**CS 685**

**COMPUTER SECURITY**

**APRIL 19,2016**

**PREPARED BY:**

**Akash Rajendra Ventekar**

**Himanshu Koyalkar**

**Poornimadevi Palanisamy**

**PREPARED FOR:**

**Dr.Feng Zhu**

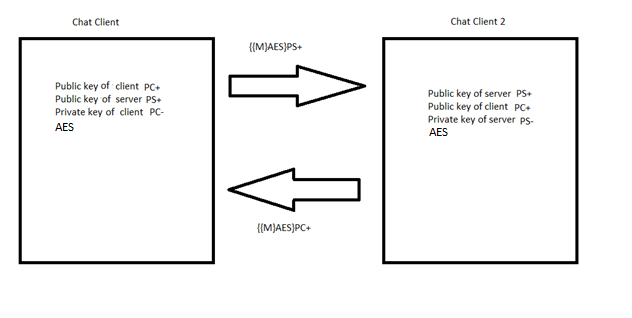
**SPRING 2016**

**CONTENT:**

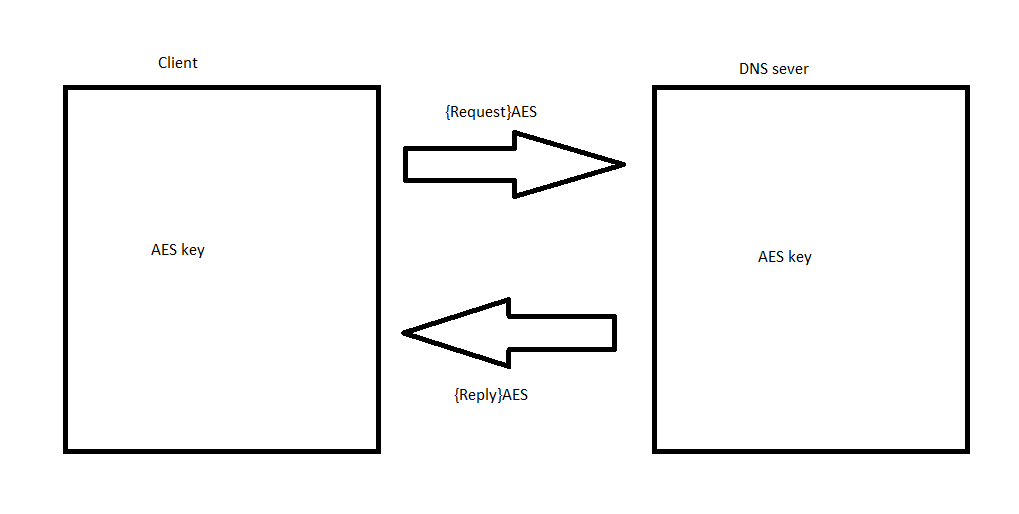
|  |  |
| --- | --- |
| **CONTENT** | **PAGE NO** |
| **BLOCK DIAGRAM** | **3** |
| **PROJECT DESCRIPTION** | **4** |
| **CUSTOM AES IMPLEMENTATION** | **5** |
| **RSA IN BUILT IMPLEMENTATION** | **15** |
| **DIGITAL SIGNATURE USING RSA** | **21** |
| **AES IN BUILT IMPLEMENTATION** | **22** |
| **SOCKET IMPLEMENTATION FOR**  **AES AND RSA ENCRYPTION AND DECRYPTION** | **25** |
| **HASH FUNCTION** | **30** |
| **BONUS IMPLEMENTATION:**  **DNS SERVER WITH ENCRYPTION** | **31** |
| **PERFORMANCE EVALUATION** | **42** |

**Block diagram:**

**Chat Application:**



**DNS implementation:**



**PROJECT DESCRIPTION:**

We have developed 4 separate implementations.

1. AES in-built implementation
2. RSA in-built implementation
3. Custom AES implementation
4. Custom RSA implementation

We have a chat application which uses our in-built AES implementation and custom RSA implementation.

**Application Layer:** Chat application

**Communication Layer:** This part is coded using Java sockets.

**Cryptographic Algorithm Layer:** RSA 1024 and AES 128 bit key for encryption and decryption

In addition, we have a DNS server implementation which uses in-built implementation.

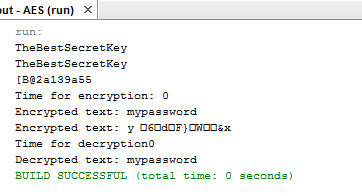
**Application Layer:** DNS server ( Forward and reverse DNS queries)

**Communication Layer:** This part is coded using Java sockets.

**Cryptographic Algorithm Layer:** AES 128 bit key for encryption and decryption

**Custom AES implementation:**

**Output:**



**AES.java: Source code for AES implementation**

package aes;

public class AES {

private static int Nb, Nk, Nr;

private static byte[][] w;

//Sbox initialization

private static int[] sbox = { 0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F,

0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82,

0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C,

0xA4, 0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,

0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15, 0x04, 0xC7, 0x23,

0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27,

0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52,

0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84, 0x53, 0xD1, 0x00, 0xED,

0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58,

0xCF, 0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9,

0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92,

0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,

0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E,

0x3D, 0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A,

0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB, 0xE0,

0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62,

0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E,

0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08, 0xBA, 0x78,

0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B,

0xBD, 0x8B, 0x8A, 0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E,

0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E, 0xE1, 0xF8, 0x98,

0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55,

0x28, 0xDF, 0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41,

0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16 };

