VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

RSA BASED SECURED CHAT ROOM

PROJECT REPORT

CSE3501

Information Security Analysis and Audit

by

19BCE1533 – Mourya Vardhan Reddy

Under the guidance of

Dr. Anusha.k

School of Computer Science and Engineering

Fall
2020-21

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. - Abstract:

In this era modern communication filled with real-time and online chatting systems. A secure system is needed for clients who regularly need to share sensitive information and data to and from clients belonging to same or different organisation. In general, in most of existing system the messages are being transmitted without being encrypted across various servers. To prevent non-malicious threats to an organization this system can be adapted with ease and cost efficiency. To secure privacy and integrity of each sensitive message from clients, we are proposing to establish a client server system, incorporating cryptography at the client side to provide maximum privacy andsecurity. Even if hackers were to tap into the server side or any data links of the network they will have a very hard time decrypting the cipher text. In this time interval he/she can be located.

2. - Introduction:

Most Instant Messaging Services are plagued by the same problem. All the communications between the parties are transmitted in clear text. Anyone with a traffic sniffer on the network or at any hop in between the two end stations can eavesdrop on the conversation. Another problem is that all the clear text traffic goes through the messaging servers of these companies, therefore at any time they could choose to log all conversations between individuals. This can affect both corporate and personal users alike. Securing network communications is a practical problem facing all interconnected applications. It's not limited to Instant Messaging, but also affects E-mail, FTP, Telnet, and Virtual Desktop applications.

1.3 - Literature Survey:

S.no	Name of the Research Paper	Concept Used	Issues Identifies/research gaps
1	Security Aspect in Instant Mobile MessagingApplications	Securing the messages sentusing a chatroom application.	1] Many Applications don't have an option of encrypting messages
2	Client controlled securityfor web applications	Encrypting system for webapplication, where encryption is done on the client side.	1. Plain text is stored in database rather than cipher text. 2. Encryption is transparent and client don't know much about the security mechanism.
3	A Study on Web Application Security andDetecting Security Vulnerabilities	Different types of web security and its weakness and web security standards.	1] Vulnerabilities: Man in themiddle attack 2.) Major issue is transmission of data un-encrypted which is prone to certain vulnerabilities.
4	Modified RSA EncryptionAlgorithm (MREA)	In MREA unlike RSA, it isone way, the public key is used only for encryption, and the private key is used only for decryption.	1. Changing a part of encrypted message changes the complete meaning of plaintext. 2. Increasing the n- bit length increases the security but decreases the speed of encryption, decryption and key generation.

5	A-RSA: Augmented RSA	*A-RSA* cryptosystem is semantically secure and mitigates the indirect attacks. Choosing a message *M* and two different random components *r*1 and *r*2, leads to different ciphers for the same message	1. Suitable only for large plain text. In-efficient for small plain text 2. A-RSA algorithm is slower for larger files

1.3 - Literature Survey:

1.4 - Motivation:

Now-a-days attackers are very active and stealing the sensitive information and damaging the company with that information. Although chatting is secured, those texts are being attacked by the attackers using other methods. Members of same or different organization think twice before sharing sensitive or personal information over insecure channels. Hence to solve the problem we have come up 2 step verification system to provide high security to all clients.

1.5 - Objectives:

To provide secure channel of communication for clientsbelonging to organisation/s. Ensure integrity and privacy of messages. Provide private chatting rooms to share data and sensitive information for employees/clients. Building web application to give clients flexibility for access. Ensure each message in special secured text so as even if any hackers tap intodata-links deciphering the text is not easy.

