Exercise III-1

Since the encryption function raises the plain text to the public exponent i.e. 3 and then the cipher text is obtained. In RSA when prime number is used as a public exponent, it fastens the algorithm as prime number in binary does not contains more zeros. So prime numbers like 3, 17, 65537 have proved to be practical and so enable speedy algorithm. But according to this scenario no doubt the encryption algorithm would work fast, and then it can no longer be secure. If low exponent is taken and the same message is sent to all recipients encrypting it with same exponent, then there can be an attack known as Low-Exponent Attack or Common Modulus Attack. And thus the data privacy is lost and the algorithm is no longer secure as it can be easily be broken by very less computing. Now as per the situation described above, e i.e. public exponent is small. Now to determine the public exponent it should always satisfy the equation. I.e. where e is the public exponent and N is the modulus. If that condition is satisfied then it is very hard to compute the eth root and also it is greater than modulus so Modular arithmetic is also performed. That forms a secure cipher and has full privacy. Now he situation that is described the public exponent is small, i.e. 3 and is smaller than modular N. so by just calculating the cube root of cipher text will lead us to the plain text. (In general eth root is calculated, but as e is 3 over here so cube root is calculated).

Question 2

Program to break Textbook RSA. To break the Textbook RSA eth root of the cipher text is taken, As value of e is 3 so have calculated the cube root of cipher text.

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.math.BigDecimal;

import java.math.BigInteger;

public class RSA {

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

BigDecimal c=new BigDecimal("674472526620593903800497637242400187916753185909");

double d=Math.cbrt(c.doubleValue());

System.out.printf("%f%n",d);

BigDecimal ans=new BigDecimal(d);

BigInteger t=ans.toBigInteger();

String str =ans.toString();

byte[] b=new byte[str.length()];

for(int i=0;i<str.length()-1;i=i+2){

char a=str.charAt(i);

char z=str.charAt(i+1);

String dec=a+""+z;

int ex=Integer.parseInt(dec);

System.out.print((char)ex);

}

}

}

Question 3

Textbook RSA (also known as RSA without padding) can be enhanced to render this and other attacks impossible. It can be done in the following way:-

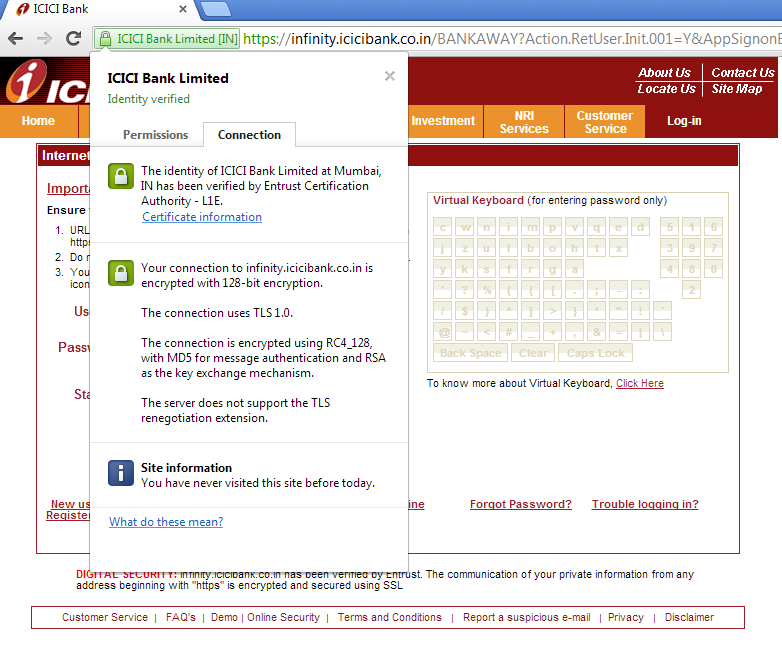
> The RSA can be made more secure, i.e. by using the padding (also known as adding padding to the message). By Introducing the padding of zeros, and adding it to the message, it creates randomness in the message. Hence the problem of RSA being deterministic is resolved. By adding these features we can make this a CPA secure encryption scheme and make the attacks impossible. Also other thing is RSA-OEAP. It also has an option to add hash as well as give own structure to cipher text.

The three major security benefits of this method are:-

* Adds a specific structure to the cipher text.
* Resistance against mangling of cipher text.
* Creates randomness in encryption.