//Inverse Sbox initialization

private static int[] inv\_sbox = { 0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5,

0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB, 0x7C, 0xE3,

0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4,

0xDE, 0xE9, 0xCB, 0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D,

0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E, 0x08, 0x2E, 0xA1,

0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B,

0xD1, 0x25, 0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4,

0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92, 0x6C, 0x70, 0x48, 0x50,

0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D,

0x84, 0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4,

0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06, 0xD0, 0x2C, 0x1E, 0x8F, 0xCA,

0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,

0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF,

0xCE, 0xF0, 0xB4, 0xE6, 0x73, 0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD,

0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E, 0x47,

0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E,

0xAA, 0x18, 0xBE, 0x1B, 0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79,

0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4, 0x1F, 0xDD,

0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27,

0x80, 0xEC, 0x5F, 0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D,

0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF, 0xA0, 0xE0, 0x3B,

0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53,

0x99, 0x61, 0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1,

0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D };

private static int Rcon[] = { 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,

0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,

0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,

0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,

0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,

0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,

0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,

0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,

0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,

0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,

0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,

0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,

0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,

0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,

0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,

0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb };

//Method to XOR bytes

private static byte[] xor\_func(byte[] a, byte[] b) {

byte[] out = new byte[a.length];

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

out[i] = (byte) (a[i] ^ b[i]);

}

return out;

}

//Method for generation of subkeys

private static byte[][] generateSubkeys(byte[] key) {

byte[][] tmp = new byte[Nb \* (Nr + 1)][4];

int i = 0;

while (i < Nk) {

tmp[i][0] = key[i \* 4];

tmp[i][1] = key[i \* 4 + 1];

tmp[i][2] = key[i \* 4 + 2];

tmp[i][3] = key[i \* 4 + 3];

i++;

}

i = Nk;

while (i < Nb \* (Nr + 1)) {

byte[] temp = new byte[4];

for(int k = 0;k<4;k++)

temp[k] = tmp[i-1][k];

if (i % Nk == 0) {

temp = SubWord(rotateWord(temp));

temp[0] = (byte) (temp[0] ^ (Rcon[i / Nk] & 0xff));

} else if (Nk > 6 && i % Nk == 4) {

temp = SubWord(temp);

}

tmp[i] = xor\_func(tmp[i - Nk], temp);

i++;

}

return tmp;

}

//Substitution of each word

private static byte[] SubWord(byte[] in) {

byte[] tmp = new byte[in.length];

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

tmp[i] = (byte) (sbox[in[i] & 0x000000ff] & 0xff);

return tmp;

}

//Shift left operation

private static byte[] rotateWord(byte[] input) {

byte[] tmp = new byte[input.length];

tmp[0] = input[1];

tmp[1] = input[2];

tmp[2] = input[3];

tmp[3] = input[0];

return tmp;

}

//Method for adding rounding key

private static byte[][] AddRoundKey(byte[][] state, byte[][] w, int round) {

byte[][] tmp = new byte[state.length][state[0].length];

for (int c = 0; c < Nb; c++) {

for (int l = 0; l < 4; l++)

tmp[l][c] = (byte) (state[l][c] ^ w[round \* Nb + c][l]);

}

return tmp;

}

//Method for byte substitution

private static byte[][] SubBytes(byte[][] state) {

byte[][] tmp = new byte[state.length][state[0].length];

for (int row = 0; row < 4; row++)

for (int col = 0; col < Nb; col++)

tmp[row][col] = (byte) (sbox[(state[row][col] & 0x000000ff)] & 0xff);

return tmp;

}

//Inverse byte substitution

private static byte[][] InvSubBytes(byte[][] state) {

for (int row = 0; row < 4; row++)

for (int col = 0; col < Nb; col++)

state[row][col] = (byte)(inv\_sbox[(state[row][col] & 0x000000ff)]&0xff);

return state;

}

//Method to shift rows

private static byte[][] ShiftRows(byte[][] state) {

byte[] t = new byte[4];

for (int r = 1; r < 4; r++) {

for (int c = 0; c < Nb; c++)

t[c] = state[r][(c + r) % Nb];

for (int c = 0; c < Nb; c++)

state[r][c] = t[c];

}

return state;

}

//Inverse shift of rows

private static byte[][] InvShiftRows(byte[][] state) {

byte[] t = new byte[4];

for (int r = 1; r < 4; r++) {

for (int c = 0; c < Nb; c++)

t[(c + r)%Nb] = state[r][c];

for (int c = 0; c < Nb; c++)

state[r][c] = t[c];

}

return state;

}

//Inverse of mix columns

private static byte[][] InvMixColumns(byte[][] s){

int[] sp = new int[4];

byte b02 = (byte)0x0e, b03 = (byte)0x0b, b04 = (byte)0x0d, b05 = (byte)0x09;

for (int c = 0; c < 4; c++) {

sp[0] = FFMul(b02, s[0][c]) ^ FFMul(b03, s[1][c]) ^ FFMul(b04,s[2][c]) ^ FFMul(b05,s[3][c]);

sp[1] = FFMul(b05, s[0][c]) ^ FFMul(b02, s[1][c]) ^ FFMul(b03,s[2][c]) ^ FFMul(b04,s[3][c]);

sp[2] = FFMul(b04, s[0][c]) ^ FFMul(b05, s[1][c]) ^ FFMul(b02,s[2][c]) ^ FFMul(b03,s[3][c]);

sp[3] = FFMul(b03, s[0][c]) ^ FFMul(b04, s[1][c]) ^ FFMul(b05,s[2][c]) ^ FFMul(b02,s[3][c]);

for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);

