexdigest();
}




using std::cout; using std::endl;

int main(int argc, char *argv[])
{
    cout << "MD5 of 'information': " << md5("information") << endl;
    return 0;
}
```

OUTPUT

338 lines | 

MD5 of 'information': bb3ccd5881d651448ded1dac904054ac
Program ended with exit code: 0

All Output ↕   

IMPLEMENT SHA256 ALGORITHM

SHA256.h

```
#ifndef SHA256_H
#define SHA256_H
#include <string>
using namespace std;

class SHA256
{
protected:
    typedef unsigned char uint8;
    typedef unsigned int uint32;
    typedef unsigned long long uint64;

    const static uint32 sha256_k[];
    static const unsigned int SHA224_256_BLOCK_SIZE = (512/8);
public:
    void init();
    void update(const unsigned char *message, unsigned int len);
    void final(unsigned char *digest);
    static const unsigned int DIGEST_SIZE = ( 256 / 8);

protected:
    void transform(const unsigned char *message, unsigned int block_nb);
    unsigned int m_tot_len;
    unsigned int m_len;
    unsigned char m_block[2*SHA224_256_BLOCK_SIZE];
    uint32 m_h[8];
};

string sha256(string input);

#define SHA2_SHFR(x, n)    (x >> n)
#define SHA2_ROTR(x, n)   ((x >> n) | (x << ((sizeof(x) << 3) - n)))
#define SHA2_ROTL(x, n)   ((x << n) | (x >> ((sizeof(x) << 3) - n)))
#define SHA2_CH(x, y, z)  ((x & y) ^ (~x & z))
#define SHA2_MAJ(x, y, z) ((x & y) ^ (x & z) ^ (y & z))
#define SHA256_F1(x) (SHA2_ROTR(x, 2) ^ SHA2_ROTR(x, 13) ^ SHA2_ROTR(x, 22))
#define SHA256_F2(x) (SHA2_ROTR(x, 6) ^ SHA2_ROTR(x, 11) ^ SHA2_ROTR(x, 25))
#define SHA256_F3(x) (SHA2_ROTR(x, 7) ^ SHA2_ROTR(x, 18) ^ SHA2_SHFR(x, 3))
#define SHA256_F4(x) (SHA2_ROTR(x, 17) ^ SHA2_ROTR(x, 19) ^ SHA2_SHFR(x, 10))

#define SHA2_UNPACK32(x, str) \
{ \
```

```

        *((str) + 3) = (uint8) ((x)      ); \
        *((str) + 2) = (uint8) ((x) >> 8); \
        *((str) + 1) = (uint8) ((x) >> 16); \
        *((str) + 0) = (uint8) ((x) >> 24); \
    }

#define SHA2_PACK32(str, x) \
{ \
    *(x) = \
        ((uint32) *((str) + 3)      ) \
        | ((uint32) *((str) + 2) << 8) \
        | ((uint32) *((str) + 1) << 16) \
        | ((uint32) *((str) + 0) << 24); \
}
#endif

```

SHA256.cpp

```

#include <bits/stdc++.h>
#include <fstream>
#include "sha256.h"
using namespace std;

const unsigned int SHA256::sha256_k[64] = //UL = uint32
    {0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
     0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
     0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
     0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
     0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
     0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
     0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
     0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
     0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
     0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
     0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
     0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
     0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
     0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
     0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
     0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2};

void SHA256::transform(const unsigned char *message, unsigned int block_nb)
{
    uint32 w[64];
    uint32 ww[8];
    uint32 t1, t2;
    const unsigned char *sub_block;

```

```

int i;
int j;
for (i = 0; i < (int) block_nb; i++) {
    sub_block = message + (i << 6);
    for (j = 0; j < 16; j++) {
        SHA2_PACK32(&sub_block[j << 2], &w[j]);
    }
    for (j = 16; j < 64; j++) {
        w[j] = SHA256_F4(w[j - 2]) + w[j - 7] + SHA256_F3(w[j - 15]) + w[j -
16];
    }
    for (j = 0; j < 8; j++) {
        wv[j] = m_h[j];
    }
    for (j = 0; j < 64; j++) {
        t1 = wv[7] + SHA256_F2(wv[4]) + SHA2_CH(wv[4], wv[5], wv[6])
            + sha256_k[j] + w[j];
        t2 = SHA256_F1(wv[0]) + SHA2_MAJ(wv[0], wv[1], wv[2]);
        wv[7] = wv[6];
        wv[6] = wv[5];
        wv[5] = wv[4];
        wv[4] = wv[3] + t1;
        wv[3] = wv[2];
        wv[2] = wv[1];
        wv[1] = wv[0];
        wv[0] = t1 + t2;
    }
    for (j = 0; j < 8; j++) {
        m_h[j] += wv[j];
    }
}

void SHA256::init()
{
    m_h[0] = 0x6a09e667;
    m_h[1] = 0xbb67ae85;
    m_h[2] = 0x3c6ef372;
    m_h[3] = 0xa54ff53a;
    m_h[4] = 0x510e527f;
    m_h[5] = 0x9b05688c;
    m_h[6] = 0x1f83d9ab;
    m_h[7] = 0x5be0cd19;
    m_len = 0;
    m_tot_len = 0;
}

void SHA256::update(const unsigned char *message, unsigned int len)
{
    unsigned int block_nb;
    unsigned int new_len, rem_len, tmp_len;

```

```

const unsigned char *shifted_message;
tmp_len = SHA224_256_BLOCK_SIZE - m_len;
rem_len = len < tmp_len ? len : tmp_len;
memcpy(&m_block[m_len], message, rem_len);
if (m_len + len < SHA224_256_BLOCK_SIZE) {
    m_len += len;
    return;
}
new_len = len - rem_len;
block_nb = new_len / SHA224_256_BLOCK_SIZE;
shifted_message = message + rem_len;
transform(m_block, 1);
transform(shifted_message, block_nb);
rem_len = new_len % SHA224_256_BLOCK_SIZE;
memcpy(m_block, &shifted_message[block_nb << 6], rem_len);
m_len = rem_len;
m_tot_len += (block_nb + 1) << 6;
}

void SHA256::final(unsigned char *digest)
{
    unsigned int block_nb;
    unsigned int pm_len;
    unsigned int len_b;
    int i;
    block_nb = (1 + ((SHA224_256_BLOCK_SIZE - 9)
        < (m_len % SHA224_256_BLOCK_SIZE)));

    len_b = (m_tot_len + m_len) << 3;
    pm_len = block_nb << 6;
    memset(m_block + m_len, 0, pm_len - m_len);
    m_block[m_len] = 0x80;
    SHA2_UNPACK32(len_b, m_block + pm_len - 4);
    transform(m_block, block_nb);

    for (i = 0 ; i < 8; i++) {
        SHA2_UNPACK32(m_h[i], &digest[i << 2]);
    }
}

string sha256(string input)
{
    unsigned char digest[SHA256::DIGEST_SIZE];
    memset(digest, 0, SHA256::DIGEST_SIZE);

    SHA256 ctx = SHA256();
    ctx.init();
    ctx.update( (unsigned char*)input.c_str(), input.length());
    ctx.final(digest);

    char buf[2*SHA256::DIGEST_SIZE+1];

```



```

    buf[2*SHA256::DIGEST_SIZE] = 0;
    for (int i = 0; i < SHA256::DIGEST_SIZE; i++)
        sprintf(buf+i*2, "%02x", digest[i]);
    return string(buf);
}

int main(int argc, char *argv[])
{
    string input = "secret";
    string output1 = sha256(input);

    cout << "SHA256('"<< input << "') : " << output1 << endl;
    return 0;
}

```

OUTPUT

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Users\alpna\Desktop\6TH SEM\IS\codes> cd "c:\Users\alpna\Desktop\6TH SEM\IS\codes\" ;
ha256 }
SHA256('secret') : 2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b
PS C:\Users\alpna\Desktop\6TH SEM\IS\codes> 

```

PLOT AN ELLIPTIC CURVE OVER FINITE FIELD. CHECK WHETHER THE POINTS LIES ON THE ELLIPTIC CURVE OR NOT.

```
import numpy as np
import matplotlib.pyplot as plt

def ecc(x, p, A, B):
    assert (4*A**3 + 27*B**2) % p!= 0
    return (x**3 + A*x + B) % p

# Find the elements in x^2 that are equal to elements in y^2, which in finite
field is to find sqrt
def sqrt_f(x, y2, p):
    x2 = x**2 % p
    y = [(i, *y_i) for i, y_i in enumerate([np.where(y2_i == x2)[0] for y2_i in
y2]) if y_i.size > 0]
    return y

# Order
p = 31
x = np.array(range(0, p))

A = -5
B = 8

x2 = x**2 % p
y2 = ecc(x, p, A, B)

# Compute the sqrt of the elliptic curve for plotting
y = sqrt_f(x, y2, p)

fig = plt.figure(dpi=100)
for y_p in y:
    [plt.scatter(y_p[0], i, c='b') for i in y_p[1:]]

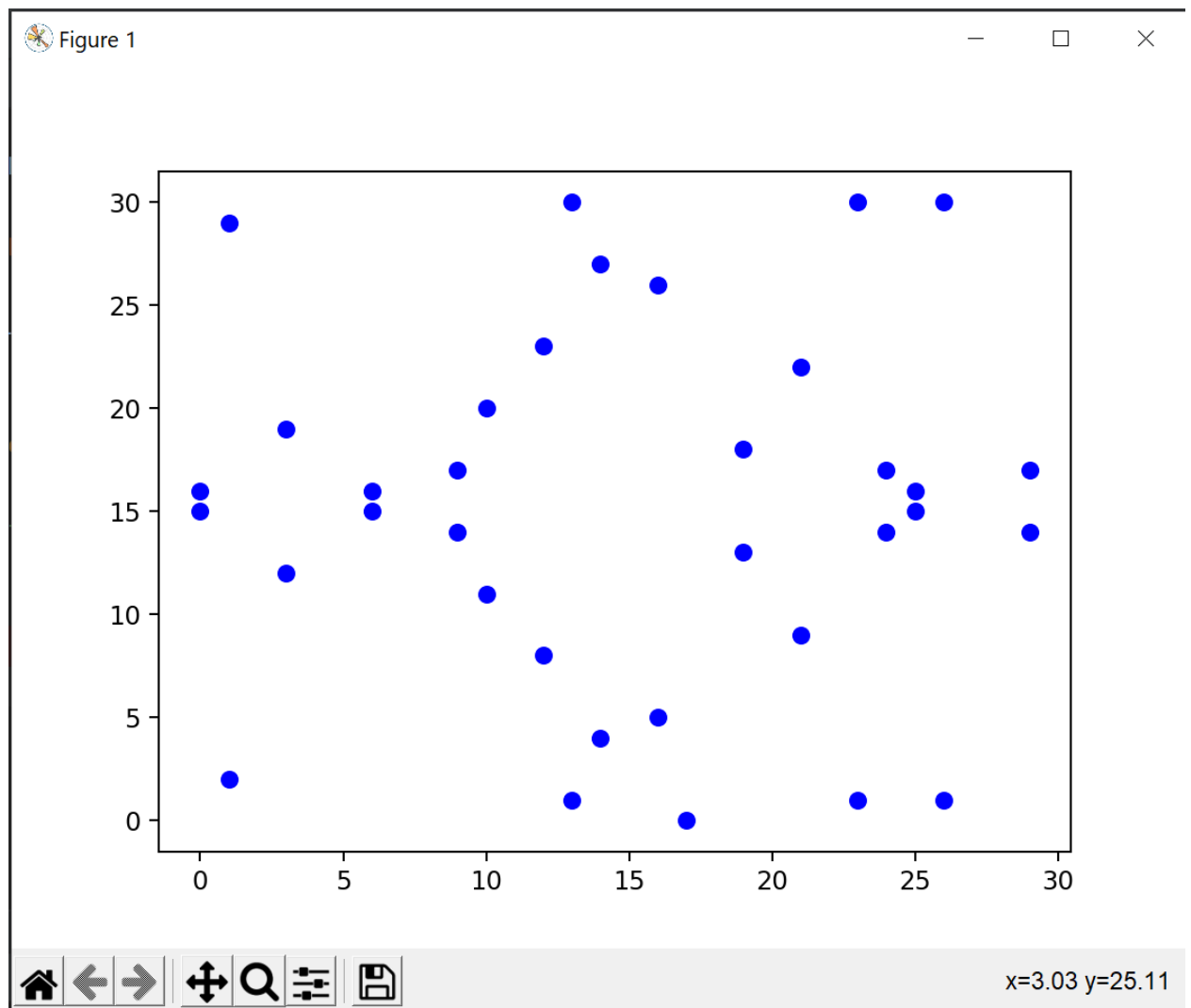
# Number of points and +1 for point at infinity
len(sum([y_p[1:] for y_p in y], ())) + 1

print(*['[x:{}, y:{}]'.format(y_p[0], y_p[1:]) for y_p in y])

plt.show()
```