CHAPTER -2

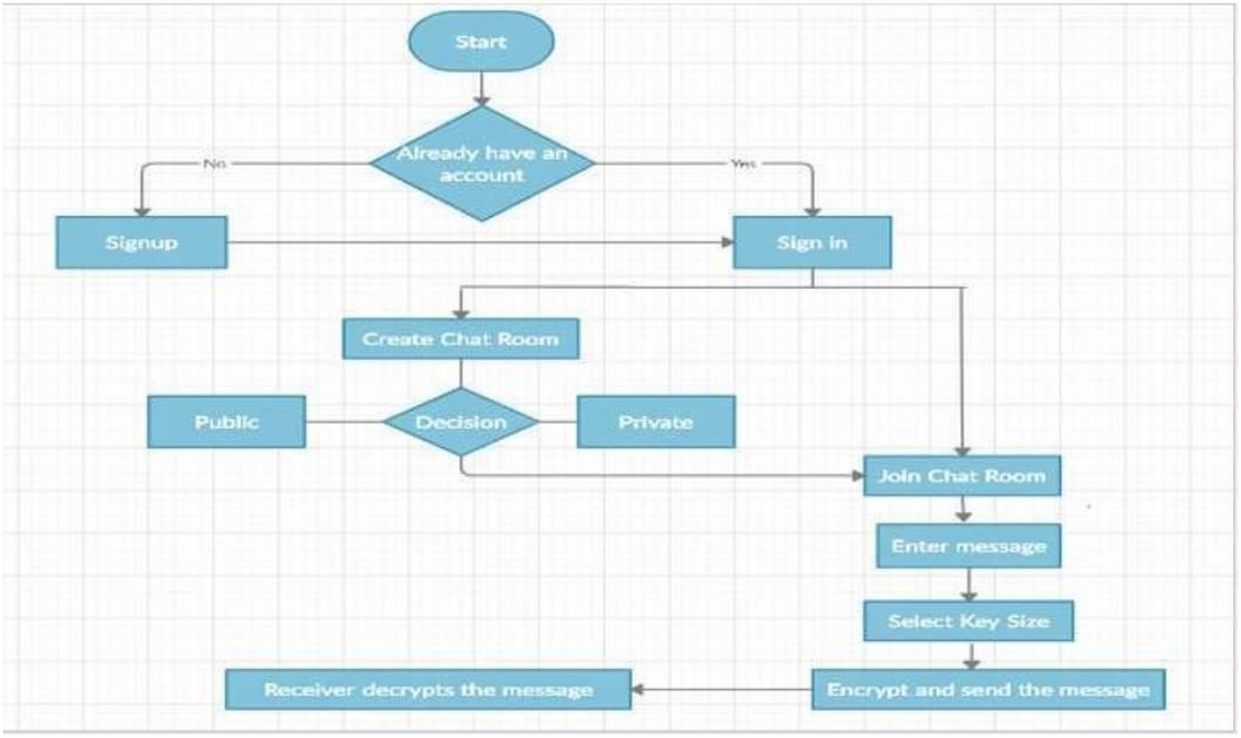
DESIGN OF PROPOSED SYSTEM

1. - Introduction:

This chapter graphically conveys the working and processing of our Proposed system which ensures high security. Here System architecture Shows the connection and linking of the different modules used. ModuleWise Architecture explains all the working and processing done in each module.so by the end we can grasp the overview of the whole system Inside out.

2. - System Architecture and Description:

Flow-chart

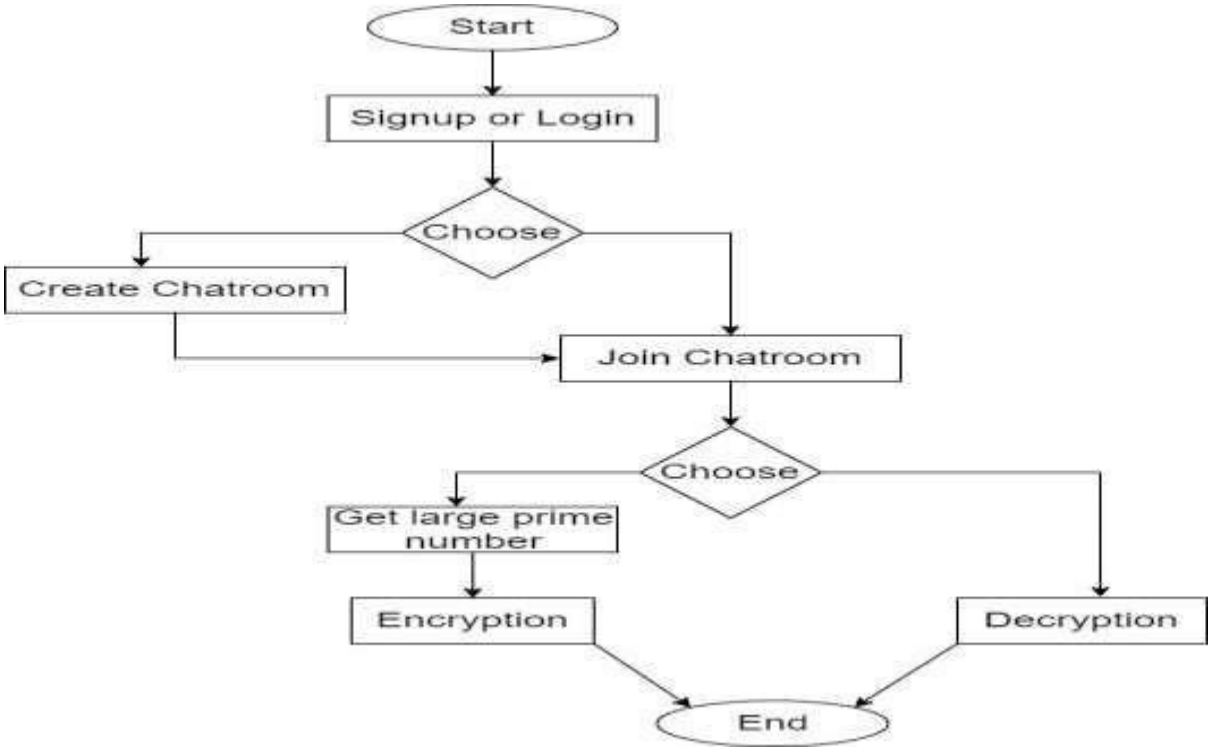


Description

This architecture represents or depicts the whole system methodology. Starting off with the application if user is new to application they are asked to register to get privilege to access the application. If the user has registered successfully he can access the application by logging into application.

Once the user logs into the application he will be directed to home page. Here user is given with two options join roomand create room. If user is accessing the application for the first time a new room has to be created. Mode of room may be public or private. If user selects create public chat room with immediate effect a room is created. But if user selects create private chat room user has to set password for room. Now if user wants to join room they can enter password to enter corresponding chat room or if it is public user can directly enter corresponding chat room. User can now entire message, click encrypt and send to other users in chat room. Other users who wants to read the text can decrypt and read the message.

3. - Functional Architecture:



Description

The architecture above represents functional architecture. Functional

architecture mainly describes how data flows through modules. This application mainly has four modules namely signup/login module, create room module, join room module, RSA module.