}

return s;

}

//Method to mix columns

private static byte[][] MixColumns(byte[][] s){

int[] sp = new int[4];

byte b02 = (byte)0x02, b03 = (byte)0x03;

for (int c = 0; c < 4; c++) {

sp[0] = FFMul(b02, s[0][c]) ^ FFMul(b03, s[1][c]) ^ s[2][c] ^ s[3][c];

sp[1] = s[0][c] ^ FFMul(b02, s[1][c]) ^ FFMul(b03, s[2][c]) ^ s[3][c];

sp[2] = s[0][c] ^ s[1][c] ^ FFMul(b02, s[2][c]) ^ FFMul(b03, s[3][c]);

sp[3] = FFMul(b03, s[0][c]) ^ s[1][c] ^ s[2][c] ^ FFMul(b02, s[3][c]);

for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);

}

return s;

}

//For multiplying

public static byte FFMul(byte a, byte b) {

byte aa = a, bb = b, r = 0, t;

while (aa != 0) {

if ((aa & 1) != 0)

r = (byte) (r ^ bb);

t = (byte) (bb & 0x80);

bb = (byte) (bb << 1);

if (t != 0)

bb = (byte) (bb ^ 0x1b);

aa = (byte) ((aa & 0xff) >> 1);

}

return r;

}

//AES block encryption

public static byte[] encryptBloc(byte[] in) {

byte[] tmp = new byte[in.length];

byte[][] state = new byte[4][Nb];

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

state[i / 4][i % 4] = in[i%4\*4+i/4];

state = AddRoundKey(state, w, 0);

for (int round = 1; round < Nr; round++) {

state = SubBytes(state);

state = ShiftRows(state);

state = MixColumns(state);

state = AddRoundKey(state, w, round);

}

state = SubBytes(state);

state = ShiftRows(state);

state = AddRoundKey(state, w, Nr);

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

tmp[i%4\*4+i/4] = state[i / 4][i%4];

return tmp;

}

//AES block decryption

public static byte[] decryptBloc(byte[] in) {

byte[] tmp = new byte[in.length];

byte[][] state = new byte[4][Nb];

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

state[i / 4][i % 4] = in[i%4\*4+i/4];

state = AddRoundKey(state, w, Nr);

for (int round = Nr-1; round >=1; round--) {

state = InvSubBytes(state);

state = InvShiftRows(state);

state = AddRoundKey(state, w, round);

state = InvMixColumns(state);

}

state = InvSubBytes(state);

state = InvShiftRows(state);

state = AddRoundKey(state, w, 0);

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

tmp[i%4\*4+i/4] = state[i / 4][i%4];

return tmp;

}

//AES encryption

public static byte[] encrypt(byte[] in,byte[] key){

Nb = 4;

Nk = key.length/4;

Nr = Nk + 6;

int lenght=0;

byte[] padding = new byte[1];

int i;

lenght = 16 - in.length % 16;

padding = new byte[lenght];

padding[0] = (byte) 0x80;

for (i = 1; i < lenght; i++)

padding[i] = 0;

byte[] tmp = new byte[in.length + lenght];

byte[] bloc = new byte[16];

w = generateSubkeys(key);

int count = 0;

for (i = 0; i < in.length + lenght; i++) {

if (i > 0 && i % 16 == 0) {

bloc = encryptBloc(bloc);

System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);

}

if (i < in.length)

bloc[i % 16] = in[i];

else{

bloc[i % 16] = padding[count % 16];

count++;

}

}

if(bloc.length == 16){

bloc = encryptBloc(bloc);

System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);

}

return tmp;

}

//AES decryption

public static byte[] decrypt(byte[] in,byte[] key){

int i;

byte[] tmp = new byte[in.length];

byte[] bloc = new byte[16];

Nb = 4;

Nk = key.length/4;

Nr = Nk + 6;

w = generateSubkeys(key);

for (i = 0; i < in.length; i++) {

if (i > 0 && i % 16 == 0) {

bloc = decryptBloc(bloc);

System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);

}

if (i < in.length)

bloc[i % 16] = in[i];

}

bloc = decryptBloc(bloc);

System.arraycopy(bloc, 0, tmp, i - 16, bloc.length);

tmp = deletePadding(tmp);

return tmp;

}

//Remove padding

private static byte[] deletePadding(byte[] input) {

int count = 0;

int i = input.length - 1;

while (input[i] == 0) {

count++;

i--;

}

byte[] tmp = new byte[input.length - count - 1];

System.arraycopy(input, 0, tmp, 0, tmp.length);

return tmp;

}

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

//Message to be sent

String password = "mypassword";

byte[] in = password.getBytes();

byte[] keyValue =

new byte[] { 'T', 'h', 'e', 'B', 'e', 's', 't',

'S', 'e', 'c', 'r','e', 't', 'K', 'e', 'y' };

System.out.println(new String(keyValue));

String asd=new String(keyValue);

System.out.println(new String(asd.getBytes()));

System.out.println(in);

int start=(int) System.currentTimeMillis();

byte enc[]=encrypt(in,keyValue);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for encryption: "+ (stop-start));

System.out.println("Encrypted text: "+new String(in));

System.out.println("Encrypted text: "+new String(enc));

start=(int) System.currentTimeMillis();

byte dec[]=decrypt(enc, keyValue);

stop=(int) System.currentTimeMillis();

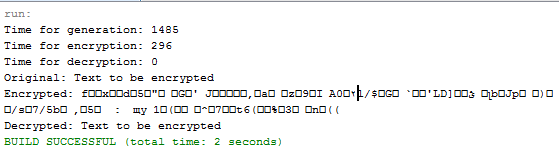
System.out.println("Time for decryption"+ (stop-start));

System.out.println("Decrypted text: "+new String(dec));

}

}

**RSA in-built implementation:**



**EncryptionUtil.java:Source code for in-built implementation of RSA**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package encryptionutil;

/\*\*

\*

\* @author Akash, Himanshu and Poornima

\*/

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.securiy.KeyPair;

import java.security.KeyPairGenerator;

import java.security.NoSuchAlgorithmException;

import java.security.PrivateKey;

import java.security.PublicKey;

import javax.crypto.Cipher;

public class EncryptionUtil {

/\*\*

\* String to hold name of the encryption algorithm.

