RSA Algorithm and its Applications

A Project Submitted
in Partial Fulfillment of the Requirements for
the Degree of
Bachelor of Technology
in
Computer Science Engineering/Computer Science

by
Pranshu Sood
Shivam Kumar
Shivang Ahuja
Parikshit Palsania
Manas Yadav

SCHOOL OF ENGINEERING AND TECHNOLOGY
BML MUNJAL UNIVERSITY GURGAON
November, 2019

Contents

Abstract

Information Security has become a vital issue in digital communication. Cryptography has emerged as an answer and plays a vital role in data security systems. This project discusses cryptography analysis and describes the RSA cryptosystem and its applications. Application include Secure File Transmission (File Encryption and Decryption) , Digital Image Encryption , Big Data Protection and Digital Signatures verification .Two keys are generated in RSA, one key is used for encryption & other key which is only known to authenticated receiver can decrypt the message. No other key can decrypt the message. Every communicating party needs just a key pair for communicating with any number of other communicating parties[7].

Introduction

Cryptography is the art of writing or solving codes. This definition may be correct historic, but it does not have the feel of modern cryptography.Firstly,it focuses only on the problem of secret communication. Secondly, the definition refers to cryptography as a form of art. Indeed, until the 20th century (and arguably until late in that century), cryptography was an art. In the late 20th century, this picture of cryptography radically changed.Furthermore, the field of cryptography now encompasses much more than secret communication, including message authentication, digital signatures, protocols for exchanging secret keys, authentication protocols, electronic auctions and elections, and digital cash [8].

The creation of public key cryptography by Diffie and Hellman in 1976 and the subsequent invention of the RSA public key cryptosystem by Rivest, Shamir, and Adleman in 1978 are watershed events in the long history of secret communications. [9].

In 1978, a paper was published by R. Rivest, A. Shamir, and L. Adleman. In this paper they describe a public-key cryptosystem, including key generation and a public-key cipher, whose security rests upon the presumed difficulty of factoring integers into their prime factors. This cryptosystem, which has come to be known by the acronym from the authors' names, the *RSA cryptosystem* has stood the test of time to this day, where it is used in cryptographic applications from banking, and email security to e-commerce on the Internet [10].

The RSA Algorithm has withstood every attack from the best cryptographic minds. The power of the algorithm, the absence of rigorous proof notwithstanding, provides security. According to Dan Boneh, a computer science professor at Stanford University, "we kind to chip at the sides, but no one have figured out how to get at the heart of it." (Robinson, 2003, pp.6)

There are many applications for RSA, but in practice it is most often used for :-

1. Digital Image Encryption and Decryption
2. Digital Signatures
3. Big Data Protection, Especially those which are highly confidential
4. File Encryption and Decryption

The reason for which we are using this algorithm is the problem we are facing

in securing the above mentioned application , thus we need a concrete algorithm to deal with almost all against any of the above applications.

Details of the Work

In this project we have worked on four most important application of RSA Algorithm :-

1. **Digital Image Encryption and Decryption**

WORKING

The RSA algorithm in image encryption and decryption consists of three major steps :