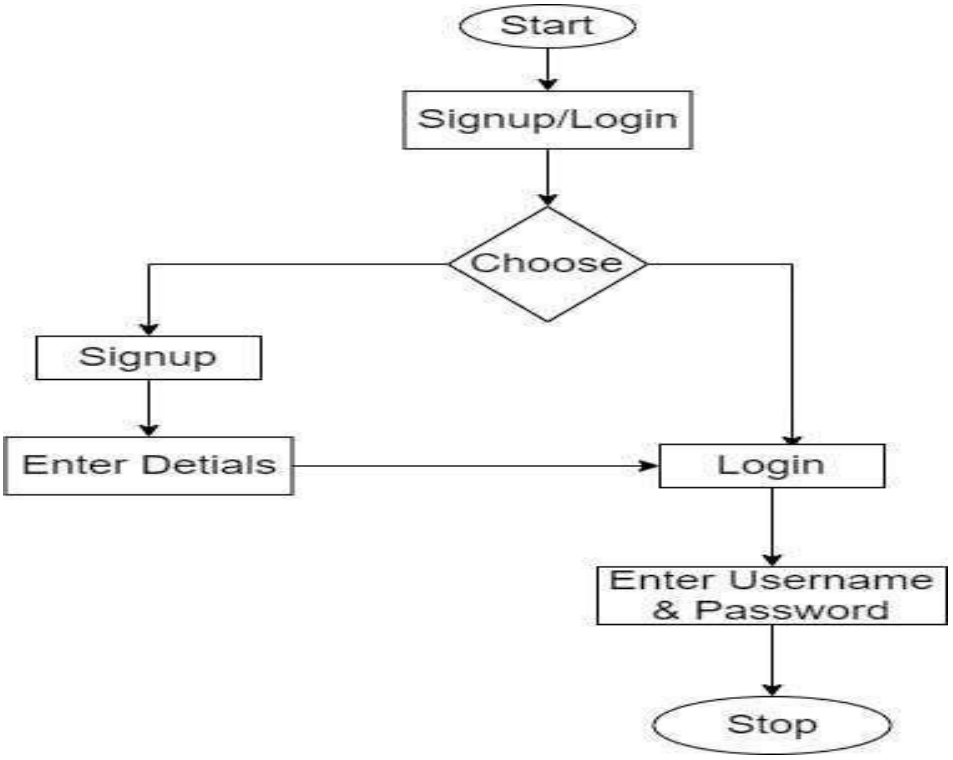
This flow chart depicts typemodules implemented in the application and how these modules are inter-connected with each other.