\*/

public static final String ALGORITHM = "RSA";

/\*\*

\* String to hold the name of the private key file.

\*/

public static final String PRIVATE\_KEY\_FILE = "D:/keys4/private.key";

/\*\*

\* String to hold name of the public key file.

\*/

public static final String PUBLIC\_KEY\_FILE = "D:/keys4/public.key";

/\*\*

\* Generate key which contains a pair of private and public key using 1024

\* bytes. Store the set of keys in Private key and Public key files.

\*

\* @throws NoSuchAlgorithmException

\* @throws IOException

\* @throws FileNotFoundException

\*/

public static void generateKey() {

try {

int start=(int) System.currentTimeMillis();

final KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ALGORITHM);

keyGen.initialize(1024);

final KeyPair key = keyGen.generateKeyPair();

int stop=(int) System.currentTimeMillis();

System.out.println("Time for generation: "+ (stop-start));

File privateKeyFile = new File(PRIVATE\_KEY\_FILE);

File publicKeyFile = new File(PUBLIC\_KEY\_FILE);

// Create files to store public and private key

if (privateKeyFile.getParentFile() != null) {

privateKeyFile.getParentFile().mkdirs();

}

privateKeyFile.createNewFile();

if (publicKeyFile.getParentFile() != null) {

publicKeyFile.getParentFile().mkdirs();

}

publicKeyFile.createNewFile();

// Saving the Public key in a file

ObjectOutputStream publicKeyOS = new ObjectOutputStream(

new FileOutputStream(publicKeyFile));

publicKeyOS.writeObject(key.getPublic());

publicKeyOS.close();

// Saving the Private key in a file

ObjectOutputStream privateKeyOS = new ObjectOutputStream(

new FileOutputStream(privateKeyFile));

privateKeyOS.writeObject(key.getPrivate());

privateKeyOS.close();

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* The method checks if the pair of public and private key has been generated.

\*

\* @return flag indicating if the pair of keys were generated.

\*/

public static boolean areKeysPresent() {

File privateKey = new File(PRIVATE\_KEY\_FILE);

File publicKey = new File(PUBLIC\_KEY\_FILE);

if (privateKey.exists() && publicKey.exists()) {

return true;

}

return false;

}

/\*\*

\* Encrypt the plain text using public key.

\*

\* @param text

\* : original plain text

\* @param key

\* :The public key

\* @return Encrypted text

\* @throws java.lang.Exception

\*/

public static byte[] encrypt(String text, PublicKey key) {

byte[] cipherText = null;

try {

// get an RSA cipher object and print the provider

final Cipher cipher = Cipher.getInstance(ALGORITHM);

// encrypt the plain text using the public key

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

cipherText = cipher.doFinal(text.getBytes());

} catch (Exception e) {

e.printStackTrace();

}

return cipherText;

}

/\*\*

\* Decrypt text using private key.

\*

\* @param text

\* :encrypted text

\* @param key

\* :The private key

\* @return plain text

\* @throws java.lang.Exception

\*/

public static String decrypt(byte[] text, PrivateKey key) {

byte[] dectyptedText = null;

try {

// get an RSA cipher object and print the provider

final Cipher cipher = Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key

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

dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {

ex.printStackTrace();

}

return new String(dectyptedText);

}

/\*\*

\* Test the EncryptionUtil

\*/

public static void main(String[] args) {

try {

// Check if the pair of keys are present else generate those.

if (!areKeysPresent()) {

// Method generates a pair of keys using the RSA algorithm and stores it

// in their respective files

generateKey();

}

final String originalText = "Text to be encrypted ";

ObjectInputStream inputStream = null;

// Encrypt the string using the public key

int start=(int) System.currentTimeMillis();

inputStream = new ObjectInputStream(new FileInputStream(PUBLIC\_KEY\_FILE));

final PublicKey publicKey = (PublicKey) inputStream.readObject();

final byte[] cipherText = encrypt(originalText, publicKey);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for encryption: "+ (stop-start));

// Decrypt the cipher text using the private key.

start=(int) System.currentTimeMillis();

inputStream = new ObjectInputStream(new FileInputStream(PRIVATE\_KEY\_FILE));

final PrivateKey privateKey = (PrivateKey) inputStream.readObject();

final String plainText = decrypt(cipherText, privateKey);

stop=(int) System.currentTimeMillis();

System.out.println("Time for decryption: "+ (stop-start));

// Printing the Original, Encrypted and Decrypted Text

System.out.println("Original: " + originalText);

System.out.println("Encrypted: " +new String(cipherText));

System.out.println("Decrypted: " + plainText);

} catch (Exception e) {

e.printStackTrace();