OUTPUT

```
Run: main
C:\IS\venv\Scripts\python.exe C:/IS/main.py
[x:0, y:(15, 16)] [x:1, y:(2, 29)] [x:3, y:(12, 19)] [x:6, y:(15, 16)] [x:9, y:(14, 17)] [x:10, y:(11, 20)] [x:12, y:(8, 23)] [x:13, y:(1, 30)] [x:14, y:(4, 27)] [x:16, y:(5, 26)] [x:17, y:(0,)] [x:19, y:(13, 18)] [x:21, y:(9, 22)] [x:23, y:(1, 30)] [x:24, y:(14, 17)] [x:25, y:(15, 16)] [x:26, y:(1, 30)] [x:29, y:(14, 17)]
Process finished with exit code 0
```



PERFORM THE ELLIPTIC CURVE OPERATIONS LIKE ADDITION **AND MULTIPLICATION OF TWO POINTS AND FIND THE** **INVERSE OF A POINT ALSO.**

```
import numpy as np
import matplotlib.pyplot as plt

def ecc(x, p, A, B):
    assert (4 * A ** 3 + 27 * B ** 2) % p != 0
    return (x ** 3 + A * x + B) % p

# Find the elements in x^2 that are equal to elements in y^2, which in finite
# field is to find sqrt
def sqrt_f(x, y2, p):
    x2 = x ** 2 % p
    y = [(i, *y_i) for i, y_i in enumerate([np.where(y2_i == x2)[0] for y2_i in
y2]) if y_i.size > 0]
    return y

# Order
p = 31
x = np.array(range(0, p))

A = -5
B = 8

x2 = x ** 2 % p
y2 = ecc(x, p, A, B)

# Compute the sqrt of the elliptic curve for plotting
y = sqrt_f(x, y2, p)

fig = plt.figure(dpi=100)
for y_p in y:
    [plt.scatter(y_p[0], i, c='b') for i in y_p[1:]]

def get_2P(P, p, A, B):
    x, y = P

    lam = (3 * x ** 2 + A) * eea(2 * y, p) % p
    nu = (-x ** 3 + A * x + 2 * B) * eea(2 * y, p) % p
    # n = (x**4 - 2*A*x**2 - 8*B*x + A**2) % p
    # d = (4 * (x ** 3 + A * x + B)) % p
    # d = eea(d, p)
    return (lam ** 2 - 2 * x) % p, (-lam ** 3 + 2 * lam * x - nu) % p
    # return n * d % p, ecc(n * d % p, p, A, B)

def get_PQ(P, Q, p, A, B):
    x1, y1 = P
    x2, y2 = Q

    lam = (y2 - y1) * eea((x2 - x1) % p, p) % p
    nu = (y1 * x2 - y2 * x1) * eea((x2 - x1) % p, p) % p
```

```

        return (lam ** 2 - x1 - x2) % p, (-lam ** 3 + lam * (x1 + x2) - nu) % p

def sum_on_E(P, Q, p, A, B):
    # We have a point at infinity
    if (P == 0):
        return Q
    if (Q == 0):
        return P
    if (P == Q):
        if P[0] == 0:
            # Point at infinity
            return 0
        val = get_2P(P, p, A, B)
        return val
    else:
        if P[0] == Q[0]:
            return 0

        val = get_PQ(P, Q, p, A, B)
        return val

# Extended Euclidean Algorithm for finding inverse of x mod p
def eea(x, p, debug=False):
    # Quotient
    q = int(p / x)
    # Remainder
    r = r_old = p % x

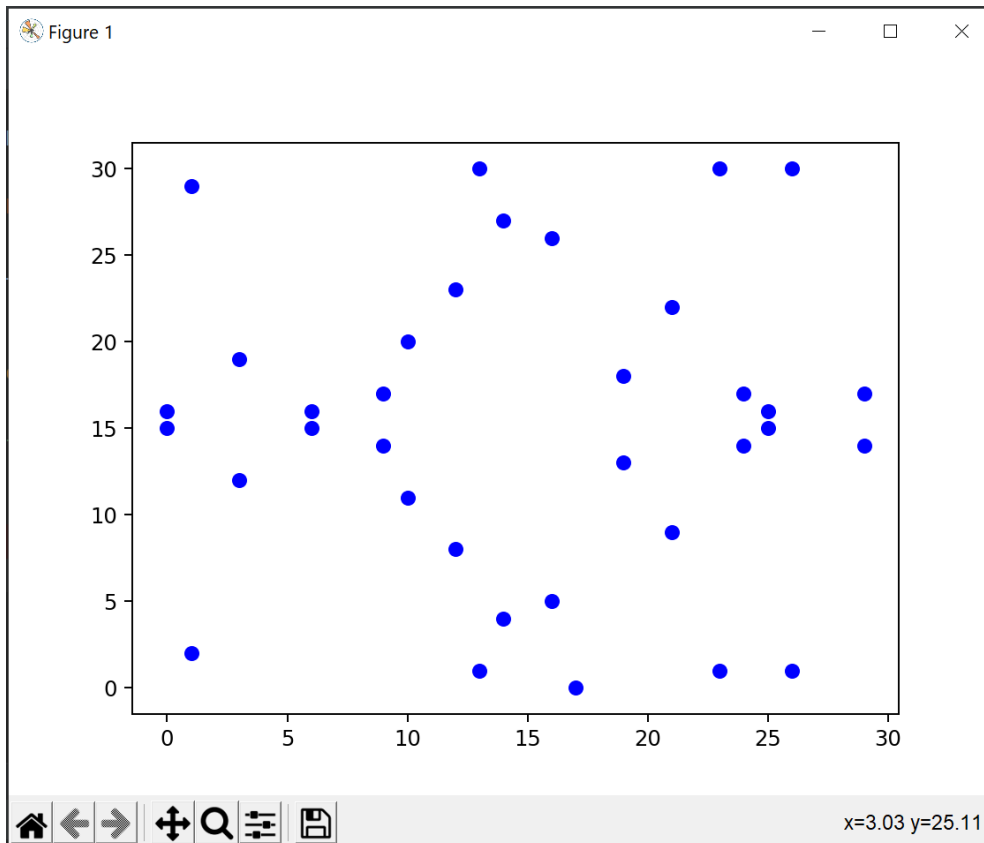
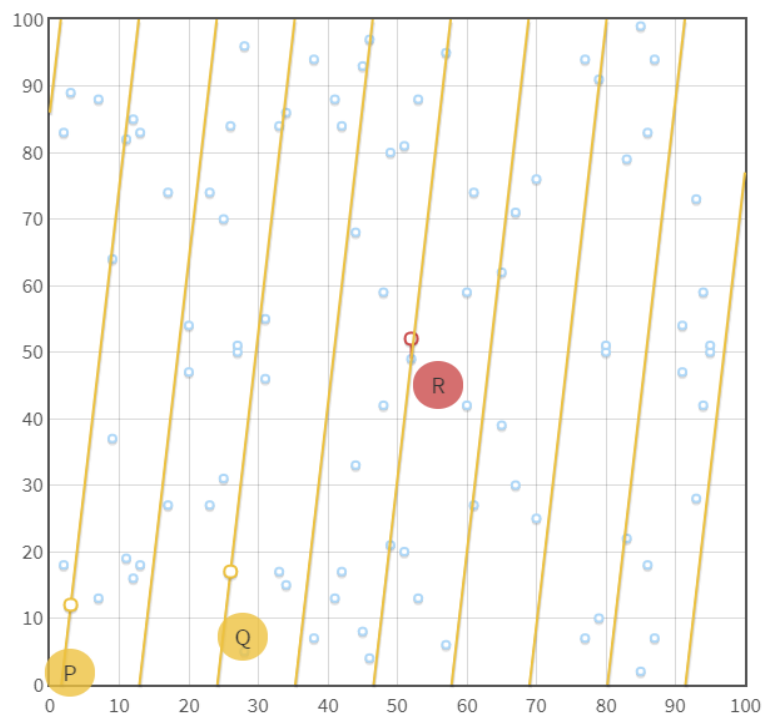
    q_s = [q]
    a_s = [0, 1]
    i = 0
    while True:
        if debug:
            print(f'{q} * x + r = {q} * {x} + {r}')
        if i > 1:
            a_i = (a_s[i - 2] - a_s[i - 1] * q_s[i - 2]) % p
            a_s.append(a_i)
        if r == 0:
            if r_old != 1:
                assert "No inverse"
            i = i + 1
            break
        r_old = r
        q = int(x / r)
        q_s.append(q)
        r = x % r
        x = r_old
        i = i + 1

    a_i = (a_s[i - 2] - a_s[i - 1] * q_s[i - 2]) % p
    return a_i

plt.show()

```

OUTPUT



IMPLEMENT THE RSA DIGITAL SIGNATURE SCHEME

```
#include <iostream>
#include<bits/stdc++.h>
using namespace std;

int modu(int b, unsigned int exp, unsigned int m){
int x = 1;
int i;
int power = b % m;
for (i = 0; i < sizeof(int) * 8; i++) {

    int least_bit = 0x00000001 & (exp >> i);
    if (least_bit)
        x = (x * power) % m;
    power = (power * power) % m;
} return x;
}

int modI(int a, int m)
{
int temp = m;
int y = 0, x = 1;
if (m == 1)
    return 0;
while (a > 1)
{ int q = a / m;
  int t = m;
  m = a % m, a = t;
  t = y;
  y = x - q * y;
  x = t;
} if (x < 0)
  x += temp;
return x;
}

int gcd(int a, int b)
{
if (a == 0 || b == 0)
    return 0;

if (a == b)
    return a;

if (a > b)
    return gcd(a-b, b);
return gcd(a, b-a);
}

int Prime(int num){
int flag = 1;
for(int i=2;i<=sqrt(num);i++)
{
```

```

    if(num%i==0)
    {
        flag = 0;
        return flag;
    }
}
return flag;
}
int lcm(int a, int b)
{
    return (a*b)/gcd(a, b);
}
int main(){
    int msg; char m;

    cout<<"\n Enter the character to be encrypted: ";
    cin>>m;
    msg = (int)m;
    cout<<"\n The corresponding ASCII value of the character is"<<msg;

    int p,q, random; int i=0; int a[2];
    srand (time(NULL));
    generate:
    while(i<2){
        random = rand() % 40 + 3;
        if(Prime(random)){
            a[i]=random;
            i++;
        } }
    i=0;p=a[0];q=a[1];
    if(p==q){
        goto generate;
    }
    cout<<"\n The Random Prime Numbers are: "<<p<<" and "<<q;

    int n; n = p*q;
    int phi = (p-1)*(q-1);
    int lambda = lcm(p-1,q-1);
    int e;
    vector<int> tot;
    for(int i=3;i<lambda;i++)
    { if(gcd(i,lambda) == 1){
        tot.push_back(i);
    }
    }
    int size = tot.size();
    int ran = rand() % size;
    e = tot[ran];

    cout<<"\n The modulus is: "<<n;
    cout<<"\n The phi(n) is: "<<phi;
    cout<<"\n The lambda(n) is:"<<lambda;
    cout<<"\n The toitent is: "<<e;
    cout<<"\n The public key is: ("<<n<<","<<e<<");
    long long int encrypt;
    encrypt = modu(msg,e,n);