4. Modules:

1] Signup/login module
2] Create room module
3] Join room module
4] RSA module

5. Module-wise Architecture:

5.1. Signup/login module

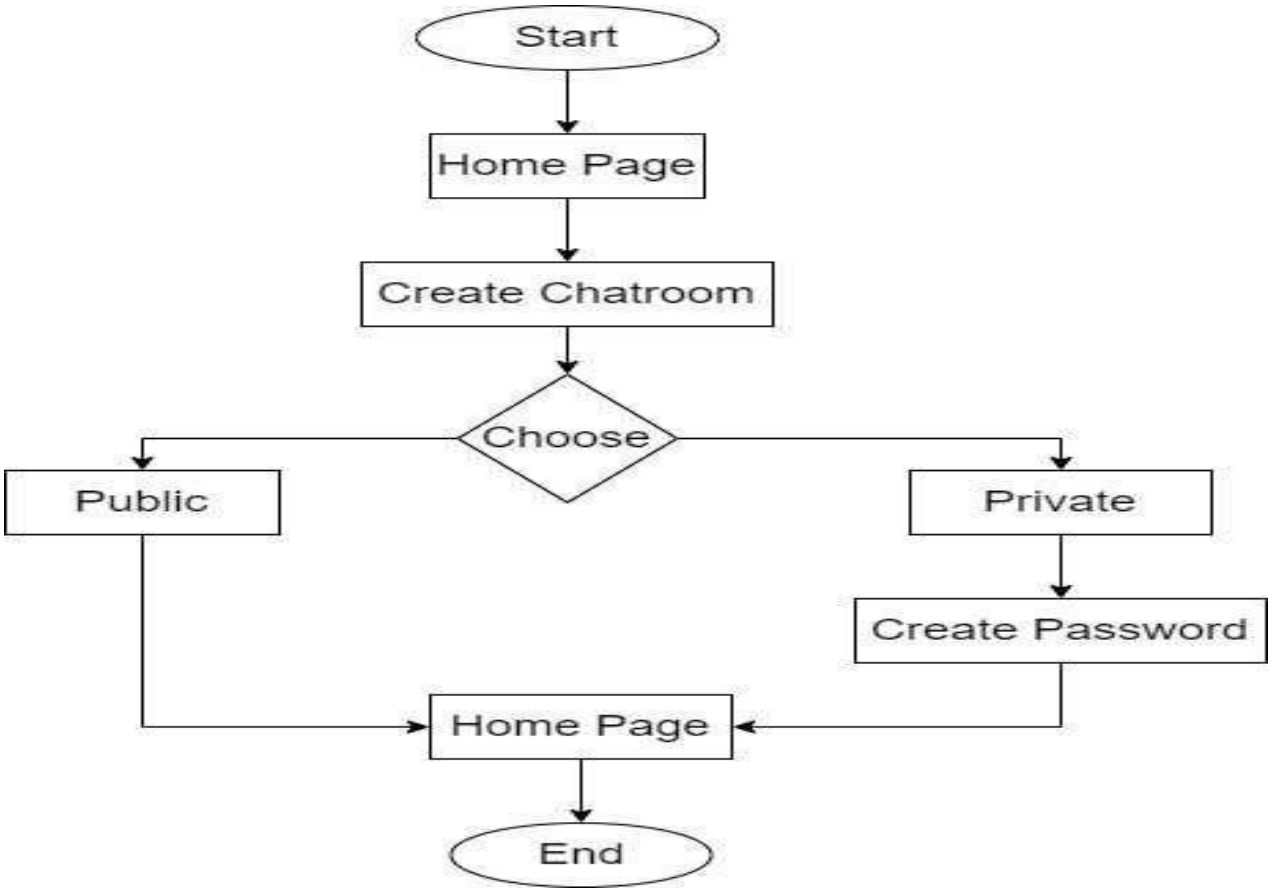


Description

Signup/Sign in module is one of the key modules of this application.This module is mainly concentrates on user access to the application. Main functionality of this module is registering user details which is useful for providing access to application.

New users that is user who are accessing application for the first time needs to register themselves that is signup to the application. Inthis process username and password are taken. Now this username and password are used to sign in into the application. Only valid usersare provided with access to the application and invalid users are not permitted to access the application

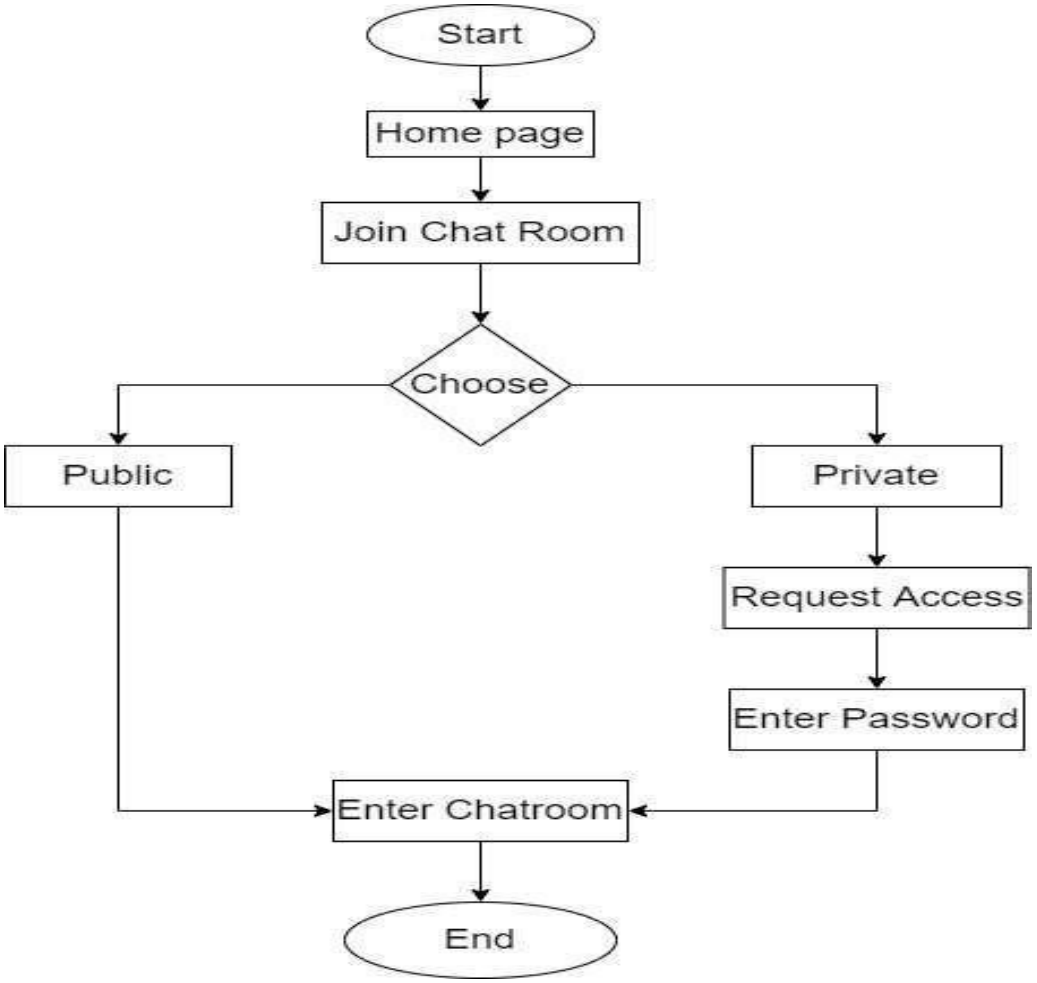
5.2. Create Room Module



Description

As this application name suggest that it is a chat room application,firstly we have to create a room for chatting purpose. Once user triesto create a room for the first time he/she will be given two optionwhether user wants to create a public chat room or private chat room.If user select public chat room with immediate effect a chat room is created. But as name suggests that it is a public chat room this roomis can be accessed by any member who has access to the application.But if user selects private room he/she needs to set password forprivate room. Users if want to access particular private chat room they need to enter password to get access to room.