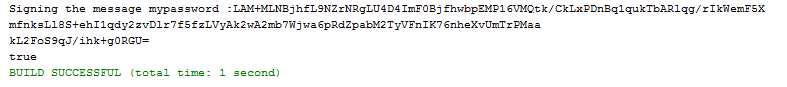
}

}

}

**Digital signature using RSA:**

**Output:**



**MainClass.java: Use private key to encrypt and verify using the public key**

package mainclass;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.Signature;

import sun.misc.BASE64Encoder;

//Reference: http://www.java2s.com/Tutorial/Java/0490\_\_Security/RSASignatureGeneration.htm

public class MainClass {

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

//Initialize and generate RSA 1024 key pair

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

kpg.initialize(1024);

KeyPair keyPair = kpg.genKeyPair();

//Text to encode using private key

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

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

//Sign using the private key

sig.initSign(keyPair.getPrivate());

sig.update(data);

byte[] signatureBytes = sig.sign();

System.out.println("Signing the message "+new String(data)+" :" + new BASE64Encoder().encode(signatureBytes));

//Verify the data using public key

sig.initVerify(keyPair.getPublic());

sig.update(data);

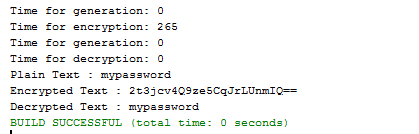
System.out.println(sig.verify(signatureBytes));

}

}

**AES in-built implementation:**

Output:



**AESMain.java: Source code for in-built AES implementation**

package aesmain;

/\*\*

\*

\* @author Akash

\*/

public class AESMain {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

String password = "mypassword";

int start=(int) System.currentTimeMillis();

String passwordEnc = AESencrp.encrypt(password);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for encryption: "+ (stop-start));

start=(int) System.currentTimeMillis();

String passwordDec = AESencrp.decrypt(passwordEnc);

stop=(int) System.currentTimeMillis();

System.out.println("Time for decryption: "+ (stop-start));

System.out.println("Plain Text : " + password);

System.out.println("Encrypted Text : " + passwordEnc);

System.out.println("Decrypted Text : " + passwordDec);

}

}

**AES Encrp.java: Generation, encrypt and decrypt method in AES**

package aesmain;

/\*\*

\*

\* @author Akash

\*/

import java.security.\*;

import java.security.spec.InvalidKeySpecException;

import javax.crypto.\*;

import javax.crypto.spec.SecretKeySpec;

import sun.misc.\*;

public class AESencrp {

private static final String ALGO = "AES";

private static final byte[] keyValue =

new byte[] { 'T', 'h', 'e', 'B', 'e', 's', 't',

'S', 'e', 'c', 'r','e', 't', 'K', 'e', 'y' };

public static String encrypt(String Data) throws Exception {

Key key = generateKey();

Cipher c = Cipher.getInstance(ALGO);

c.init(Cipher.ENCRYPT\_MODE, key);

byte[] encVal = c.doFinal(Data.getBytes());

String encryptedValue = new BASE64Encoder().encode(encVal);

return encryptedValue;

}

public static String decrypt(String encryptedData) throws Exception {

Key key = generateKey();

Cipher c = Cipher.getInstance(ALGO);

c.init(Cipher.DECRYPT\_MODE, key);

byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedData);

byte[] decValue = c.doFinal(decordedValue);

String decryptedValue = new String(decValue);

return decryptedValue;

}

private static Key generateKey() throws Exception {

int start=(int) System.currentTimeMillis();

Key key = new SecretKeySpec(keyValue, ALGO);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for generation: "+(stop-start));

return key;

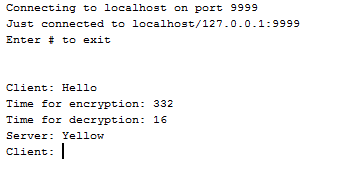
}

}

**Socket implementation for RSA and AES encryption decryption:**

**Client Implementation:**

**Output:**



**Client1.java: uses AESEncrp.jav a for AES encyption and decryption.**

package client1;

import java.net.\*;

import java.io.\*;

import java.security.\*;

import java.math.BigInteger;

import java.security.SecureRandom;

import java.security.NoSuchAlgorithmException;

class Rsa

{

private BigInteger n, d, e;

public Rsa(int bitlen) throws NoSuchAlgorithmException

{

//SecureRandom r = SecureRandom.getInstance("76ce8e44");

BigInteger p = new BigInteger("7436443145513102759346082091504489018965649844427578539182610598215255526295861138810419680153488731851639356319558363771772272791601047612382871111278903"

);

BigInteger q = new BigInteger("11715268389605343181133862358547399940348888325990729438639194032209526141405821547230226288689002408289433770695761015700029788589235283111801996922104979");

n = p.multiply(q);

//System.out.println("Value of r is "+r);

BigInteger m = (p.subtract(BigInteger.ONE))

.multiply(q.subtract(BigInteger.ONE));

e = new BigInteger("3");

while(m.gcd(e).intValue() > 1) e = e.add(new BigInteger("2"));

d = e.modInverse(m);

}

public String encrypt(String message) {

return (new BigInteger(message.getBytes())).modPow(e, n).toString();

}

public String decrypt(String message) {

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

}

public BigInteger encrypt(BigInteger message)

{

return message.modPow(e, n);

}

public BigInteger decrypt(BigInteger message)

{

return message.modPow(d, n);

}

}

public class Client1

{

public static String MD5(String text) throws NoSuchAlgorithmException{

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

m.update(text.getBytes(),0,text.length());

return (new BigInteger(1,m.digest()).toString(16));