1. Key generation
2. Encryption
3. Decryption

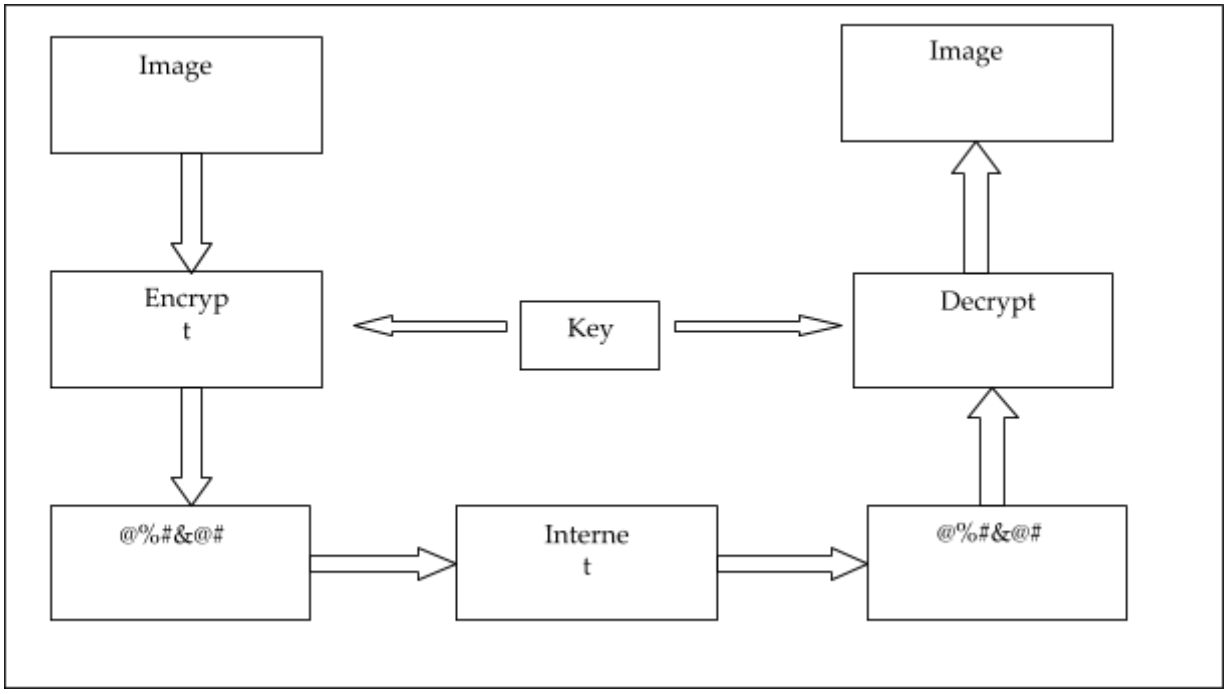
The keys for the image encryption and decryption is generated by the following steps:

1. Choose two distinct prime numbers p and q.
2. For security purposes, these prime numbers p and q should be chosen at random and must be of similar bit strength.
3. Compute n = pq. Here n is used as the modulus for both the public and private keys. Its length is expressed in bits which is key length.
4. Compute $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1) = n - (p + q - 1)$, where ϕ is Euler's totient function.
5. Choose an integer e such that $1 < e < \phi n)$ and gcd(e, $\phi(n)$) = 1; i.e., e and $\phi(n)$ are co-prime and here e is the public key exponent. e is having a short bit-length and small Hamming weight results such as: 216 + 1 = 65,537. However, if the value of e is small. e.g:- e= 3 have been less secure.
6. Determine d as $d \equiv e^{-1} \pmod{\phi(n)}$;i.e., d is the multiplicative inverse of e(modulo $\phi(n)$).

Encryption:
Split a message M into a sequence of blocks M1, M2, . . . , Mt where each Mi satisfies $0 \leq Mi < n$. Then encrypt these blocks as[13] $c \equiv m^e \pmod{n}$

Decryption:
Given the private key d and the cipher text C, the decryption function is:
$m \equiv c^d \pmod{n}$ [12]

Note that encryption does not increase the size of a message. Both the message and the cipher text are integers in the range 0 to n - 1. The encryption key is thus the pair of positive integers (e; n). Similarly, the decryption key is the pair of positive integers (d; n). Each user makes his encryption key public, and keeps the corresponding decryption key private[13].

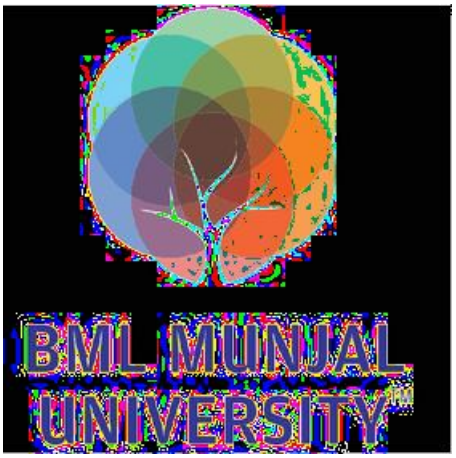


Sample result of the image encryption using the RSA Algorithm :-

P1	11
P2	23
E	49
D	9

After applying the decryption algorithm , we get almost the image back and the results can be seen below :-

Encrypted Image



Decrypted Image

2. File Encryption and Decryption

Here we have used Java programming language for writing the code for encrypting and decrypting the file , so the working is as follows :-

- The first step is to import different java classes for implementing the RSA file encryption.