```



```

cout<<"\n The Cipher text is: "<<(char)encrypt;
cout<<"\n The ASCII value of Cipher Text is: "<<encrypt;

long long int d = modI(e,lambda);
if(d==e){
    cout<<"\n";
    goto generate;
}
cout<<"\n The private key is: "<<d;
long long int decrypted;
decrypted = modu(encrypt,d,n);
cout<<"\n The Decrypted/Plain text is : "<<(char)decrypted;
cout<<"\n The ASCII value of Plain text is: "<<decrypted;
return 0;
}

```

OUTPUT

```

Enter the character to be encrypted: B

The corresponding ASCII value of the character is66
The Random Prime Numbers are: 37 and 3
The modulus is: 111
The phi(n) is: 72
The lambda(n) is:36
The toitent is: 23
The public key is: (111,23)
The Cipher text is: <
The ASCII value of Cipher Text is: 60
The private key is: 11
The Decrypted/Plain text is : B
The ASCII value of Plain text is: 66

```

IMPLEMENT ELGAMAL DIGITAL SIGNATURE SCHEME

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int checkIfPrime (int newPrime);
int checkRoot (int newPrime, int newRoot);
int computeYa (int newRoot, int newXa, int newPrime);
int computeInverseK (int userK, int newPrime);
int computeS2 (int inverseK, int userHash, int userXa, int userS1, int newPrime);
int computeV2 (int Ya, int S1, int S2, int newPrime);

int main ()
{
    int p = 0, q = 0, randXa = 0, UserAYa = 0, userS1 = 0, userS2 = 0;
    int primeCheck = 0, primitiveCheck = 0, userInt = 0;
    int hashValue = -1, randK = 0, inverseK = 0;
    char userChoice;
    char userMessage[256];
    int gcd;
    int V1 = 0, V2 = 0;
    printf ("Welcome to the ElGamal digital signature this works with primes up to
17\n\nThe global elements of this signature are a prime q\nand p, which is the
primitive root of q\n");
    while (primeCheck == 0)
    {
        printf ("\nEnter a prime number q: ");
        scanf ("%d", &q);
        primeCheck = checkIfPrime (q);
        if (primeCheck == 1)
            printf (" This is in fact prime\n");
        else
        {
            printf (" Whoops,not prime\n");
            primeCheck = 0;
        }
    }

    while (primitiveCheck == 0)
    {
        printf ("\nEnter a primitive root of p: ");
        scanf ("%d", &p);
        primitiveCheck = checkRoot (q, p);
        if (primitiveCheck == 1) printf ("\nThis is in fact a primitive root\n");
        else
        {
            printf ("\nWhoops, not a primitive root\n");
            primeCheck = 0;
        }
    }
}
```

```

    }
}
printf ("\n\nNext User A generates a private/public key pair. First your random
integer is generated");
printf ("\nWould you like to enter a number greater than 1 but less than %d, y
for yes or n for no: ", q - 1);
scanf (" %c", &userChoice);
if (userChoice == 'n')
{
    while ((randXa <= 1) || (randXa >= q - 1))
    {
        randXa = rand ();
    }
}
else
{
    while (randXa == 0)
    {
        printf ("Enter your random integer: ");
        scanf ("%d", &randXa);
        if ((randXa <= 1) || (randXa >= q - 1))
            randXa = 0;
    }
}
printf ("\n\nNext we computer Ya = p^Xa mod q");
UserAYa = computeYa (p, randXa, q);
printf ("\nUser A's Ya is %d", UserAYa);

printf ("\nA's private key Xa is %d, A's public key is {q, p, Ya}
{%d,%d,%d}\n",
randXa, q, p, UserAYa);
while (getchar () != '\n'); //Clears the buffer
printf ("\n\nPlease enter your message:");
scanf ("%[^\n]", userMessage);
printf ("\nNext we are gonna give the user Message M a hash value m.");
printf ("\nWould you like to enter one? y/n: ");
scanf (" %c", &userChoice);
if (userChoice == 'n')
{
    while ((hashValue < 0) || (hashValue > q - 1))
    {
        hashValue = rand ();
    }
}
else
{
    while (hashValue == -1)
    {
        printf ("Enter your hash value: ");
        scanf ("%d", &hashValue);
        if ((hashValue < 0) || (hashValue > q - 1)) hashValue = -1;
    }
}

```

```

    }
}
printf ("\nThe hash value is %d", hashValue);
printf ("\n\nA then forms a digital signature as follows");
while (randK < 1)
{
    while ((randK < 1) || (randK > q - 1))
    {
        randK = rand ();
    }
    for (int i = 1; i <= randK && i <= (q - 1); i++)
    {
        if ((randK % i == 0) && ((q - 1) % i == 0))
            gcd = i;
    }
    if (gcd != 1)
        randK = 0;
}
printf ("\n The a random integer K is generated %d\n", randK);
printf ("\n Next we compute S1 = p^K mod q\n");
userS1 = computeYa (p, randK, q);
printf (" S1 is %d\n", userS1);
printf ("\n Next we compute K^-1 mod(q-1)");
inverseK = computeInverseK (randK, q);
printf ("\n K^-1 is %d because %d*%d mod %d = 1", inverseK, inverseK,
randK, q - 1);
printf ("\n\n Next we compute S2 = K^-1(m-XaS1)mod(q-1)");
userS2 = computeS2 (inverseK, hashValue, randXa, userS1, q);
printf ("\n S2 is %d because it is %d*(%d - %d*%d)mod(%d) ", userS2,
inverseK, hashValue, randXa, userS1, q - 1);
printf ("\n\nThe signature consists of (S1, S2) which are %d , %d", userS1,
userS2);
//Verification
printf ("\nWould you like to verify? y/n: ");
scanf (" %c", &userChoice);
if (userChoice == 'y')
{
    printf ("\n\nTo check first we calculate V1 = p^m mod q");
    V1 = computeYa (p, hashValue, q);
    printf ("\n\nThen we calculate V2 = (Ya^S1 * S1^S2)mod q");
    V2 = computeV2 (UserAYa, userS1, userS2, q);
    printf ("\n\nFinally, we check whether they equal each other");
    if (V1 == V2) printf ("\nThe signature is valid V1 %d == V2 %d\n\n", V1,
V2);
    else printf ("\nSignature is invalid V1 %d != V2 %d\n\n", V1, V2);
}

return 0;
}

int computeV2 (int Ya, int S1, int S2, int newPrime)

```

```

{
    int newV2 = 0;
    long long int exYa = 1, exS1 = 1;
    while (S2 != 0)
    {
        exS1 *= S1;
        --S2;
    }
    while (S1 != 0)
    {
        exYa *= Ya;
        --S1;
    }
    exYa = exYa % newPrime;
    exS1 = exS1 % newPrime;
    newV2 = exYa * exS1;
    newV2 = newV2 % newPrime;
    return newV2;
}

int computeS2 (int inverseK, int userHash, int userXa, int userS1, int newPrime)
{
    int newS2 = 0;
    newS2 = inverseK * (userHash - (userXa * userS1));
    newS2 = newS2 % (newPrime - 1);
    if (newS2 < 0)
        newS2 = newS2 + (newPrime - 1);
    return newS2;
}

int computeInverseK (int userK, int newPrime)
{
    int inverseK = 0;
    int testValue, posValue = 1;
    while (inverseK == 0)
    {
        testValue = (posValue * userK) % (newPrime - 1);
        if (testValue == 1)
        {
            inverseK = posValue;
        }
        else
        {
            posValue++;
        }
    }
    return inverseK;
}

int computeYa (int newRoot, int newXa, int newPrime)

```

```

{
    long long int newYa = 1;
    while (newXa != 0)
    {
        newYa *= newRoot;
        --newXa;
    }
    newYa = newYa % newPrime;
    return (int) newYa;
}

// Checks if number is prime
int checkIfPrime (int newPrime)
{
    if (newPrime <= 1)
        return 0;
    if (newPrime > 2)
        for (int factor = 2; factor < newPrime; factor++)
        {
            if (newPrime % factor == 0)
                return 0;
        }
    return 1;
}

int checkRoot (int newPrime, int newRoot)
{
    long long int primeCombo = 0, modCheck = 1;
    int power;
    char result;
    if (newRoot >= newPrime) return 0;

    for (int factor = 1; factor < newPrime; factor++) primeCombo = primeCombo +
factor;
    printf ("\n Starting PrimeCombo %d", primeCombo);
    for (int exponent = 1; exponent < newPrime; exponent++)
    {
        power = exponent;
        while (power != 0){modCheck *= (long long int) newRoot; --power;}
        modCheck = modCheck % (long long int) newPrime;
        printf ("\n%d^%d mod %d is %lli", newRoot, exponent, newPrime, modCheck);
        modCheck = 1;
    }
    printf ("\nDid any value repeat?(y for yes or n for no) ");
    scanf (" %c", &result);
    if (result == 'y')
        return 0;
    else
        return 1;
}

```

OUTPUT

```
PS C:\Users\Manoj Kumar> cd "c:\Users\Manoj Kumar\Downloads\" ; if ($?) { g++ elgamal.cpp -o elgamal } ; if ($?) { .\elgamal }
Welcome to the ElGamal digital signature this works with primes up to 17
```

The global elements of this signature are a prime q
and p , which is the primitive root of q

Enter a prime number q : 19
This is in fact prime

Enter a primitive root of p : 10

Starting PrimeCombo 171

```
10^1 mod 19 is 10
10^2 mod 19 is 5
10^3 mod 19 is 12
10^4 mod 19 is 6
10^5 mod 19 is 3
10^6 mod 19 is 11
10^7 mod 19 is 15
10^8 mod 19 is 17
10^9 mod 19 is 18
10^10 mod 19 is 9
10^11 mod 19 is 14
10^12 mod 19 is 7
10^13 mod 19 is 13
10^14 mod 19 is 16
10^15 mod 19 is 8
10^16 mod 19 is 4
10^17 mod 19 is 2
10^18 mod 19 is 1
```

Did any value repeat?(y for yes or n for no) n

This is in fact a primitive root

Next User A generates a private/public key pair. First your random integer is generated
Would you like to enter a number greater than 1 but less than 18, y for yes or n for no: y
Enter your random integer: 16

Next we computer $Y_a = p^{X_a} \bmod q$

User A's Y_a is 4

A's private key X_a is 16, A's public key is $\{q, p, Y_a\}$ {19,10,4}

Please enter your message:this is my message

Next we are gonna give the user Message M a hash value m .
Would you like to enter one? y/n: n

The hash value is 8

A then forms a digital signature as follows
The a random integer K is generated 1

Next we compute $S_1 = p^K \bmod q$
 S_1 is 10

Next we compute $K^{-1} \bmod (q-1)$
 K^{-1} is 1 because $1*1 \bmod 18 = 1$

Next we compute $S_2 = K^{-1}(m - X_a S_1) \bmod (q-1)$
 S_2 is 10 because it is $1*(8 - 16*10) \bmod (18)$

The signature consists of (S_1, S_2) which are 10 , 10
Would you like to verify? y/n: y

To check first we calculate $V_1 = p^m \bmod q$

Then we calculate $V_2 = (Y_a^{S_1} * S_1^{S_2}) \bmod q$

Finally, we check whether they equal each other
The signature is valid $V_1 17 == V_2 17$

IMPLEMENT SCHNORR DIGITAL SIGNATURE SCHEME

```
#include <bits/stdc++.h>
using namespace std;
#define ld long long

ld ModInv(ld b, ld n)
{
    ld r1 = n, r2 = b, t1 = 0, t2 = 1;
    while (r2 > 0)
    {
        ld q = r1 / r2;
        ld r = r1 - q * r2;
        r1 = r2;
        r2 = r;
        ld t = t1 - q * t2;
        t1 = t2;
        t2 = t;
    }
    if (t1 < 0)
        t1 += n;
    return t1;
}

ld powerLL(ld x, ld n, ld MOD)
{
    ld result = 1;
    while (n)
    {
        if (n & 1)
            result = result * x % MOD;
        n = n / 2;
        x = x * x % MOD;
    }
    return result;
}

ld powerStrings(string sa, string sb, ld MOD)
{
    ld a = 0, b = 0;
    for (int i = 0; i < sa.length(); i++)
        a = (a * 10 + (sa[i] - '0')) % MOD;
```