}

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

{

Rsa a = new Rsa(1024);

String serverName = args[0];

int port = Integer.parseInt(args[1]);

try

{

System.out.println("Connecting to " + serverName +

" on port " + port);

Socket client = new Socket(serverName, port);

System.out.println("Just connected to "

+ client.getRemoteSocketAddress());

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

OutputStream os = client.getOutputStream();

DataOutputStream out = new DataOutputStream(os);

String s;

System.out.println("Enter # to exit\n\n");

do{

System.out.print("Client: ");

s = br.readLine();

/\* String myMD5 = MD5(s);

System.out.println("MD5: "+myMD5);

s=s+"+"+myMD5;\*/

int start=(int) System.currentTimeMillis();

s = AESencrp.encrypt(s);

String enc = a.encrypt(s);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for encryption: "+ (stop-start));

out.writeUTF(enc);

InputStream inFromServer = client.getInputStream();

DataInputStream in =

new DataInputStream(inFromServer);

String t = in.readUTF();

start=(int) System.currentTimeMillis();

String dec = a.decrypt(t);

dec = AESencrp.decrypt(dec);

stop=(int) System.currentTimeMillis();

System.out.println("Time for decryption: "+ (stop-start));

System.out.println("Server: " + dec);

}while(!s.equals('#'));

client.close();

}catch(IOException e)

{

e.printStackTrace();

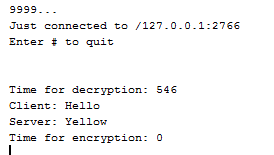
}

}

}

**Server implementation:**

**Output:**



**SecServer1.java:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package secserver1;

import java.security.NoSuchAlgorithmException;

import java.security.spec.RSAPrivateKeySpec;

import java.security.spec.RSAPublicKeySpec;

import java.net.\*;

import java.io.\*;

import java.security.\*;

import java.math.BigInteger;

import java.security.SecureRandom;

class Rsa

{

private BigInteger n, d, e;

public Rsa(int bitlen) throws NoSuchAlgorithmException

{

//SecureRandom r = new SecureRandom();

//System.out.println("r value is "+r);

BigInteger p = new BigInteger("7436443145513102759346082091504489018965649844427578539182610598215255526295861138810419680153488731851639356319558363771772272791601047612382871111278903" );

BigInteger q = new BigInteger("11715268389605343181133862358547399940348888325990729438639194032209526141405821547230226288689002408289433770695761015700029788589235283111801996922104979");

//System.out.println("Values of P and Q are "+p+"\n"+q);

//System.out.println("Value of r is "+r);

n = p.multiply(q);

BigInteger m = (p.subtract(BigInteger.ONE))

.multiply(q.subtract(BigInteger.ONE));

e = new BigInteger("3");

while(m.gcd(e).intValue() > 1) e = e.add(new BigInteger("2"));

d = e.modInverse(m);

}

public String encrypt(String message) {

return (new BigInteger(message.getBytes())).modPow(e, n).toString();

}

public String decrypt(String message) {

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

}

public BigInteger encrypt(BigInteger message)

{

return message.modPow(e, n);

}

public BigInteger decrypt(BigInteger message)

{

return message.modPow(d, n);

}

}

class Secserver1{

public static String MD5(String text) throws NoSuchAlgorithmException{

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

m.update(text.getBytes(),0,text.length());

System.out.println("MD5: "+new BigInteger(1,m.digest()).toString(16));

return (new BigInteger(1,m.digest()).toString(16));

}

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

{

Rsa a = new Rsa(1024);

try{

int port = Integer.parseInt(args[0]);

ServerSocket serverSocket = new ServerSocket(port);

System.out.println(serverSocket.getLocalPort() + "...");

Socket server = serverSocket.accept();

System.out.println("Just connected to "

+ server.getRemoteSocketAddress());

DataInputStream in =

new DataInputStream(server.getInputStream());

DataOutputStream sout =

new DataOutputStream(server.getOutputStream());

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

String s;

System.out.println("Enter # to quit\n\n");

String f,g;

do

{

String t=in.readUTF();

if(t.length()>1)

{

//t=t+"+"+MD5(t);

int start=(int) System.currentTimeMillis();

f = a.decrypt(t);

f = AESencrp.decrypt(f);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for decryption: "+ (stop-start));

System.out.println("Client: "+f);

System.out.print("Server: ");

}

s = br.readLine();

int start=(int) System.currentTimeMillis();

s = AESencrp.encrypt(s);

g = a.encrypt(s);

int stop=(int) System.currentTimeMillis();

System.out.println("Time for encryption: "+ (stop-start));

sout.writeUTF(g);

}while(!s.equals('#'));

server.close();

}

catch(Exception e){

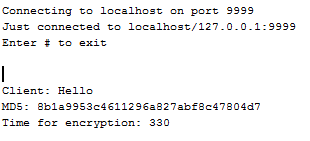
System.out.println(e);

}}}

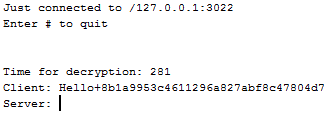
**Hash function:**

**Output:**

**At client:**



**At server:**



public static String MD5(String text) throws NoSuchAlgorithmException{

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

m.update(text.getBytes(),0,text.length());