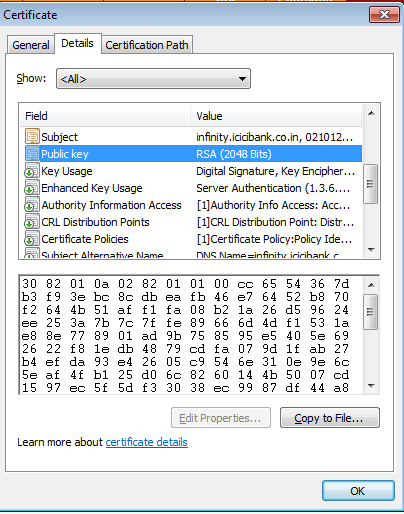
Exercise III-2

Question 1



RSA Public key

30 82 01 0a 02 82 01 01 00 cc 65 54 36 7d b3 f9 3e bc 8c db ea fb 46 e7 64 52 b8 70 f2 64 4b 51 af f1 fa 08 b2 1a 26 d5 96 24 ee 25 3a 7b 7c 7f fe 89 66 6d 4d f1 53 1a e8 8e 77 89 01 ad 9b 75 85 95 e5 40 5e 69 26 22 f8 1e db 48 79 cd fa 07 9d 1f ab 27 b4 ef da 93 e4 26 05 c9 54 6e 31 0e 9e 6c 5e af 4f b1 25 d0 6c 82 60 14 4b 50 07 cd 15 97 ec 5f 5d f3 30 38 ec 99 87 df 44 a8 82 8a d7 07 90 45 64 60 21 05 76 12 f2 15 d0 9a ea 97 91 18 f1 55 65 7c df 99 db c6 77 89 eb 63 c6 ac 39 c3 2e 5b 81 3a da 0c b3 b5 48 d6 a8 ee 72 08 66 96 e7 61 18 e9 15 39 c6 ed 11 0f fa b4 d8 5b b2 38 dc 35 8e 38 c6 36 45 c0 43 aa 85 90 66 b3 b9 81 d2 5f 0e 2a 72 7d 9f de 18 5e 41 7a 64 17 56 b1 1a d1 64 10 dd 69 d7 36 f5 a9 dd a1 48 63 a6 95 04 af ac 3a 27 95 dc 50 25 d5 54 44 35 c9 62 86 55 70 5c 80 50 3a ab 02 03 01 00 01



Question 2

Digital Signatures are implemented using RSA. Also known as RSA signature. To communicate over Secured Socket Layer or TLS, client and server needs to perform hand shake first. The steps involved in hand shake are as follow:-

* First the client and server agree on same version of protocol.
* Then they select their cryptographic functions.
* Authenticate each other by exchanging digital certificates.
* Use asymmetric encryption techniques to generate a shared secret key, which avoids the key distribution problem. SSL or TLS then uses the shared key for the symmetric encryption of messages, which is faster than asymmetric encryption.

For servers authentication client uses its public key to encrypt data and that is used to computer secret key. Server can generate secret key only if it decrypts the data. Whereas for client authentication, the server uses the clients public key in the client certificate to decrypt the data that client sends during handshake. Then the exchange of finished messages using secret key confirms that authentication is complete.

Digital Signatures in SSL/TLS

Digital signatures are formed by encrypting the message representation. The encryption uses the private key of the person who is signing the data. And for efficiency, the Digital signatures are performed on message digest. When the receiver received the signed Document, then it decrypts the signature using his public key. After decrypting the Signature the receiver then computes message digest from the message and then verifies the two digests are same or not.

GnuPG Exercise

Determine two ways on how to verify that your GnuPG version is indeed integer

* By Checking the SHA1SUM (Verifying the SHA1 checksum)
* Checking the Signature of the file gnupe-{{{gnupg\_ver}}}.tar.bz2

How can you be convinced that this public key is actually integer? What steps can you take to verify

that you got the right key?

The steps that I would take to verify that this public key is integer is by using the command ‘pgpdump’ command. The Dump Integer option will display the values used for the key. And to verify that I have the right key I would check the identity of key while importing it and also the email address. Basically I would follow the steps as follows:-

Retrive:-

When retrieving new encryption key, the first step is to search for email address of the user in question.

Fingerprint:-

Key fingerprint verification is actually much easier than it sounds. In a nutshell both sides should compare the fingerprints of the public key in question. If fingerprints match they can proceed further if not then the they cannot be trusted.

Encrypt:-

As a third and final safety measure, a test message should sen to the user with the key in question requesting a response that includes decrypted message.

Trust:-

Once the identity is been confirmed and verified then that key should be marked as trusted.