```

    for (int i = 0; i < sb.length(); i++)
        b = (b * 10 + (sb[i] - '0')) % (MOD - 1);
    return powerLL(a, b, MOD);
}

ld mod(string num, ld a)
{
    ld res = 0;

    // One by one process all digits of 'num'
    for (int i = 0; i < num.length(); i++)
        res = (res * 10 + (int)num[i] - '0') % a;

    return res;
}

int main()
{
    ld p, q, e0, d, m, r, MOD;
    cout << "Enter the value of p, q, e0, d, m and r respectively: ";
    cin >> p >> q >> e0 >> d >> m >> r;
    // 2267 103 2 30 1000 11

    ld a, b, t, bef_v, bef_s1, e1, e2, s1, s2, v;
    e1 = powerStrings(to_string(e0), to_string((p - 1) / q), p);

    if (e1 < 0)
        e1 += p;
    e2 = powerStrings(to_string(e1), to_string(d), p);
    if (e2 < 0)
        e2 += p;

    //////////// For concat S1 we are finding both components
    a = powerStrings(to_string(e1), to_string(r), p);
    if (a < 0)
        a += p;
    bef_s1 = stoll(to_string(m) + to_string(a), NULL, 10);

    s1 = 200;

    s2 = (r + ((d * s1) % q)) % q;
    //////////// for concat V we are finding both components
    a = powerStrings(to_string(e1), to_string(s1), p);
    if (a < 0)
        a += p;
    b = powerStrings(to_string(e2), to_string(s2), p);

```

```

if (b < 0)
    b += p;
t = (a * b) % p;

bef_v = stoll(to_string(m) + to_string(t), NULL, 10);

cout << "e1=" << e1 << "\n";
cout << "e2=" << e2 << "\n";
cout << "s2=" << s2 << "\n";
cout << "bef_s1=" << bef_s1 << "\n";
cout << "bef_v=" << bef_v << "\n";
}

```

OUTPUT

```

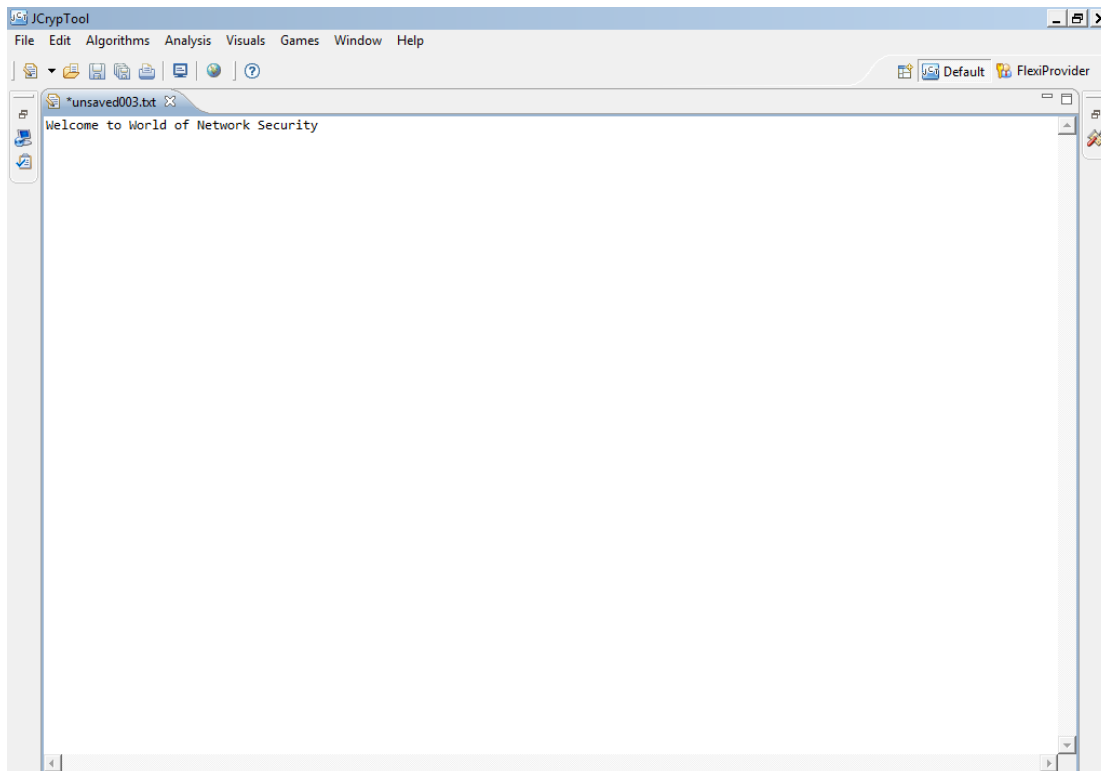
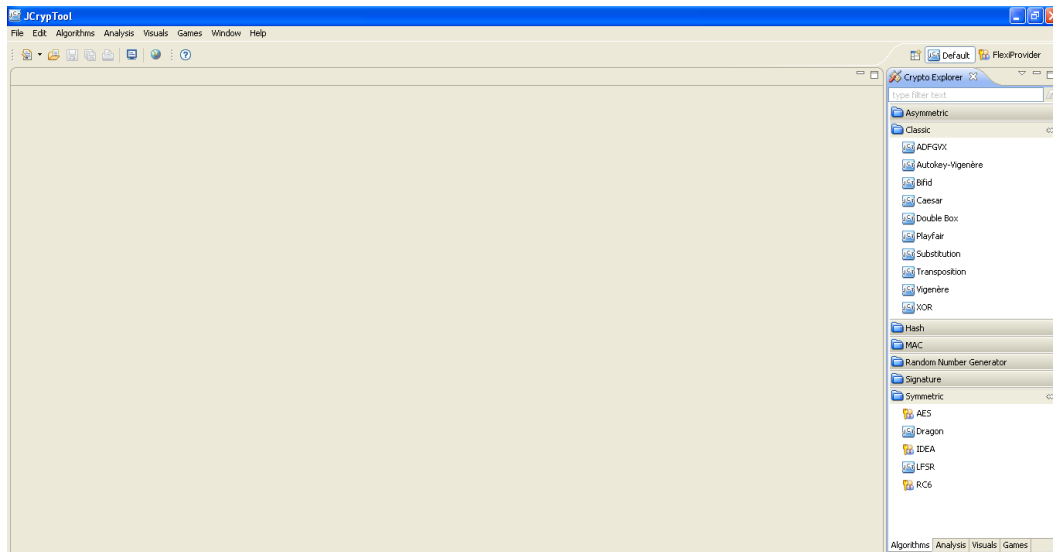
PS C:\Users\Manoj Kumar\Desktop\Software\lab> cd "c:\Users\Manoj Kumar\Desktop\Software\lab\" ;
Enter the value of p, q, e0, d, m and r respectively: 2267 103 2 30 1000 11
e1=354
e2=1206
s2=37
bef_s1=1000630
bef_v=10002053

```

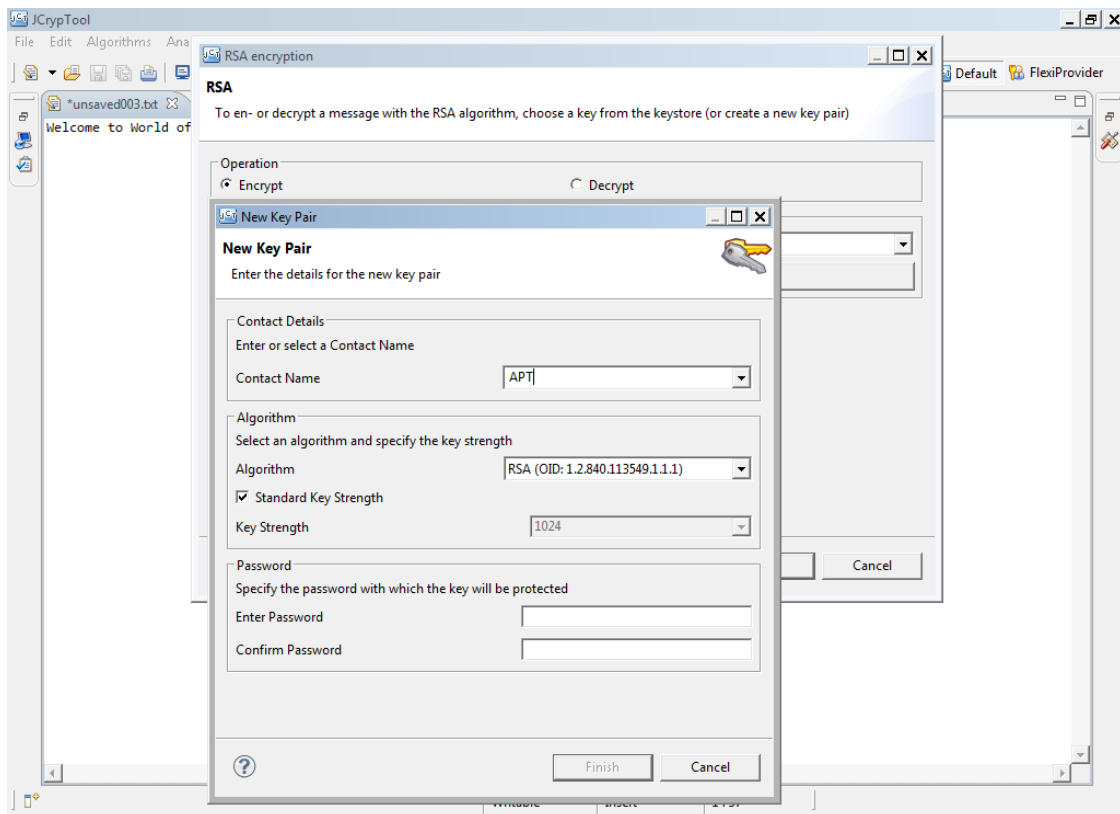
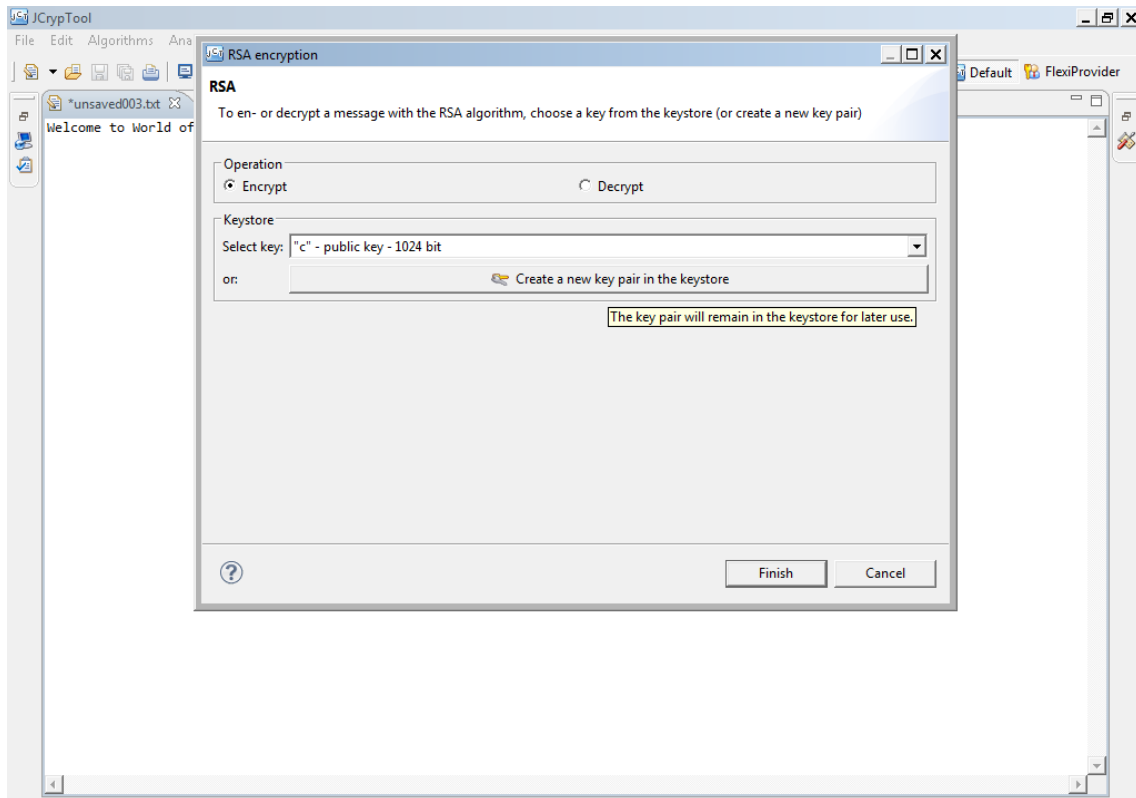
USING JCRIPT TOOL (OR ANY OTHER EQUIVALENT) TO DEMONSTRATE ASYMMETRIC, SYMMETRIC CRYPTO ALGORITHM

ASYMMETRIC ALGORITHM

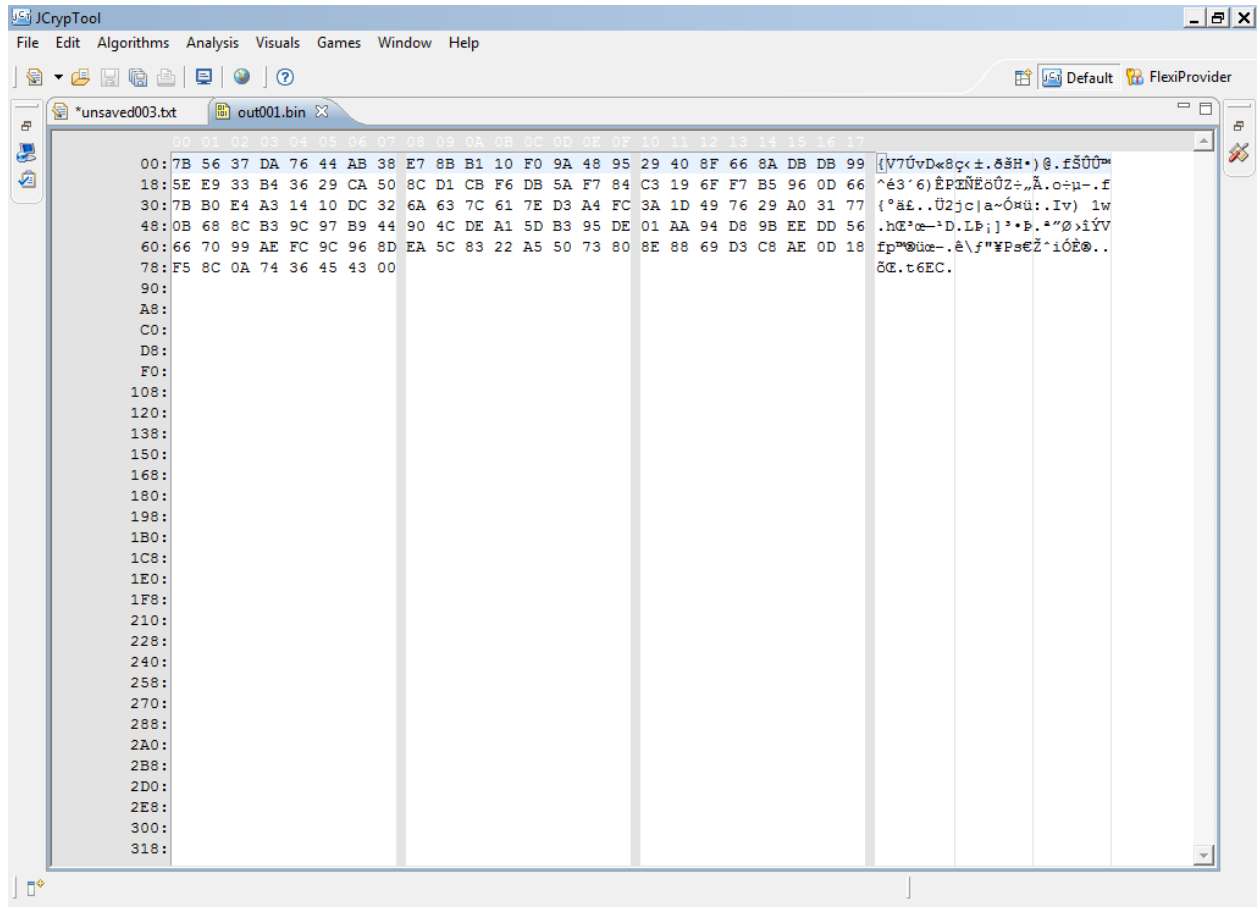
- Download Jcrypt tool from Cryptool Website and Install .
- Open Jcrypt Software and Click on NEW text editor, type the text information into it



- Click on the Algorithm menu bar and Select Asymmetric algorithm RSA for encryption.
- Click create a New KeyPair and type in the contact name[xxxxx] and enter the password and confirm password, then Click finish again.



- Now you can see RSA output bin file is generated.



- The same output bin file to decrypt select RSA Algorithm and Click on Decrypt, Select keyname you have declared earlier and Click Finish.
- Enter the password to Decrypt and see the output with original Decrypted text on the Screen.

SYMMETRIC ALGORITHM

- Click on Algorithm Menu bar Select Symmetric♦AES and Click on it.
- Click on create a new key, type contact name and enter the password and confirm, Click finish♦ Click finish again.
- Enter the password to open the output file.
- To Decrypt Select Algorithms♦ Symmetric♦Select the key which you have created and Click Finish.
- Enter the password and see the result in output bin file with hexadecimal values and plain text.

IMPLEMENT THE IDENTITY-BASED ENCRYPTION (IBE) **USE THE EMAIL ADDRESS OF THE RECIPIENT TO GENERATE** **THE KEY FOR A DESTINATION.**

Central Authority Code:

PKG.java