- Then generate the RSA public and private keys. The keys are generated using the class KeyPairGenerator. The length of the RSA key generated is 2048 bits as it would be able to provide good security.

- Once the keys are generated, we need to save them for future use. The public key can be distributed, but the private key needs to be secured properly. The public keys is saved into a file with the extension *.pub* and the private key is stored in a file with the extension *.key*. Both keys are stored in binary format.

- Then the encryption process starts which is done with the help of the private key,The main class handling the encryption is the *Cipher* class. We create a suitable cipher, initialize it with the private key (or the public key as required, see above), and perform the encryption.

- Then the decryption process takes place ,Once we obtain the file encrypted with the private key, we can decrypt it using the public key. The process is similar and shown below. We write the decrypted data to an output file for verification .

3. Big data protection

Big data actually involves all the data which contains almost end number of values, some good examples for these are hospital data which includes all the details of patients , tourism management platform etc . The way these types of data are protected is very much the same as the way in which the image is encrypted , the mathematics involved in all these application is almost same .

1) Key Generation process :-

(1) The two large enough and different prime numbers u and v, which should be kept secretly.
(2) Calculate the value of x and y, x=u×v, x is public. y=(u-1)×(v-1), y is secret.

(3) Select a public random integer s(0<s<y), and Gcd (s,y) =1, s is generally a smaller integer. And {s,x} is the public key.
(4) Calculate the value of g,g=s-1mod y, t is secret, {g,x} is private key.

2) Encryption and decryption

This whole process includes the conversion of plain text to cipher text with the help of public skills and further with the help of private key again decryption can be done.

In the RSA algorithm operation, the Fast Exponential Algorithm was used for the encryption and decryption.Because the number is relatively large and the algorithm is used for simulation, it is stored with long integer.
The description of the fast exponential

$$\sum_{i=0}^{k} z_i 2^i = \sum_{z_i=1} 2^i$$

algorithm is that the binary of p is zkzk-1...z0, and zi={0,1}(i=0,1,...,k), m= ,then

$$e^p \bmod n = e^{\sum_{z_i=0} 2^i} \bmod n = \left[\prod_{z_i=1} e^{(2^i)}\right] \bmod n = \left(\prod_{z_i=1} [e^{(2^i) \bmod n}]\right) \bmod n$$

4. Digital Signatures

How Signing Works :-

For any RSA public/private key triple, (*e*, *d*, *n)*, the key mathematical fact is that the encryption and decryption functions are inverses of one another. That is, if

f (m) = m^e mod *n* is the encryption function (which is public) and

G(m) = m^d mod *n* is the decryption function (which is private),

then,

f (g(m)) = *m* and *g(f (m))* = *m*

The idea behind a digital signature using RSA is that *f* is a function that is known to everyone, but only you know your decryption function.In order for A to sign a message, *m*, sends $g_A(m)$ together with an indication that the message is from A. When B gets it, he sees that the

message is from A and applies her public encryption function, f_A, to $g_A(m)$ and will get m. If C tries to send a spoofed message, m ', purportedly from A to B, he has no way of being successful since he doesn't know g_A.

Simultaneous Encryption and Signing :-

If A wants to sign and encrypt a message, she can follow the diagram on the bottom left, starting at the bottom of the diagram with a plaintext message and traveling clockwise. If the message is m, then the results coming out of the first two boxes are $f_B(m)$ and $g_A(f_B(m))$, respectively. The latter of the two numbers is what is sent to B. When applies A's public key to what is received, the result is

Conclusion

In the digital world, the security of images,files and big datasets has become more important as the communication has increased rapidly. Here, the image encryption algorithm ,file encryption algorithm proposed efficient and highly securable with high level of security and less computation[12]. With the implementation of RSA algorithm, we reach a conclusion that for better security of any text(which includes both file and big data) or image. Then we got on encrypted image/file/big dataset which is very difficult to decrypt by any other person. So, the conclusion is that the encrypted file/dataset/image is more secure[11]. The main advantage to encryption is that it separates the security if data from the security of the device where the data is transmit over the Internet.The people should keep in mind the standard email is not secure and is in fact tantamount to writing sensitive information on postcards[12].