5.3. Join Room Module

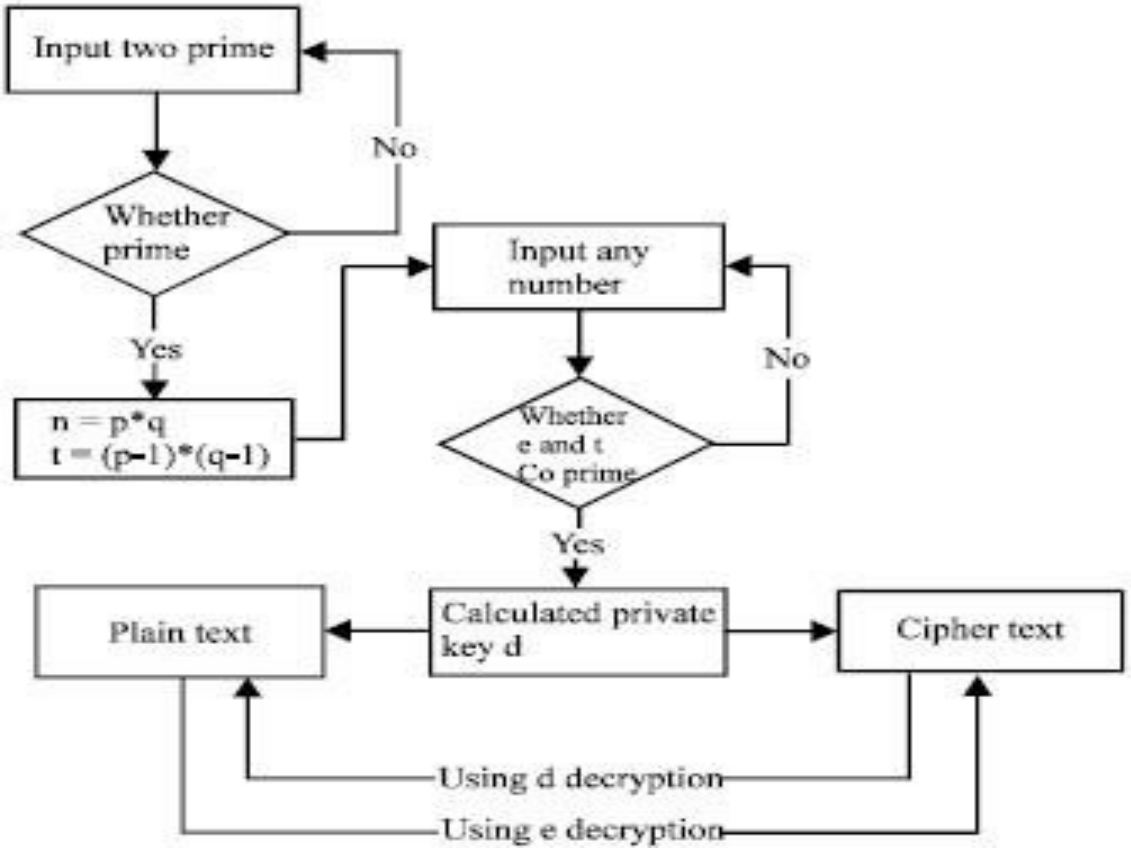


Description

This module concentrates on access of users to chat room. As we have seen in the create room module there may be public chat rooms or private chat rooms. If user wants to access any public chat room he can directly access that chat room by selecting corresponding chat room. But if user wants to access private chat room they need to typepassword of corresponding chat room to get access.

In public chat room all the messages can be viewed or read by any user who have access to the application. But in case of private chat room messages can viewed or read by users only who access to private chat room.

5.4. RSA Module



Description

This module is key part of this application. This module mainly concentrates on encryption and decryption process. In this applicationRSA crypto system is implemented which is asymmetric encryption standard.

Reason for choosing RSA crypto system is RSA is stronger than any other symmetric key algorithm. RSA has overcome the weakness of symmetric algorithm i.e. authenticity and confidentiality. On top of thesereasons RSA crypto system is easier to implement when compared to other asymmetric encryption standards.

CHAPTER-3

INTRODUCTION

1. Introduction

This section basically focuses on the solution explanation and development. And to maximize its application in the society. It shows our achieved output and all its explanation step by step.Screen shot after screen shot

2. Solution Methodology

To give more secure platform over simple networks between organisations and between members of same/different organisation we have come up with this two-step verification and authorisation system, to provide clients with trustworthychatting experience with strangers and colleagues.

We not only have a login authorisation system but also for every client who wants to enter into a private chat room, he/she must enter the correct chat room password to enter and access the content being shared in it. Hence ensuring security and privacy.

3. Algorithm:

In this project, for the implementation we use RSA algorithm. RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. This algorithm is asymmetric cryptography algorithm nobody else except browser can decrypt the data even if a third party has public key of browser.

It works on two keys:

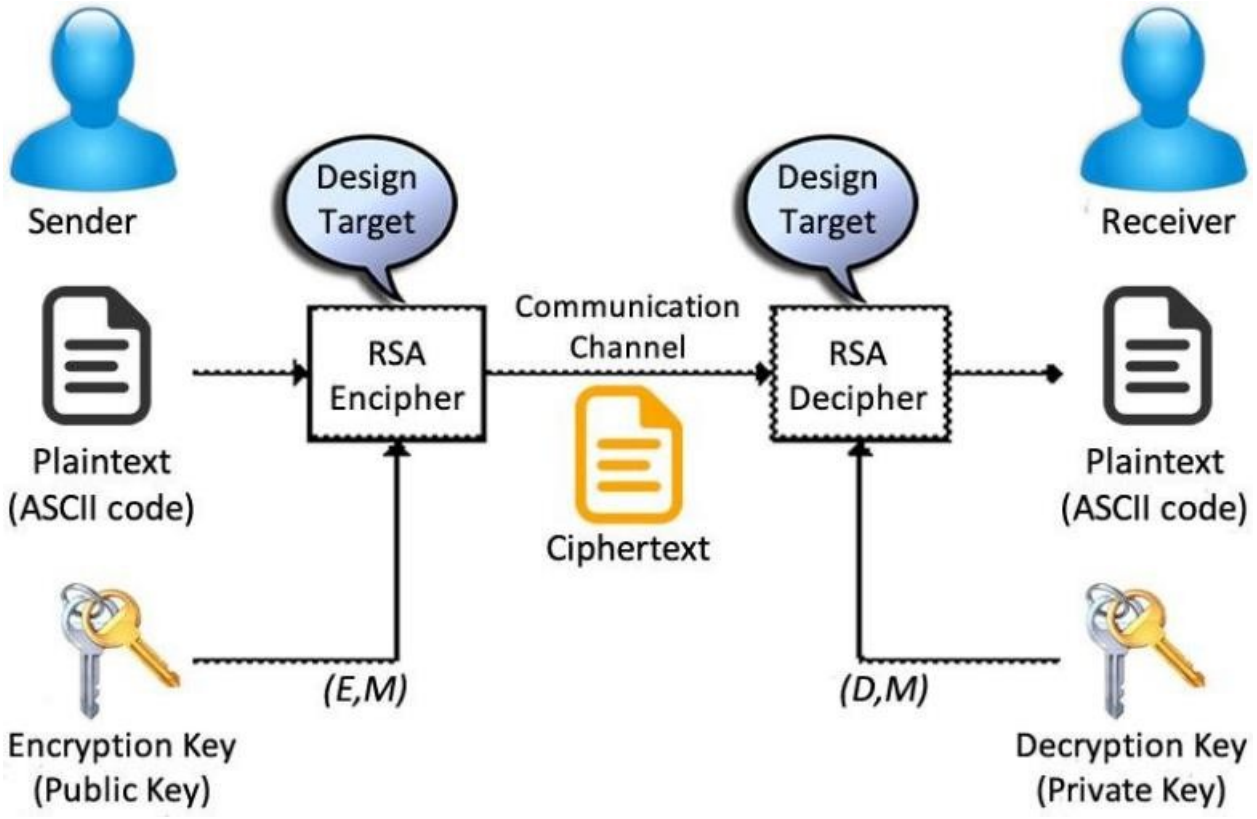
Public key: It comprises of two numbers, in which one number is the result of the product of two large prime numbers. This key is provided toall the users.

Private key: It is derived from the two prime numbers involved in publickey and it always remains private.

Applications of Methodology

We aimed our project not for a specific age group or sex but a community of all members of any organisation in general, It can be members of a hospital (i.e.: doctors and nurses) or software company (i.e. employees, etc.) or a school (i.e.: teachers and admin) who can share sensitive content among themselves or with other similarorganisations.

This system can also be helpful for government bodies to communicate with each other by creating chat room at different hierarchy level so as to maintain privacy while communication and still being efficient aboutit. Using public chatrooms messages can be sent to whole organisations and while using private chat rooms ONLY a targeted audience receives it and even if the message is caught by attackers it cannot be understoodas it is encrypted



Objectives :

To provide secure channel of communication for clients belonging to

organizations. Ensure integrity and privacy of messages. Provide private chatting rooms to share data and sensitive information for employees or clients. Building web application to give clients flexibility for access. Ensure each message in special secured text so as even if any hackers tap into data-links deciphering the text is not easy.

Codes:

Forms.py

```
from django import forms
from .models import messageModel, roomModel, userModel
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

#### FORM CREATION
class SignUpForm(UserCreationForm):
    first name = forms.CharField(max length=30, required=False,
help_text='Optional.')
    last name = forms.CharField(max length=30, required=False,
help_text='Optional.')

    class Meta:
        model = userModel
        fields = ('username', 'first_name', 'last_name', 'password1', 'password1',)

class messageForm(forms.ModelForm):

    class Meta:
        model = messageModel
        fields = ('text', 'bits')

    def   init  (self, *args, **kwargs):
        super(messageForm, self).  init  (*args, **kwargs)
        self.fields['text'].widget.attrs\
            .update({
                'id': 'chat-message-input',
                'placeholder': 'Enter your message here\n\nAlways encrypt after typing
your message'
            })

class roomForm(forms.ModelForm):
    class Meta:
        model = roomModel
        fields = ('roomName', 'roomType')

class passwordForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    class Meta:
        model = roomModel
        fields = ('password',)

    def save(self, commit=True):
        # Save the provided password in hashed format
        room = super(passwordForm, self).save(commit=False)
        room.set password(self.cleaned data["password"])
        if commit:
            room.save()
        return room

class verificationForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    class Meta:
        model = roomModel
        fields = ('password',)
```