```
package CentralAuthority;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;

import java.io.IOException;
import java.math.BigInteger;

import java.util.Scanner;

public class PKG {

    private static BigInteger KEY , n;

    //creating file object from given path
    java.net.URL url1 = getClass().getResource("Data.txt");           // opening
user file
    File file = new File(url1.getPath());

    public void set_key(String id){

        RSA rsa = new RSA(id);

        KEY = rsa.get_public_key();
        n = rsa.getn();

        try(
            FileWriter fileWriter = new FileWriter(file,true);
            BufferedWriter bufferFileWriter = new BufferedWriter(fileWriter);

        ) {

            fileWriter.append(id);
            fileWriter.append("\n");

            fileWriter.append(rsa.get_public_key().toString());
```

```

        fileWriter.append("\n");

        fileWriter.append(rsa.get_private_key().toString());
        fileWriter.append("\n");

        fileWriter.append(n.toString());
        fileWriter.append("\n");

        bufferFileWriter.close();
        fileWriter.close();

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}

public BigInteger getn(){

    return n;

}

public BigInteger get_public_key(String id){                                     // Public
Key generator

    BigInteger pk = BigInteger.valueOf(-1), sk = BigInteger.valueOf(-1);
    boolean flag = true;

    try {

        Scanner p = new Scanner(file);
        String ID;

        while(p.hasNext()){

            ID = p.next();
            pk = p.nextBigInteger();
            sk = p.nextBigInteger();
            n = p.nextBigInteger();

            if(id.compareTo(ID) == 0){                                     // User is already
registered

                flag = false;
                break;

            }


```

```

    }

    p.close();
}
catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

if(!flag){

    KEY = pk;
    return KEY;

}

set_key(id);                                     // User is not registered ....
therefore registering
return KEY;

}

public BigInteger get_private_key(String id){      // Private Key
generator

    BigInteger x = BigInteger.valueOf(-1);
    boolean flag = true;

    try {

        Scanner p = new Scanner(file);
        String ID;

        while(p.hasNext()){

            ID = p.next();
            x = p.nextBigInteger();
            x = p.nextBigInteger();
            n = p.nextBigInteger();

            if(id.compareTo(ID) == 0){              // User is already
registered

                flag = false;
                break;

            }

        }
    }
}

```



```

        p.close();
    }
    catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    if(flag){
        set_key(id); // User is not
registered .... therefore registering
        x = get_private_key(id);
    }

    return x;
}
}

```

RSA.java

```

package CentralAuthority;

import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;

public class RSA {

    BigInteger public_key,private_key;

    private long public_key_temp;

    private BigInteger p;
    private BigInteger q;
    private BigInteger n;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
    private int bitlength = 1024;
    private Random r;

    String ID;
}

```

```

RSA(String ID){

    this.ID = ID ;
    public_key_temp = Math.abs(ID.hashCode());

    r = new Random();
    p = BigInteger.probablePrime(bitlength, r);
    q = BigInteger.probablePrime(bitlength, r);
    n = p.multiply(q);

}

public BigInteger get_public_key(){ // generating public
key

    phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
    e = BigInteger.valueOf(public_key_temp);

    while (phi.gcd(e).compareTo(BigInteger.valueOf(1)) != 0 ) {
        e = e.divide(phi.gcd(e));
    }

    public_key = e;
    get_private_key();

    return public_key;

}

public BigInteger get_private_key(){ // generating private key

    d = public_key.modInverse(phi);

    private_key =
extendedEuclid(public_key, (this.p.subtract(BigInteger.ONE)).multiply(this.q.subtrac
t(BigInteger.ONE)));

    return private_key;
}

public BigInteger getn(){

    return n;

}

```

```

public BigInteger extendedEuclid(BigInteger a, BigInteger b) {

    BigInteger x = BigInteger.valueOf(1), y = BigInteger.valueOf(0);
    BigInteger xLast = BigInteger.valueOf(0), yLast = BigInteger.valueOf(0);
    BigInteger q, r, m, n;

    while(a.compareTo(BigInteger.valueOf(0)) != 0) {

        q = b.divide(a);
        r = b.remainder(a);
        m = xLast.subtract(q.multiply(x));
        n = yLast.subtract(q.multiply(y));

        xLast = x;
        yLast = y;

        x = m;
        y = n;
        b = a;
        a = r;

    }

    if(xLast.compareTo(BigInteger.valueOf(0))<0)
        xLast =
xLast.add((this.p.subtract(BigInteger.ONE)).multiply(this.q.subtract(BigInteger.ONE
)));

    return xLast;

}

public long power(long a, long b, long p) { // power
function a^b%p

    long r = 1;

    while(b!=0) {

        if((b & 1) != 0) r = r * a % p;
        a = (a * a)% p;
        b >>= 1;

    }

    return r;

}

public BigInteger gcd1(BigInteger x, BigInteger y){ // computing
gcd

```

```

        if(y.compareTo(BigInteger.valueOf(0)) == 0) return x;
        return gcd1(y,x.reminder(y));
    }
}

```

Client code:

Client.java

```

package Client;

import java.math.BigInteger;

import java.util.Scanner;

import CentralAuthority.PKG;

public class Client {

    public static void main(String args[]){

        String message,ID,temp,ID1;

        Scanner p = new Scanner (System.in);

        System.out.println("Hi ! ...Client");

        System.out.println("Enter the your ID (for eg: xyz@gmail.com)");
        ID1 = p.next();

        System.out.println("Enter the User ID of Server (for eg: xyz@gmail.com)");
        ID = p.next();

        temp = p.nextLine();

        System.out.println("Enter the Message");
        temp = p.nextLine();

        message = ID1;
        message += " has sent you a message:\n";
        message += temp;

        PKG pkg = new PKG();
        BigInteger Public_key = pkg.get_public_key(ID1);
        Public_key = pkg.get_public_key(ID);
        BigInteger n = pkg.getn();
    }
}

```

```

        System.out.println("\npublic key of Server is : " + Public_key);

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

        Encrypt encrypt1 = new Encrypt(ID,message,n,Public_key);

        encrypt1.encrypt();