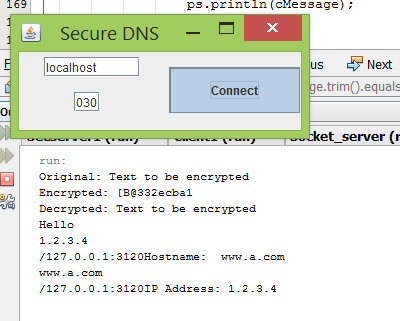
System.out.println("MD5: "+new BigInteger(1,m.digest()).toString(16));

return (new BigInteger(1,m.digest()).toString(16)); }

**Bonus implementation: DNS server with AES encryption**

**Client implementation:**

**Output:**



**myClient.java:Uses AESEncrp.java for encryption and decryption**

package myclient;

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.Insets;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.net.\*;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.util.\*;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

/\*

\* Client program

\*/

public class myClient extends JFrame{

public static int port;

public static String IPAddress;

public Socket clientSocket;

public static myClient frame;

static ObjectInputStream inputStream = null;

static EncryptionUtil enc;

public myClient(String Name)

{

super(Name);

}

public static void callEncryption()

{

enc=new EncryptionUtil();

try {

// Check if the pair of keys are present else generate those.

if (!enc.areKeysPresent()) {

// Method generates a pair of keys using the RSA algorithm and stores it

// in their respective files

enc.generateKey();

}

final String originalText = "Text to be encrypted ";

// Encrypt the string using the public key

inputStream = new ObjectInputStream(new FileInputStream(enc.PUBLIC\_KEY\_FILE));

final PublicKey publicKey = (PublicKey) inputStream.readObject();

final byte[] cipherText = enc.encrypt(originalText, publicKey);

// Decrypt the cipher text using the private key.

inputStream = new ObjectInputStream(new FileInputStream(enc.PRIVATE\_KEY\_FILE));

final PrivateKey privateKey = (PrivateKey) inputStream.readObject();

final String plainText = enc.decrypt(cipherText, privateKey);

// Printing the Original, Encrypted and Decrypted Text

System.out.println("Original: " + originalText);

System.out.println("Encrypted: " +cipherText.toString());

System.out.println("Decrypted: " + plainText);

} catch (Exception e) {

e.printStackTrace();

}

}

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

{

//Create an object and call the method run()

frame = new myClient("Secure DNS");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container pane = frame.getContentPane();

//Set up the content pane.

pane.setLayout(null);

JTextField IPInput = new JTextField("Enter IP address");

JTextField portInput = new JTextField("Port");

JButton buttonConnect = new JButton("Connect");

pane.add(IPInput);

pane.add(portInput);

pane.add(buttonConnect);

Insets insets = pane.getInsets();

Dimension size = IPInput.getPreferredSize();

IPInput.setBounds(25 + insets.left, 5 + insets.top,

size.width, size.height);

size = portInput.getPreferredSize();

portInput.setBounds(55 + insets.left, 40 + insets.top,

size.width, size.height);

size = buttonConnect.getPreferredSize();

buttonConnect.setBounds(150 + insets.left, 15 + insets.top,

size.width + 50, size.height + 20);

buttonConnect.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

port=Integer.parseInt(portInput.getText());

IPAddress=IPInput.getText();

try {

callEncryption();

frame.run();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}});

//Size and display the window.

Insets insets1 = frame.getInsets();

frame.setSize(300 + insets1.left + insets1.right,

125 + insets1.top + insets1.bottom);

frame.setVisible(true);

}

public void run() throws Exception

{

//Create Socket,connect

clientSocket = new Socket(IPAddress,port);

//Sends message to the server

System.out.println("Hello");

PrintStream ps = new PrintStream(clientSocket.getOutputStream());

//Scanner class to receive input from user

Scanner scan = new Scanner(System.in);

scan.useDelimiter("[\r\n]+");

String cMessage ="";

InputStreamReader ir = new InputStreamReader(clientSocket.getInputStream());

BufferedReader br = new BufferedReader(ir);

//Terminate program when you send "END"

while(!(cMessage.trim().equals("END"))){

//Receive input from user

cMessage = scan.nextLine();

/\*inputStream = new ObjectInputStream(new FileInputStream(enc.PUBLIC\_KEY\_FILE));

final PublicKey publicKey = (PublicKey) inputStream.readObject();

cMessage = new String((enc.encrypt(cMessage, publicKey)));

System.out.println("yo"+cMessage+"yo");\*/

cMessage = AESencrp.encrypt(cMessage);

if(cMessage.trim().equals("END"))

{

break;

}

//Prevent from sending empty input to server

if (cMessage.trim().isEmpty()) {

continue;

}

//Send data to server

ps.println(cMessage);

ps.flush();

//Read and display response from server

String message = br.readLine().trim();

System.out.println(clientSocket.getLocalSocketAddress()+message);

}

System.out.println("Terminating program");

//Close sockets

br.close();

ir.close();

scan.close();

ps.close();

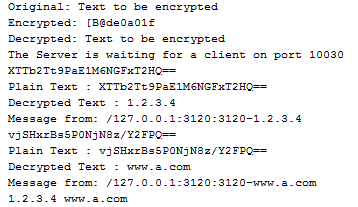
clientSocket.close();

}

}

**Server implementation:**

**Output:**



**Socket\_Server.java: Uses AESEncrp.java for encryption and decryption**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package socket\_server;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.ObjectInputStream;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @author Akash, Himanshu and Poornima

\*/

public class Socket\_server {

static ObjectInputStream inputStream = null;

static EncryptionUtil enc;

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

{

Socket\_server myServer = new Socket\_server();

callEncryption();

myServer.run();

}

public static void callEncryption()

{

enc=new EncryptionUtil();

try {

// Check if the pair of keys are present else generate those.