Views.py

```
import random, json
from . import crypt, genLargePrimes
from .models import messageModel, roomModel, userModel
from .forms import messageForm, roomForm, SignUpForm, passwordForm,
verificationForm
from django.http import HttpResponseRedirect
from django.utils.safestring import mark safe
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate
from django.contrib.auth.models import User
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from datetime import datetime

def register(request):
    if(request.method == 'POST'):
        form = SignUpForm(request.POST)
        if form.is valid():
            form.save()
            uname = form.cleaned data.get('username')
            raw pass = form.cleaned data.get('password1')
            user = authenticate(username=uname, password=raw pass)
            login(request, user)
            return redirect('login')
    else:
        form = SignUpForm()
    return render(request, 'rsaDemo/register.html', {'form' : form})

@login required
def home(request):
    return render(request, 'rsaDemo/home.html')

@login_required
def joinRoom(request):
    rooms = roomModel.objects.all()
    context = {
        'rooms' : rooms,
    }
    return render(request, 'rsaDemo/joinRoom.html', context)


def privateChat(request):
    users = userModel.objects.all()
    context = {
        'users' : users,
    }
    return render(request, 'rsaDemo/privateChat.html', context)

@login required
def createRoom(request):
    form = roomForm(request.POST)
    if(request.method == 'POST'):
        if form.is valid():
            roomName = form.cleaned data['roomName']
            if not roomModel.objects.filter(roomName=roomName).exists():
                form.save()
                roomType = form.cleaned data['roomType']
                if roomType == 'private':
                    return redirect('rsaDemo:putPassword', room name=roomName)

                messages.success(request, 'Room successfully created')
                return redirect('rsaDemo:room', room_name=roomName)
```

```
            else:
                messages.info(request, 'Room already exists')
                return redirect('rsaDemo:createRoom')
        else:
            messages.warning(request, 'Please correct the error below.')
    else:
        form = roomForm()
    return render(request, 'rsaDemo/createRoom.html', {'form': form})



def putPassword(request, room_name):
    room = roomModel.objects.get(roomName=room_name)
    form = passwordForm(request.POST)
    if request.method == 'POST':
        if form.is_valid():
            password = form.cleaned_data.get('password')
            room.password = password
            room.save()
            return redirect('rsaDemo:room', room_name=room_name)
    else:
        form = passwordForm()
    context = {
        'room' : room,
        'form' : form,
    }
    return render(request, 'rsaDemo/putPassword.html', context)

@login_required
def room(request, room_name):
    form = messageForm(request.POST)
    if request.method == 'POST':
        if form.is_valid():
            plaintext = form.cleaned_data['text']
            bitlength = form.cleaned_data['bits']
            form.save()
            rsaObj = messageModel.objects.get(text=plaintext)
            rsaObj.date = datetime.now()
            rsaObj.sender = request.user.username
            rsaObj.room = roomModel.objects.get(roomName=room_name)
            rsaObj.save(update_fields=['date', 'sender', 'room'])
            n, phi, private_key, public_key = crypt.runRSA(bitlength)
            ct_list, ciphertext = crypt.encrypt(plaintext, public_key, n)
            deciphered_text = crypt.decrypt(ct_list, ciphertext, private_key, n)
            rsaObj.modulus = n
            rsaObj.cipherText = ciphertext
            rsaObj.ctlist = json.dumps(ct_list)
            rsaObj.publicKey = str(public_key)
            rsaObj.phi = str(phi)
            rsaObj.save(update_fields=['modulus', 'cipherText', 'ctlist', 'publicKey',
'phi'])

            messages.success(request, "Message encrypted successfully. Your Private
key is : " + str(private_key))
            context = {
                'form' : form,
                'plaintext' : plaintext,
                'ciphertext' : ciphertext,
                'ct_list' : ct_list,
                'modulus' : n,
                'phi' : phi,
                'private_key' : private_key,
                'public_key' : public_key,
                'room_name_json': (mark_safe(json.dumps(room_name))),
            }
```

```
        print("public key : ", publickey )
        print("totient : ", totient)
        print("PRIVATE KEY : ", privateKey)

        jsonDec = json.decoder.JSONDecoder()
        ct_list = jsonDec.decode(item.ctlist)

        deciphered text = crypt.decrypt(ct list, cipherText , privateKey, modulus)
        message dict = {
            'ciphertext' : str(item.cipherText),
            'sender' : str(item.sender),
            'time' : str(date),
            'deciphered text' : str(deciphered text),
            'bits' : bits,
        }
        data.append(message dict)
        item.read = True
        item.save(update_fields=['read'])

    context = {
        'data' : data,
        'flag' : flag,
        'room' : room name,
    }

    print(context)

    return render(request, 'rsaDemo/decrypt.html', context)

@login_required
def chatlog(request, room name):
    room = roomModel.objects.get(roomName=room_name)
    messages_ = room.messagemodel set.all().filter(read=False)
    messages = room.messagemodel_set.all().filter(read=True).order_by('-date')

    if len(messages_):
        flag=True
    else:
        flag=False

    if len(messages) == 0:
        flag1=True
    else:
        flag1=False

    chat_log = []
    for item in messages:
        bits = item.bits
        sender = item.sender
        ciphertext = item.cipherText
        publicKey = item.publicKey
        time = item.date
        plaintext = item.text

        chat = {
            'bits' : bits,
            'sender' : sender,
            'ciphertext' : ciphertext,
            'publicKey' : publicKey,
            'time' : time,
            'plaintext' : plaintext,
        }

        chat_log.append(chat)
```

```
    context = {
        'chatlog':chat_log,
        'flag':flag,
        'flag1':flag1,
        'room' : room_name,
    }

    return render(request, 'rsaDemo/chatlog.html', context)
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User
from datetime import datetime
bitlength_choices = (
    (128, '128 bits'),
)
#### Room type choice is choosen here
roomType_choices = (
    ("private", "private"),
    ("public", "public"),
)
class userModel(User):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    uid = models.BigAutoField(primary_key=True)

    def __str__(self):
        return self.user.username
########## Room details
class roomModel(models.Model):
    roomId = models.BigAutoField(primary_key=True)
    roomName = models.CharField(max_length=200, unique=True)
    roomType = models.CharField(max_length=50,choices=roomType_choices,
default="private")
    createdOn = models.DateTimeField(default=datetime.now, blank=False)
    password = models.CharField(max_length=100, default="")

    def __str__(self):
        return self.roomName

class createsModel(models.Model):
    uid = models.ForeignKey(userModel, on_delete=models.CASCADE) # needs to be a
user
    roomId = models.ForeignKey(roomModel, on_delete=models.CASCADE) # needs to be a
room
### Details regarding message of particualr room
class messageModel(models.Model):
    messageId = models.BigAutoField(primary_key=True)
    room = models.ForeignKey(roomModel, on_delete=models.CASCADE, null=True)
    text = models.TextField(blank=True, null=True)
    bits = models.IntegerField(choices=bitlength_choices, default='128')
    cipherText = models.TextField(blank=True, null=True)
    sender = models.CharField(max_length=200, default="")
    date = models.DateTimeField(default=datetime.now, editable=True)
    read = models.BooleanField(default=False)
    ctlist = models.TextField(blank=True, null=True)
    publicKey = models.TextField(blank=True, null=True)
    phi = models.TextField(blank=True, null=True)
    modulus = models.TextField(blank=True, null=True)

class roomHasMessages(models.Model):
    roomId = models.ForeignKey(roomModel, on_delete=models.CASCADE) # needs to be a
room
```

```
    messageId = models.ForeignKey(messageModel, on_delete=models.CASCADE) # needs to
be a room
```