    }
}

```

Encrypt.java

```

package Client;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.math.BigInteger;

import CentralAuthority.PKG;

public class Encrypt {

    private String Message,ID;
    BigInteger n,Public_key;

    java.net.URL url = getClass().getResource("../Server/EncryptedMessage.txt");
    File file = new File(url.getPath());

    public Encrypt(String ID,String Message , BigInteger n , BigInteger
    Public_key){

        this.ID = ID;
        this.Message = Message;
        this.n = n;
        this.Public_key = Public_key;

    }
}

```

```

//Encrypt message
    public byte[] encrypt() {                                     // Encrypting message

        byte[] message = Message.getBytes();

        System.out.println(bytesToString((new
        BigInteger(message)).modPow(Public_key, n).toByteArray()));

        byte [] encryptedMessage = messageEncrypt(message);

        try(

            FileWriter fileWriter = new FileWriter(file,true);
            BufferedWriter bufferFileWriter = new BufferedWriter(fileWriter);

        ) {

            fileWriter.append(ID);
// Sending Message to server
            fileWriter.append(" ");
            fileWriter.append(Integer.toString(encryptedMessage.length));
            fileWriter.append(" ");

            for(byte e_message : encryptedMessage){
                fileWriter.append(Byte.toString(e_message));
                fileWriter.append(" ");
            }

            fileWriter.append("\n");
            bufferFileWriter.close();
            fileWriter.close();

        } catch (IOException e) {
            // TODO Auto-generated catch block

            e.printStackTrace();

        }

        return (new BigInteger(message)).modPow(Public_key, n).toByteArray();

    }

    private byte [] messageEncrypt(byte [] message){                // Actual
message Encryption

```

```

        return (new BigInteger(message)).modPow(Public_key, n).toByteArray();
    }

    private static String bytesToString(byte[] encrypted) {

        String test = "";

        for (byte b : encrypted) {
            test += Byte.toString(b);
            //test += " ";
        }
        return test;
    }
}

```

Server code:

Server.java

```

package Server;

import java.math.BigInteger;
import java.util.Scanner;

import CentralAuthority.PKG;

public class Server {

    public static void main(String args[]){

        String ID;

        Scanner p = new Scanner (System.in);

        System.out.println("Enter your User ID (for eg: abc@gmail.com)");
        ID = p.next();

        System.out.println("Hi... I am Server,\nSearching for messages\n");

        PKG pkg = new PKG();
        BigInteger Private_key =pkg.get_private_key(ID);
        BigInteger n = pkg.getn();

        System.out.println("\nMy Private Key is :- " + Private_key );
    }
}

```

```

        Decrypt decryptMessage = new Decrypt(ID,n,Private_key);

        System.out.println("\n -----Decrypted message is -----
        -----\n");

        decryptMessage.decrypt();

    }
}

```

Decrypt.java

```

package Server;

import java.io.File;
import java.io.FileNotFoundException;
import java.math.BigInteger;
import java.util.Scanner;
import Client.*;

public class Decrypt {

    BigInteger n , private_key;
    String ID;

    java.net.URL url = getClass().getResource("EncryptedMessage.txt");
    File file = new File(url.getPath());

    public Decrypt( String ID,BigInteger n , BigInteger private_Key){

        this.ID = ID;
        this.private_key = private_Key;
        this.n = n;

    }

    public String decrypt() { // First Reading Message
then crypting

        byte [] message = new byte[0];
        int k;
        boolean flag = false;

        try {

```



```

        Scanner p = new Scanner(file);
        while(p.hasNext()){

            if(p.next().compareTo(ID) == 0){                // Message Found
for this server

                k = p.nextInt();
                message = new byte[k];

                for(int i=0;i<k;i++){

                    message[i] = p.nextByte();

                }

                flag = true;
                System.out.println(new String(decryptMessage(message)) +
"\n");

            }
            else{

                k = p.nextInt();

                for(int i=0;i<k;i++){

                    p.nextByte();

                }

            }

        }

        p.close();
        if(flag)    return new String(decryptMessage(message));

    }
    catch (FileNotFoundException e) {

        // TODO Auto-generated catch block
        e.printStackTrace();

    }

    System.out.println(new String("There is no message for you !!!!!!!"));
    return new String("There is no message for you !!!!!!!");    // No Message
Found for this server

}

```

```

private static String bytesToString(byte[] encrypted) {

    String test = "";

    for (byte b : encrypted) {
        test += Byte.toString(b);
    }

    return test;

}

public byte[] decryptMessage(byte[] message) { // Actual Message
Decryption

    return (new BigInteger(message)).modPow(private_key, n).toByteArray();

}

}

```

OUTPUT

Encrypted message text

```

1  facebook.com 256 34 87 -80 72 43 12 22 48 -92 55 50 -89 5 -84 -58 -71 18 80 -11 -106 38 60 90 84 -69 -58 23 96 0 15 48 36 -54 -101 0 -33 74 62 -1 -78 -56 127 -69 -9
120 62 77 85 95 -8 125 -92 -1 -14 -37 -48 11 -102 -126 -14 -58 3 43 82 125 -58 44 -57 60 -109 -128 -64 -48 -20 -20 -104 -49 -47 50 -41 65 -13 126 56 -109 -79 124 63
-96 -35 29 -16 50 -55 111 71 37 14 63 65 -112 1 79 89 -104 -3 117 83 -108 8 34 38 -22 -13 -102 124 15 42 -52 32 -32 109 -82 -52 -22 28 14 59 -110 76 -102 78 -95 13 -45
46 -43 -106 -43 125 -31 119 118 -113 8 -34 112 11 87 19 -123 -105 123 55 -84 66 5 68 56 105 -109 -65 68 -115 -37 -58 -30 19 -127 49 112 34 51 -73 124 -93 -83 -48 -49
-85 75 79 88 78 -120 64 21 -6 122 -97 -31 30 97 -31 -90 76 -35 -70 64 59 -125 24 65 -81 -31 -47 42 -26 -114 22 -57 98 -106 40 -103 -102 84 -46 -34 20 -59 40 -93 56 -94
86 8 7 -45 67 62 -64 63 114 -7 -58 82 22 -67 23 88 -104 -15 -110 22 115 -4 -8 5 -107 18 -37 116 13 -88 113

2  facebook.com 256 20 -24 -94 69 -35 90 30 81 -38 67 -123 -12 -27 -93 -114 -85 -39 79 6 -78 -69 120 -118 -63 -40 41 -112 -126 29 -15 8 -128 58 112 -104 -82 -104 87 -92
91 127 -68 -75 115 -105 0 -90 34 -30 33 8 54 -4 -115 4 57 48 110 91 43 97 107 54 42 -92 83 91 -88 -123 33 117 45 115 -84 -113 96 -48 -80 21 -64 -27 107 10 17 -76 23 63
52 -122 29 29 -75 -112 96 -19 -23 -80 66 108 -60 -117 -68 -121 12 -118 -113 40 -88 120 -10 -91 -97 -104 3 -8 -118 -105 -63 115 -61 -36 -80 -68 27 44 76 1 29 16 10 -84
3 69 67 -56 -38 -94 -24 -28 -24 -110 -45 -28 69 43 109 75 -7 -97 48 -37 -128 -111 3 19 -12 58 -67 41 108 -71 -53 -1 -25 -29 -33 -38 -46 -80 29 73 -28 -45 -40 112 121
42 -34 -90 80 -46 -109 9 47 111 -29 54 -18 92 -86 60 -38 -26 -118 41 -7 5 3 -35 16 16 12 -95 98 -62 -99 37 1 77 26 27 -3 99 122 -93 -67 -68 -3 78 -123 -34 -64 -67 -116
93 -73 21 72 31 91 15 67 -75 -57 40 -84 -27 7 90 38 62 30 101 -27 48 -62 -25 26 -34 113 59 -24 -49 60 114 44

3  facebook.com 256 104 -49 101 -94 76 10 -102 -45 -77 -6 62 118 24 -104 -74 15 -49 103 -108 -96 85 -21 -95 48 75 -38 -117 -81 123 -48 126 -50 -40 -118 10 -61 52 73 -20
42 27 -123 -106 -17 -8 -76 -53 24 -100 -105 -55 -94 9 33 21 87 52 107 23 -58 -32 33 19 -74 116 -4 -42 123 93 12 21 7 7 -34 124 116 -107 19 -7 31 -20 105 7 85 -102 18
-97 -107 76 94 49 -57 -119 117 -97 -89 -97 -12 -122 -114 -70 8 -100 -16 18 111 -24 -26 -47 -67 45 -77 -108 -55 101 12 -111 -2 -128 97 -124 30 29 124 65 52 -82 -88 -99
-1 -75 6 11 97 -81 -4 89 79 88 124 -32 -125 -119 -88 -4 64 46 -89 -127 -75 -50 -96 -24 93 -16 -41 -120 -24 -4 97 124 -26 -59 -42 -56 -8 103 87 -10 -66 -126 48 119 40
62 79 116 -58 -61 -37 52 -34 50 -16 -6 -123 -87 124 -9 -6 36 -93 97 103 -89 106 64 -89 -50 -6 5 -94 99 65 -12 -71 -6 9 32 -106 108 118 -59 -32 -50 101 -63 -90 86 91 -7
18 -91 -22 48 100 -63 -60 110 -60 78 -16 -90 75 69 -99 -6 -114 3 86 -55 71 -92 -115 78 -117 86 88 -85 -120 -108 100 -7 -119 -82 6

4  samsung.com 257 0 -93 -97 -46 -85 125 -123 -98 110 -128 121 81 34 86 39 -32 -101 -32 40 -20 -101 -48 6 79 -45 89 -125 -4 -44 98 43 61 11 58 -85 41 110 12 -98 -122 124
33 -114 -101 -29 122 -63 -27 98 -27 -44 36 81 -29 75 93 80 58 -22 -7 88 -104 100 7 -29 5 -100 47 -108 -44 -112 -12 -123 -100 -82 124 -44 -122 59 98 62 -78 32 -10 -91
-104 121 7 112 114 -92 -106 -72 -93 -37 113 34 60 -54 12 126 -107 75 80 -50 -40 -100 -120 -15 107 95 -35 -37 -77 -5 -37 89 -79 63 31 -26 0 -126 93 27 -48 107 52 19 -8
99 106 2 -13 -39 -31 22 82 24 -14 21 71 -117 48 -32 -19 112 74 -81 15 -77 -73 -53 -98 -161 88 -62 -18 55 -63 -30 -21 118 -119 43 14 52 70 13 -122 45 68 -88 -81 46 -105
108 32 -89 90 111 125 -62 -56 73 -58 -40 70 54 -39 -117 102 -52 62 -25 -7 3 109 98 -111 -59 79 5 62 4 -77 -52 -68 39 11 -66 -52 -55 76 127 107 25 9 -24 7 54 64 126 124
63 -126 -100 5 -5 -24 43 -76 93 69 -53 -12 -117 -94 -108 -25 17 -19 -88 -25 12 105 -97 -127 0 -110 -9 -62 -90 71 -59 31 -98

5  google.com 257 0 -100 -62 -65 -109 79 -77 18 85 1 21 -46 89 94 -64 -108 110 39 -114 93 -87 42 3 40 65 -34 -9 -77 68 -77 71 1 -106 -20 -73 22 -60 107 -44 -115 -30 76 34
-107 44 -38 13 86 -117 -31 -58 104 -84 -118 -60 69 -42 58 -47 63 -9 4 -47 -113 -28 0 24 51 65 -87 -73 -94 2 -117 -103 -19 -33 -103 -1 -59 53 -50 -53 58 -40 80 38 9 76
-123 -62 51 114 -68 69 97 -14 -54 -8 98 -58 9 -27 37 13 -107 98 76 47 27 -79 -43 85 72 127 103 -17 25 -93 95 33 -108 -48 -74 88 -45 -74 -49 49 -2 -68 -125 -16 8 63 -62 -6
-46 -112 -65 103 -88 69 -92 -67 26 4 102 102 102 19 -34 26 10 -17 75 48 10 66 -115 -82 -20 -49 -21 88 -20 -64 -78 92 69 -14 78 58 -27 110 -122 -30 -55 -18 -92 -47 49
-92 62 49 24 -104 -1 -90 113 -27 -20 82 59 56 75 3 -12 -59 7 -13 105 -99 -67 2 -13 -1 35 83 21 76 33 -2 -115 -112 4 63 127 98 -105 92 -38 24 92 -65 -117 -87 31 9 20 -9
-5 -47 -99 -13 -76 -65 90 -113 21 -26 6 -120 -89 49 12 63 -110 84 116 -41 -51 5 75 45 -8 100 -108