References

[1] Jonathan Katz, Yehuda Lindell, “Introduction to Modern Cryptography: Principles and Protocols”, ISBN: 978-1-58488-551-1, 2008.

[2] Ali E. Taki El_Deen, El-Sayed A. El-Badawy, Sameh N. Gobran, “Digital Image Encryption using RSA Algorithm”, *p- ISSN: 2278-8735.*

[3] R.L. Rivest, A. Shamir, and L. Adleman , “ A Method for Obtaining Digital Signatures and Public-Key Cryptosystem”.

[4] Rajan.S.Jamgekar, Geeta Shantanu Joshi , “ File Encryption and Decryption Using Secure RSA ” , ISSN: 2319–6378.

[5] Handbook of Applied Cryptography (Discrete Mathematics and Its Applications) by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone.

[6] Huijuan Xie , “ The Application of RSA Encryption Algorithm in the Hainan Rural Tourism Management Platform ” , Advances in Social Science, Education and Humanities Research, volume 123.

[7] Borko Furht, Darko Kirovski, “Multimedia Encryption and Authentication Techniques and Applications”, ISBN: 0-8493-7212-7, 2006.

[8] Jonathan Katz, Yehuda Lindell, “Introduction to Modern Cryptography: Principles and Protocols”, ISBN: 978-1-58488-551-1, 2008.

[9] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, “An Introduction to Mathematical Cryptography”, ISBN: 978-0-387-77993-5, 2008.

[10] Richard A. Mollin, “Codes: The Guide to Secrecy from Ancient to Modern Times”, ISBN-10: 1-58488-470-3, 2005.

[11] Sunita,”Image Encryption/Decryption Using RSA Algorithm”, ISSN: 2321-8363.

[12] S.Anandakumar ,”Image Cryptography Using RSA Algorithm in Network Security,” ISSN: 2231-0711

[13] Ali E. Taki El_Deen, El-Sayed A. El-Badawy, Sameh N. Gobran ,”Digital Image Encryption Based on RSA Algorithm”,p- ISSN: 2278-873,e-ISSN: 2278-2834.

Appendices

Code for File Encryption and Decryption using JAVA :-

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.FileWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Base64;
import java.util.Arrays;
import java.security.Key;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;

public class File
{
	static private Base64.Encoder encoder = Base64.getEncoder();
	static SecureRandom srandom = new SecureRandom();

	static private void processFile(Cipher ci,InputStream in,OutputStream
out) throws javax.crypto.IllegalBlockSizeException,
javax.crypto.BadPaddingException, java.io.IOException
	{
		byte[] ibuf = new byte[1024];
		int len;
		while ((len = in.read(ibuf)) != -1)
		{
```

```
                byte[] obuf = ci.update(ibuf, 0, len);
                if ( obuf != null ) out.write(obuf);
            }
            byte[] obuf = ci.doFinal();
            if ( obuf != null ) out.write(obuf);
        }

        static private void processFile(Cipher ci,String inFile,String outFile)
throws javax.crypto.IllegalBlockSizeException,
javax.crypto.BadPaddingException, java.io.IOException
        {
            try (FileInputStream in = new FileInputStream(inFile);
            FileOutputStream out = new FileOutputStream(outFile))
            {
                processFile(ci, in, out);
            }
        }

        static private void doGenkey(String[] args) throws
java.security.NoSuchAlgorithmException, java.io.IOException
        {
            if ( args.length == 0 )
            {
                System.err.println("genkey -- need fileBase");
                return;
            }

            int index = 0;
            String fileBase = args[index++];
            KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
            kpg.initialize(2048);
            KeyPair kp = kpg.generateKeyPair();
            try (FileOutputStream out = new FileOutputStream(fileBase +
".key"))
            {
                out.write(kp.getPrivate().getEncoded());
            }

            try (FileOutputStream out = new FileOutputStream(fileBase +
".pub"))
            {
            out.write(kp.getPublic().getEncoded());
```