if (!enc.areKeysPresent()) {

// Method generates a pair of keys using the RSA algorithm and stores it

// in their respective files

enc.generateKey();

}

final String originalText = "Text to be encrypted ";

// Encrypt the string using the public key

inputStream = new ObjectInputStream(new FileInputStream(enc.PUBLIC\_KEY\_FILE));

final PublicKey publicKey = (PublicKey) inputStream.readObject();

final byte[] cipherText = enc.encrypt(originalText, publicKey);

// Decrypt the cipher text using the private key.

inputStream = new ObjectInputStream(new FileInputStream(enc.PRIVATE\_KEY\_FILE));

final PrivateKey privateKey = (PrivateKey) inputStream.readObject();

final String plainText = enc.decrypt(cipherText, privateKey);

// Printing the Original, Encrypted and Decrypted Text

System.out.println("Original: " + originalText);

System.out.println("Encrypted: " +cipherText.toString());

System.out.println("Decrypted: " + plainText);

} catch (Exception e) {

e.printStackTrace();

}

}

public void run() throws Exception

{

int port = 10030;

File file = new File("C:\\Users\\Akash\\workspace1\\Socket\_server\\src\\Yo");

BufferedReader reader = null;

try {

reader = new BufferedReader(new FileReader(file));

} catch (IOException e) {

}

//Initializes the port the serverSocket will be on

ServerSocket serverSocket = new ServerSocket(port);

System.out.println("The Server is waiting for a client on port "+port);

//Accepts the connection for the client socket

Socket socket = serverSocket.accept();

InputStreamReader ir = new InputStreamReader(socket.getInputStream());

BufferedReader br = new BufferedReader(ir);

String message;

//= br.readLine();

//Confirms that the message was received

PrintStream ps = new PrintStream(socket.getOutputStream());

while((message =br.readLine())!=null)

{

int flag=0;

int i=0;

/\* inputStream = new ObjectInputStream(new FileInputStream(enc.PRIVATE\_KEY\_FILE));

final PrivateKey privateKey = (PrivateKey) inputStream.readObject();

System.out.println(message);

byte[] mellow= message.getBytes();

System.out.println(mellow);

System.out.println(new String(mellow));

message= enc.decrypt(message.getBytes(), privateKey);\*/

System.out.println(message);

message=message.trim();

String passwordDec = AESencrp.decrypt(message);

System.out.println("Plain Text : " + message);

System.out.println("Decrypted Text : " + passwordDec);

message=passwordDec;

System.out.println("Message from: "+ socket.getRemoteSocketAddress()+":"+socket.getPort()+"-"+message);

String pattern = "\\D+\\.\\D+\\.\\D+";

String pattern1 = "\\d+\\.\\d+\\.\\d+\\.\\d+";

String pattern2 = "(\\D+\\.\\D+\\.\\D+)";

String pattern3 = "(\\d+\\.\\d+\\.\\d+\\.\\d+)";

// Create a Pattern object

Pattern r = Pattern.compile(pattern);

Pattern r1 = Pattern.compile(pattern1);

Pattern r2 = Pattern.compile(pattern2);

Pattern r3 = Pattern.compile(pattern3);

// Now create matcher object.

Matcher m = r.matcher(message);

Matcher m1 = r1.matcher(message);

Matcher m2;

Matcher m3;

if(m.matches()==true)

{

String Line;

while ((Line = reader.readLine()) != null) {

System.out.println(Line);

m2=r2.matcher(Line);

m3=r3.matcher(Line);

if(m2.find()&&m3.find())

{

System.out.println("yo"+message+m2.group(0)+m3.group(0));

if(m2.group(0).trim().equals(message)){

//System.out.println("yo"+m3.group(0));

ps.println("IP Address: "+m3.group(0));

flag=1;

break;

}

}

}

if(flag==0)

{

ps.println("Could not find IP Address: ");

flag=0;

}

reader.close();

reader = new BufferedReader(new FileReader(file));

}

else if(m1.matches()==true)

{

//ps.println("ip address");

String Line;

while ((Line = reader.readLine()) != null) {

m2=r2.matcher(Line);

m3=r3.matcher(Line);

if(m2.find()&&m3.find())

{

if(m3.group(0).trim().equals(message)){

//System.out.println("yo"+m3.group(0));

ps.println("Hostname: "+m2.group(0));

flag=1;

break;

}

}

}

if(flag==0)

{

ps.println("Could not find IP Address: ");

flag=0;

}

reader.close();

reader = new BufferedReader(new FileReader(file));

}

else

{

ps.println(message+ " is an invalid input");

}

}

System.out.println("Connection closed");

ps.close();

br.close();

ir.close();

serverSocket.close();

}

}

**PERFORMANCE EVALUATION:**

**Library implementation of RSA generation of public and private key:**

Average Time to generate the public and private key is about 1540 ms.

Time taken to encrypt them is 1500 ms

Time taken to decrypt them is 17 ms

**AES custom implementation:**

On an average, For a 10 letter message

Time for generation is 0 ms

Time for encryption is 1 ms

Time for decryption takes almost 1 ms.

**AES Library implementation:**

The time taken for the key generation of the library function is 0ms

Time for encryption is 300 ms

Time for decryption is 1ms

**RSA custom implementation:**

Average time for generation is 1 ms

Encryption takes 1ms

Decryption takes 20 ms.

**Custom implementation for RSA & AES library encryption.**

For encryption, both the public key generation and the AES algorithm takes 300ms, and for decryption it is 20 ms.