Gen-large-prime:

```
from random import randrange, getrandbits
def is_prime(n, k=128):
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 2 == 0:
        return False
    # find r and s
    s = 0
    r = n - 1
    while r & 1 == 0:
        s += 1
        r //= 2
    # do k tests
    print("\nPerforming Rabin Miller's Primality Test\n")
    for   in range(k):
        a = randrange(2, n - 1)
        print("Selecting a random number a : ", a)
        x = pow(a, r, n)
        print("Calculating a=n(mod r) : ", x)
        if x != 1 and x != n - 1:
            j = 1
            while j < s and x != n - 1:
                x = pow(x, 2, n)
                print("Calculating x=n(mod 2) : ", x)
                if x == 1:
                    return False
                j += 1
            if x != n - 1:
                return False
    return True
def generate prime candidate(length):
    p = getrandbits(length)
    p |= (1 << length - 1) | 1
    return p
def generate_prime_number(length):
    p = 4
    while not is_prime(p, 128):
        p = generate prime candidate(length)
        print("Generating a prime candidate : ", p)
    return p
```

Crypt.py:

```
import random, os
from . import genLargePrimes

def Extended euclid(a, b):
   x0, x1, y0, y1 = 0, 1, 1, 0

   while a != 0:
      q, b, a = b//a, a, b%a
      y0, y1 = y1, y0 - q*y1
      x0, x1 = x1, x0 - q*x1

   return b, x0, y0
```

```
def generatePublicKey(totient):
    public key = random.randrange(3, totient)
    while not genLargePrimes.is_prime(public_key):
        print("Generating public key : ", public key)
        public_key += 1
    return public key


def generatePrivateKey(public_key, totient):
    g,  , private key = Extended euclid(int(totient), int(public key))
    if private_key > totient:
        private key = private key % totient
    elif private key < 0:
        private key += totient
    return private_key

def runRSA(message, bits):
    p = genLargePrimes.generate prime number(bits)
    q = genLargePrimes.generate_prime_number(bits)
    n = p*q
    totient = (p-1)*(q-1)
    public key = generatePublicKey(totient)
    private key = generatePrivateKey(public key, totient)
    print("Public key: ", public key)
    print("Private key: ", private key)
    enc_list = []
    decrypted mess = ""

    for char in message:
        mess = ord(char)
        enc mess = str(pow(mess, public key, n))
        enc list.append(enc mess)

    print("Cipher Text : ", enc mess)

    for enc mess in enc list:
        decr = (pow(int(enc_mess), private_key, n))
        decrypted mess += chr(decr)

    print("Decrypted Text: ", decrypted mess)

def runRSA(bits):
    p = genLargePrimes.generate prime number(bits)
    q = genLargePrimes.generate_prime_number(bits)
    n = p*q
    totient = (p-1)*(q-1)
    public key = generatePublicKey(totient)
    private key = generatePrivateKey(public key, totient)
    print("Prime numbers are\np : ", p, "\nq : ", q)
    print("Modulus : ", n)
    print("Euler's Totient : ", totient)
    print("Public key: ", public key)
    print("Private key: ", private_key)
    return n, totient, private key, public key


def encrypt(message, public_key, n):
    enc list = []
    print("\nEncrypting your message\n")
    for char in message:
        mess = ord(char)
        enc mess = str(pow(mess, public key, n))
        print(enc_mess)
        enc_list.append(enc_mess)
```

```
    return enc_list, enc_mess

def decrypt(ct_list, ct, private_key, n):
    decrypted_mess = ""
    print("\nDecrypting your message\n")
    for ct in ct_list:
        decr = (pow(int(ct), private_key, n))
        print(decr)
        decrypted_mess += chr(decr)

    return decrypted_mess


# if __name__ == '__main__':
#  bits = int(input("Enter bits: "))
#  n, totient, private_key, public_key = runRSA(bits)
#  message = input("Enter message: ")
#  ct_list, ct = encrypt(message, public_key, n)
#  print("Cipher Text: ", ct)
#  pt = decrypt(ct_list, ct, private_key, n)
#  print("Plain Text: ", pt)
```

Manage.py:

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'websec.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```