6  samsung.com 257 0 -118 71 29 -92 -13 -126 -97 27 37 -63 19 -11 -28 -1 -127 -87 23 -126 2 -56 -105 73 -84 68 -1 -88 78 85 59 62 -114 117 85 45 -91 -91 22 -88 -27 33 -71
-6 70 -116 42 110 -60 -55 -127 112 -44 -124 15 -123 120 74 51 -118 16 -76 -118 85 39 37 -117 34 12 -98 76 55 9 -79 34 118 -29 -94 -100 -67 97 121 -119 -72 93 99 5 25
126 -64 86 27 -100 64 98 5 -85 73 45 50 79 -50 54 68 89 16 -12 41 -43 33 109 46 86 44 87 -40 10 25 -63 58 -24 -65 126 -113 17 38 122 106 21 -75 -70 16 -107 -30 65 -41
38 82 119 -117 77 -122 -126 45 -4 43 74 9 -1 38 -124 -124 -92 -66 -67 -127 116 50 -116 -43 33 -3 -118 34 78 -81 88 8 91 -70 -25 -73 -77 -82 -104 38 -103 88 -82 92 69
-94 120 101 -95 -11 100 22 127 104 -23 33 101 36 25 -110 -70 71 93 -38 48 100 -94 -93 -120 80 -67 57 80 6 -94 92 -58 -108 -56 -61 85 -102 31 -83 58 -42 21 -8 -10 120

```

TO STUDY AND WORK WITH KF SENSOR INTRUSION DETECTION TOOL. SETUP A HONEYPOT AND MONITOR THE HONEYPOT ON THE NETWORK.

Honey Pot is a device placed on Computer Network specifically designed to capture malicious network traffic. KF Sensor is the tool to setup as honeypot when KF Sensor is running it places a siren icon in the windows system tray in the bottom right of the screen. If there are no alerts then green icon is displayed.

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:

- Production honeypots
- 2. Research honeypots

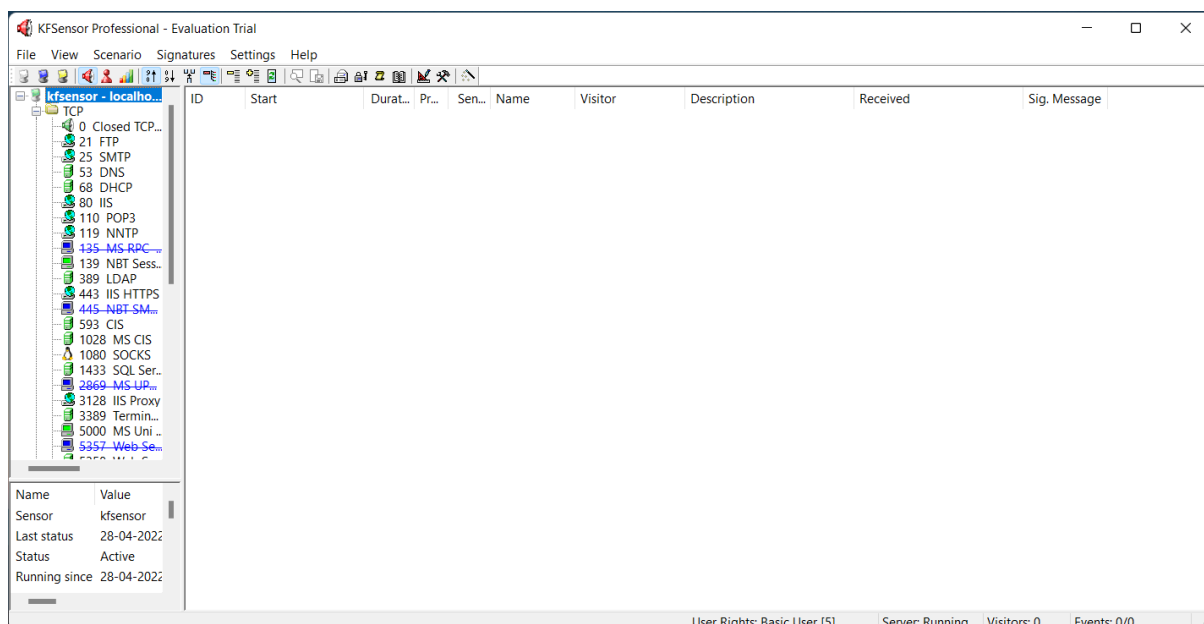
Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production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

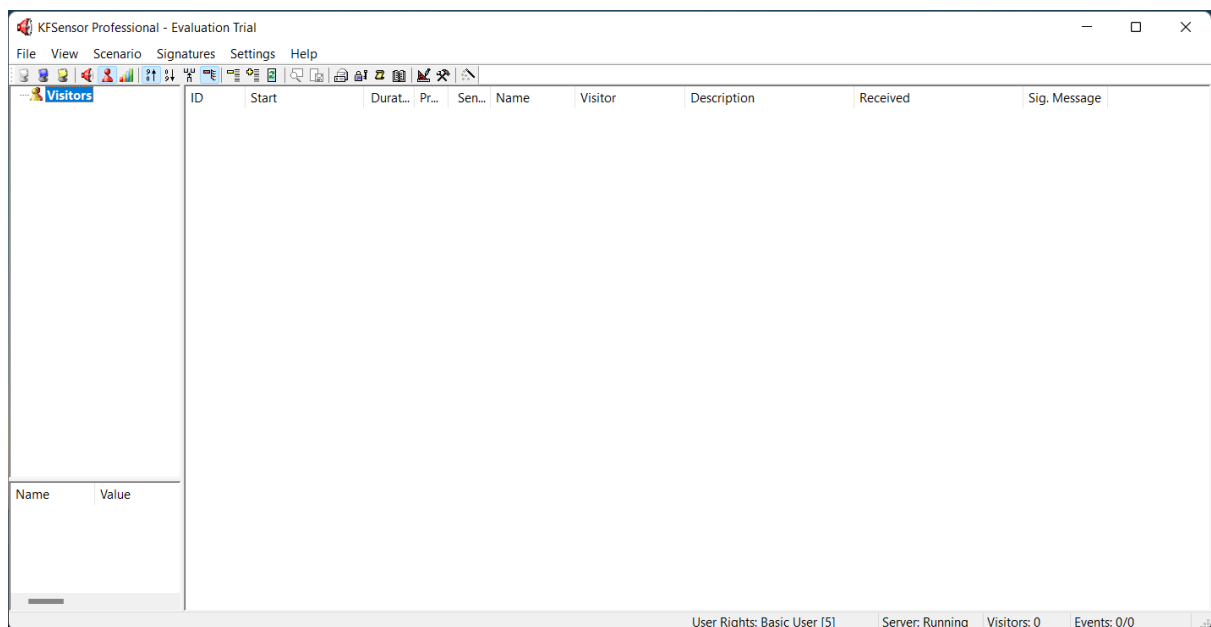
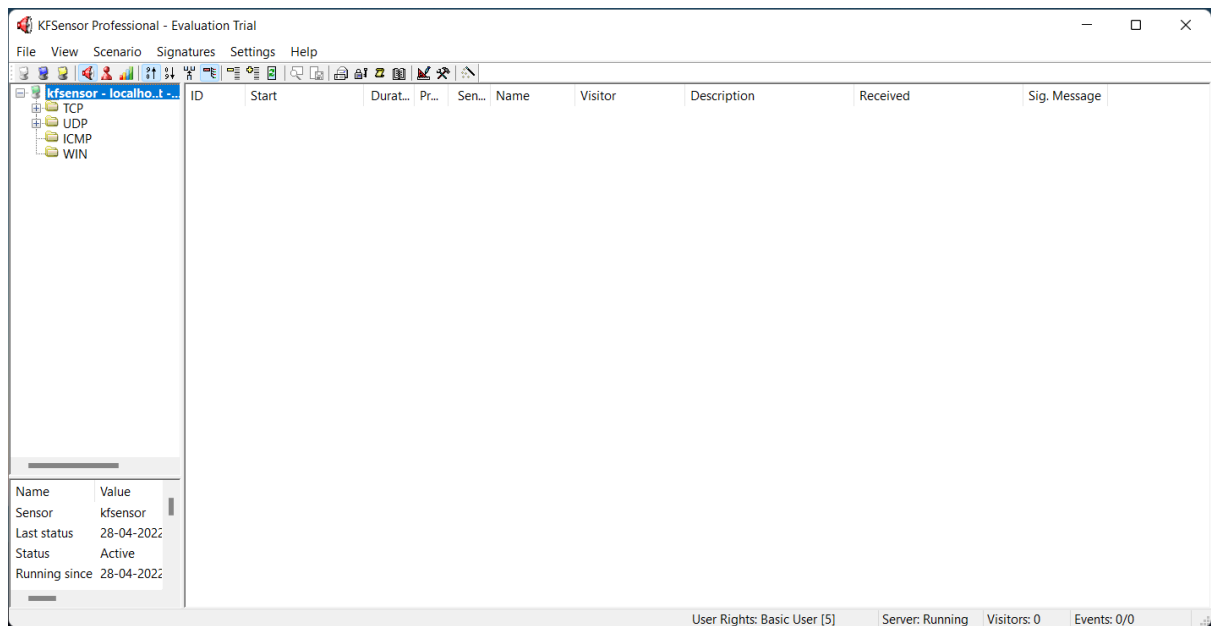
Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better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





KFSensor Professional - Evaluation Trial

File View Scenario Signatures Settings Help

Visitors

ID	Start	Durat...	Pr...	Sen...	Name	Visitor	Description	Received	Sig. Message
7	28-04-2022 12:08...	296.2...	TCP	443	IIS HTTPS	Manoj	Long running connection		
6	28-04-2022 12:08...	270.0...	TCP	443	IIS HTTPS	Manoj	Long running connection		
5	28-04-2022 12:13...	0.011	TCP	80	IIS	www.sublimetex...	url:GET /favicon.ico host:1...	GET /favicon.ico HTTP/1.1[...	
4	28-04-2022 12:13...	0.127	TCP	80	IIS	www.sublimetex...	url:GET /iisstart.png host:1...	GET /iisstart.png HTTP/1.1[...	
3	28-04-2022 12:13...	0.018	TCP	80	IIS	www.sublimetex...	url:GET / host:127.0.0.1] ag...	GET / HTTP/1.1[0D 0A]Hos...	
2	28-04-2022 12:08...	270.5...	TCP	5228	TCP Connec...	Manoj	Long running connection		
1	28-04-2022 12:08...	260.3...	TCP	443	IIS HTTPS	Manoj	Long running connection		

Name Value

User Rights: Basic User [5] Server: Attack Visitors: 2 Events: 7/7

KFSensor Professional - Evaluation Trial

File View Scenario Signatures Settings Help

Visitors

ID	Start	Durat...	Pr...	Sen...	Name	Visitor
9	28-04-2022 12:08...	270.0...	TCP	443	IIS HTTPS	Manoj
8	28-04-2022 12:08...	270.0...	TCP	443	IIS HTTPS	Manoj
7	28-04-2022 12:08...	296.2...	TCP	443	IIS HTTPS	Manoj
6	28-04-2022 12:08...	270.0...	TCP	443	IIS HTTPS	Manoj
5	28-04-2022 12:13...	0.011	TCP	80	IIS	www.sublimetex...
4	28-04-2022 12:13...	0.127	TCP	80	IIS	www.sublimetex...
3	28-04-2022 12:13...	0.018	TCP	80	IIS	www.sublimetex...
2	28-04-2022 12:08...	270.5...	TCP	5228	TCP Connec...	Manoj
1	28-04-2022 12:08...	260.3...	TCP	443	IIS HTTPS	Manoj

Name Value

Event - 9

Summary Details Signature Data

Event

Sensor ID: kfsensor Event ID: 9

Start 28-04-2022 12:08:36.977 Severity: High

Description: Long running connection

Visitor

IP: 192.168.1.5 Port: 64150

Domain: Manoj

Sensor

Name: IIS HTTPS

Protocol: TCP Port: 443

Signature

Message:

Request Data - 7 Bytes

Expand

Next Previous Close Help

Events: 9/9

TO WORK WITH SNORT TOOL TO DEMONSTRATE INTRUSION DETECTION SYSTEM. DOWNLOAD SNORT FROM SNORT.ORG.

Snort is an open-source network intrusion detection system (NIDS) and it is a packet sniffer that monitors network traffic in real time.

INTRUSION DETECTION SYSTEM:

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level.

Intrusion detection systems fall into two basic categories:

- Signature-based intrusion detection systems
- Anomaly detection systems.

Intruders have signatures, like computer viruses, that can be detected using software. You try to find data packets that contain any known intrusion-related signatures or anomalies related to Internet protocols. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts.

Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts. In some cases these methods produce better results compared to signature-based IDS. Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Snort is primarily a rule-based IDS, however input plug-ins are present to detect anomalies in protocol headers.

SNORT TOOL: Snort is based on libpcap (for library packet capture), a tool that is widely used in TCP/IP traffic sniffers and analyzers. Through protocol analysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to a pop-up window. Snort is currently the most popular free network intrusion detection software.

The advantages of Snort are numerous. According to the snort web site, "It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflow, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more" (Caswell).

One of the advantages of Snort is its ease of configuration. Rules are very flexible, easily written, and easily inserted into the rule base. If a new exploit or attack is found a rule for the attack can be added to the rule base in a matter of seconds. Another advantage of snort is that it allows for raw packet data analysis.

SNORT can be configured to run in three modes:

1. Sniffer mode
2. Packet Logger mode
3. Network Intrusion Detection System mode

1. Sniffer mode

- Snort -v Print out the TCP/IP packets header on the screen
- Snort -vd show the TCP/IP ICMP header with application data in transmit

2. Packet Logger mode

- snort -dev -l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.
- snort -dev -l c:\log -h ipaddress/24: This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory. snort -l c:\log -b This is binary mode logs everything into a single file.

3. Network Intrusion Detection System mode

- snort -d c:\log -h ipaddress/24 -c snort.conf This is a configuration file applies rule to each packet to decide it an action based upon the rule type in the file.
- Snort -d -h ipaddress/24 -l c:\log -c snort.conf This will cnfigure snort to run in its most basic NIDS form, logging packets that trigger rules speciefies in the snort.conf.

PROCEDURE:

STEP-1: Sniffer mode _____ snort -v _____ Print out the TCP/IP packets header on the screen.

STEP-2: Snort -vd _____ Show the TCP/IP ICMP header with application data in transit.

STEP-3: Packet Logger mode _____ snort -dev -l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.

STEP-4: snort -dev -l c:\log -h ipaddress/24 _____ This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory.

STEP-5: snort -l c:\log -b _____ this binary mode logs everything into a single file.

STEP-6: Network Intrusion Detection System mode _____ snort -d c:\log -h ipaddress/24 -c snort.conf _____ This is a configuration file that applies rule to each packet to decide it an action based upon the rule type in the file.

STEP-7: snort -d -h ip address/24 -l c:\log -c snort.conf _____ This will configure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.

STEP-8: Download SNORT from snort.org. Install snort with or without database support.

STEP-9: Select all the components and Click Next. Install and Close.

STEP-10: Skip the WinPcap driver installation.

STEP-11: Add the path variable in windows environment variable by selecting new classpath.

STEP-12: Create a path variable and point it at snort.exe variable name _____ path and variable value _____ c:\snort\bin.

STEP-13: Click OK button and then close all dialog boxes. Open command prompt and type the following commands

OUTPUT