```
        }
}


static private void doEncrypt(String[] args)
        throws java.security.NoSuchAlgorithmException,
        java.security.spec.InvalidKeySpecException,
        javax.crypto.NoSuchPaddingException,
        javax.crypto.BadPaddingException,
        java.security.InvalidKeyException,
        javax.crypto.IllegalBlockSizeException,
        java.io.IOException
{
        if ( args.length != 2 )
        {
                System.err.println("enc pvtKeyFile inputFile");
                System.exit(1);
        }

        int index = 0;
        String pvtKeyFile = args[index++];
        String inputFile = args[index++];
        byte[] bytes = Files.readAllBytes(Paths.get(pvtKeyFile));
        PKCS8EncodedKeySpec ks = new PKCS8EncodedKeySpec(bytes);
        KeyFactory kf = KeyFactory.getInstance("RSA");
        PrivateKey pvt = kf.generatePrivate(ks);

        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, pvt);
        processFile(cipher, inputFile, inputFile + ".enc");
}

static private void doDecrypt(String[] args)
        throws java.security.NoSuchAlgorithmException,
        java.security.spec.InvalidKeySpecException,
        javax.crypto.NoSuchPaddingException,
        javax.crypto.BadPaddingException,
        java.security.InvalidKeyException,
        javax.crypto.IllegalBlockSizeException,
        java.io.IOException
{
        if ( args.length != 2 )
```

```
            {
                System      .err.println("dec pubKeyFile inputFile");
                System.exit(1);
            }

            int index = 0;
            String pubKeyFile = args[index++];
            String inputFile = args[index++];
            byte[] bytes = Files.readAllBytes(Paths.get(pubKeyFile));
            X509EncodedKeySpec ks = new X509EncodedKeySpec(bytes);
            KeyFactory kf = KeyFactory.getInstance("RSA");
            PublicKey pub = kf.generatePublic(ks);

            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            cipher.init(Cipher.DECRYPT_MODE, pub);
            processFile(cipher, inputFile, inputFile + ".ver");
        }

        static public void main(String[] args) throws Exception
        {
            if ( args.length == 0 )
            {
            System.err.print("usage: java sample1 command params..\n" +
                    "where commands are:\n" +
                    "  genkey fileBase\n" +
                    "  tnyenc pvtKeyFile inputFile\n" +
                    "  tnydec pubKeyFile inputFile\n");

            System.exit(1);
            }

            int index = 0;
            String command = args[index++];
            String[] params = Arrays.copyOfRange(args, index, args.length);
            if ( command.equals("genkey") ) doGenkey(params);
            else if ( command.equals("enc") ) doEncrypt(params);
            else if ( command.equals("dec") ) doDecrypt(params);
            else throw new Exception("Unknown command: " + command);
        }
}
```

Code for Image Encryption and Decryption using Python and Matlab :-

```
!pip install Pillow

p1 = 11
p2 = 23

n = p1 * p2

phin = (p1-1)*(p2-1)
e = 0

import math
from google.colab import files
from io import BytesIO
from PIL import Image
import numpy as np
from IPython.display import display # to display images

def modi(a, m):
  a = a%m
  for x in range(1, m):
    if((a * x) % m == 1):
      return x
  return 1

for i in range(1, 50):
  if math.gcd(phin , i) == 1:
    e = i
d = modi(e, phin)

uploaded = files.upload()

im = Image.open(BytesIO(uploaded['bmu.jpg']))
im2 = Image.open(BytesIO(uploaded['bmu.jpg']))
w, h = im.size

display(im)

for x in range(w):
  for y in range(h):
    r,g,b = im.getpixel((x,y))
```

```
    r = ((r ** e) % n) #% 256
    g = ((g ** e) % n) #% 256
    b = ((b ** e) % n) #% 256
    l = r,g,b
    im.putpixel((x,y), l)

display(im)

for x in range(w-1):
  for y in range(h-1):
    r,g,b = im.getpixel((x,y))
    r = ((r ** d) ) % n
    g = ((g ** d) ) % n
    b = ((b ** d) ) % n
    l = r,g,b
    im.putpixel((x,y), l)

display(im)
```