```

C:\Snort> Command Prompt - snort.exe

20-10-2021 16:34          94,208 pcre.dll
12-11-2021 14:47        1,560,064 snort.exe
20-10-2021 16:34          53,326 WanPacket.dll
20-10-2021 16:34        208,974 wpcap.dll
20-10-2021 16:34        73,728 zlib1.dll
           8 File(s)    2,356,521 bytes
           2 Dir(s)    70,619,152,384 bytes free

C:\Snort\bin>snort.exe
Running in packet dump mode

---= Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{36B242D1-5017-455F-AAAA-BA80A2D98E5D}".
Decoding Ethernet

---= Initialization Complete ===

o"-
  '~
  '~~~
    -*> Snort! <*-
    Version 2.9.19-WIN64 GRE (Build 85)
    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
    Copyright (C) 2014-2021 Cisco and/or its affiliates. All rights reserved.
    Copyright (C) 1998-2013 Sourcefire, Inc., et al.
    Using PCRE version: 8.10 2010-06-25
    Using ZLIB version: 1.2.11

Commencing packet processing (pid=9144)

```

[illegible]

IMPLEMENT A CODE TO SIMULATE BUFFER OVERFLOW ATTACK.

```
#include<bits/stdc++.h>

int main(void)
{
    char buff[15];
    int pass = 0;

    printf("\n Enter the password :");
    gets(buff);

    if(strcmp(buff, "mypassword"))
    {
        printf ("Wrong Password \n");
    }
    else
    {
        printf ("Correct Password \n");
        pass = 1;
    }

    if(pass)
    {
        printf ("You are logged in as an administrator\n");
    }

    return 0;
}
```

OUTPUT

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Enter the password :hello

Wrong Password

PS C:\Users\alpna\Desktop\6TH SEM\IS\codes> cd "c:\Users\alpna\Desktop\6TH SEM\IS\codes\" ; if (\$?) { gcc low_attack }

Enter the password :password

Correct Password

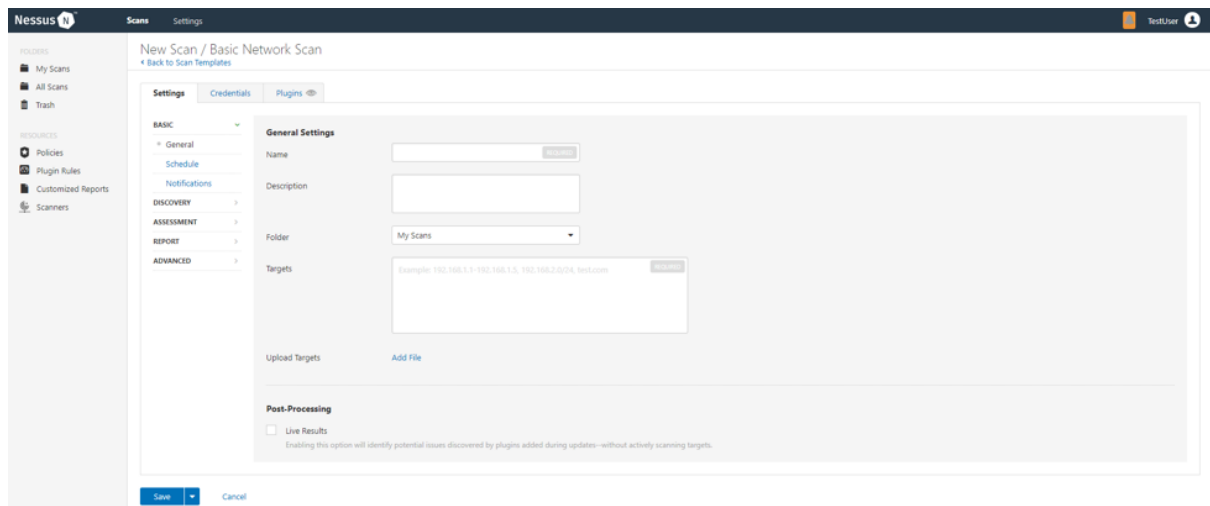
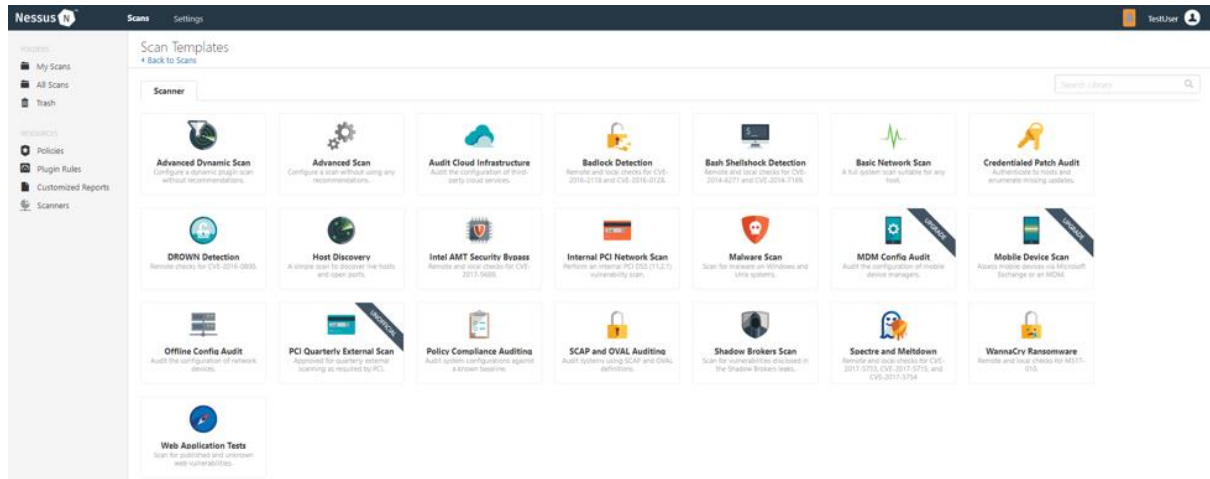
You are logged in as an administrator

PS C:\Users\alpna\Desktop\6TH SEM\IS\codes> cd "c:\Users\alpna\Desktop\6TH SEM\IS\codes\" ; if (\$?) { gcc low_attack }

Enter the password :asdf

Wrong Password

USE THE NESSUS TOOL TO SCAN THE NETWORK FOR VULNERABILITIES



Nessus

Scans

Settings

TestUser

My Scans

All Scans

Trash

Policies

Plugin Rules

Customized Reports

Scanners

New Scan / Basic Network Scan

Back to Scan Templates

Settings

Credentials

Plugins

CATEGORIES

HOST

Filter Credentials

SSH

Windows

Save

Cancel

Basic Network

[Back to My Scans](#)

Configure

Audit Trail

Launch

Export

Hosts 1

Vulnerabilities 66

Remediations 2

History 1

Filter

Search Vulnerabilities

66 Vulnerabilities

Sev	Name	Family	Count		
CRITICAL	Jenkins < 2.46.2 / 2.57 and Je...	CGI abuses	1		
CRITICAL	MS17-010: Security Update f...	Windows	1		
HIGH	Jenkins < 2.121.2 / 2.133 Mul...	CGI abuses	1		
HIGH	Jenkins < 2.138.4 LTS / 2.150...	CGI abuses	1		
HIGH	Jenkins < 2.150.2 LTS / 2.160 ...	CGI abuses	1		
HIGH	MS12-020: Vulnerabilities in ...	Windows	1		
MEDIUM	Jenkins < 2.107.2 / 2.116 Mul...	CGI abuses	1		
MEDIUM	Jenkins < 2.121.3 / 2.138 Mul...	CGI abuses	1		
MEDIUM	Jenkins < 2.138.2 / 2.146 Mul...	CGI abuses	1		
MEDIUM	Jenkins < 2.73.3 / 2.89 Multip...	CGI abuses	1		
MEDIUM	Jenkins < 2.89.2 / 2.95 Multip...	CGI abuses	1		
MEDIUM	Jenkins < 2.89.4 / 2.107 Multi...	CGI abuses	1		
MEDIUM	Microsoft Windows Remote ...	Windows	1		

Scan Details

Name: Basic Network
Status: Completed
Policy: Basic Network Scan
Scanner: Local Scanner
Start: February 25 at 9:03 AM
End: February 25 at 9:07 AM
Elapsed: 4 minutes

Vulnerabilities

