- **Réaliser par : - Oussama Elmaimouni .**
- **Filière : Science math Informatique**
- **Année : 2021 / 2022**
- **Encadre avec : El Mekkaoui .**
- **Université : Mohamed Première Oujda.**


# MINI PROJET :JAVA

## ENVOI ET RECEPTION D'E-MAILS SÉCURISÉS EN JAVA

# I –protocole SMTP .

## 1.Introducion

## 1.2 Quelques abréviations à connaître

## 1.3 Le cheminement d'un email

# II- Protocole POP3.

## 1.1 Définition

# III-La chiffrement et la Déchiffrement en utilise RSA .

## 1.1 Fonctionnement général.

## 1.2 Une schéma de RSA

# IV – Exemple d'exécution de Code JAVA

## I –protocole SMTP .

- ***Introduction***


**Si vous envoyez des emails marketing ou transactionnels, vous avez forcément entendu parler de SMTP. Un problème a priori très technique, mais qui vous concerne aussi puisqu'il impacte directement vos emails et leur délivrabilité.**
**Pourtant, rien d'insurmontable ! A la fin de cet article, on vous promet que vous comprendrez enfin de quoi il s'agit et que vous pourrez briller dans les discussions sur le sujet.**

**Pourquoi vous n'y comprenez rien :** tout simplement parce que le sigle « SMTP » est utilisé sans distinction pour désigner plusieurs choses différentes, alors forcément on s'emmêle les pinceaux.
On peut entendre par SMTP :

- Un protocole (sens strict)
- Un « relais »
- Un serveur

***SMTP** est le sigle de **Simple Mail Transfer Protocol**, littéralement « protocole simple de transfert de courrier ». Il s'agit du protocole utilisé pour transférer des emails vers les serveurs de messagerie électronique. Par extension, SMTP désigne aussi les serveurs et relais utilisés pour le transfert de ces emails.*

# Protocole SMTP

SMTP est d'abord un **protocole de communication**, c'est à dire un ensemble de règles utilisées pour transférer des emails vers les serveurs de messagerie électronique.
Concrètement, le protocole SMTP fonctionne de la manière suivante :

- On spécifie l'expéditeur puis le ou les destinataires du message
- On verifier leur existence
- On transfère le corps du message

- Quelques abbreviations à connaître :

- **MUA (Mail User Agent) :** client de messagerie (par exemple Gmail ou Outlook)

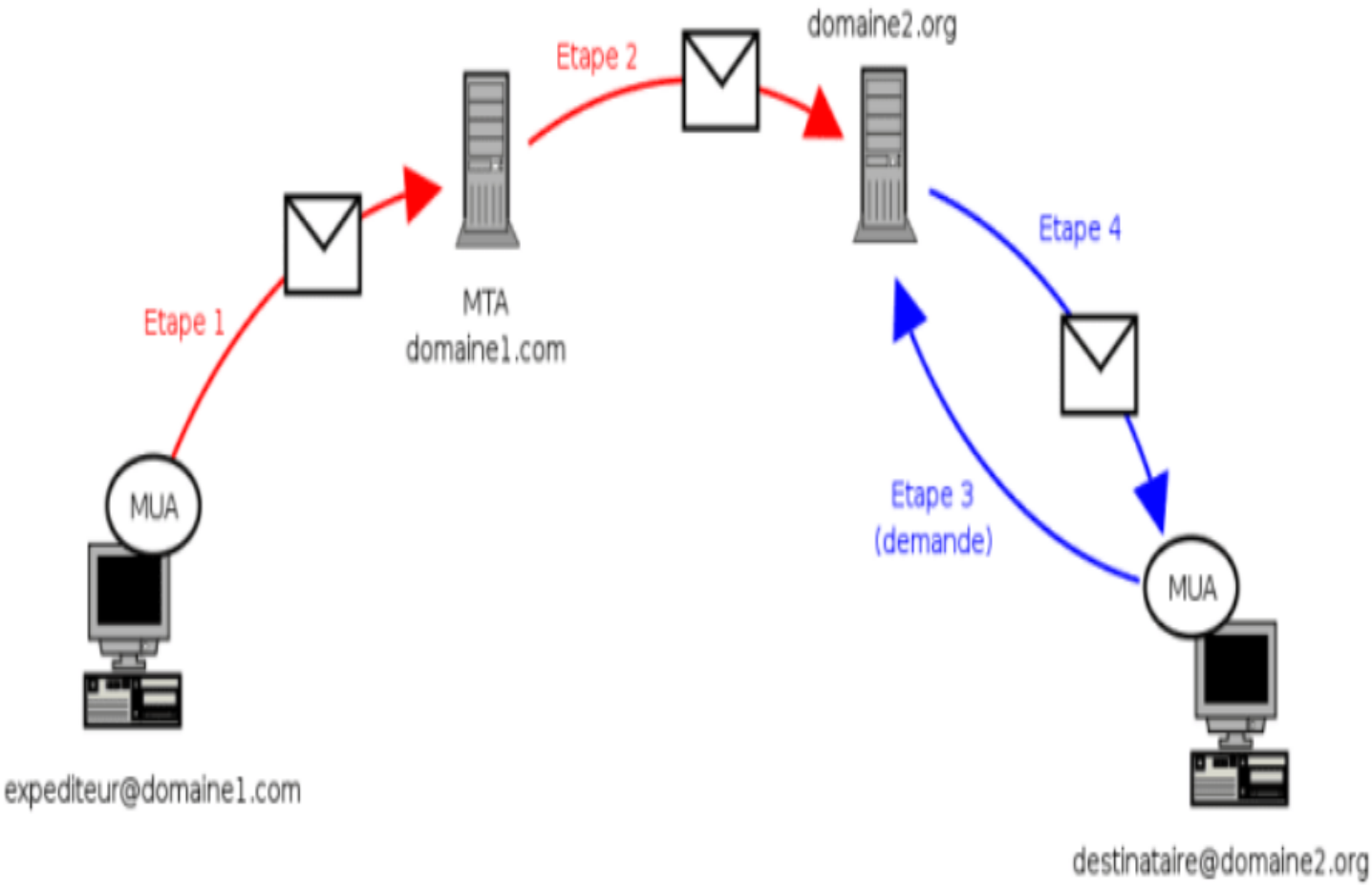
- **MTA (Mail Transfer Agent) :** logiciel qui reçoit les mails d'un MUA sur un serveur de transmission

- **MDA (Mail Delivery Agent) :** logiciel qui stocke les messages sur la boîte de réception des destinataires

- **Le cheminement d'un email peut se décomposer en 4 étapes :**

- Etape 1 : transfert depuis le MUA de l'envoyeur vers le serveur de transmission (MTA) (via le protocole SMTP)
- Etape 2 : transfert du mail de MTA en MTA jusqu'à celui du destinataire (via le protocole SMTP)
- Etape 3 : transfert depuis le MTA jusqu'au MDA qui stocke l'email en attendant qu'il soit relevé
- Etape 4 : requête du MUA destinataire au MDA pour relever ses mails

Ce sera peut-être plus clair avec un schéma :



# II- Protocole POP3.

## Tout savoir sur le protocole de messagerie POP3 (Post Office Protocol)

Le **POP est le Post Office Protocol**, un des trois principaux **protocoles de messagerie**. Aujourd'hui, c'est la version 3 qui est utilisée. Il s'agit ici de se connecter à un serveur TCP/IP afin de collecter les messages sur le serveur puis de les effacer et se déconnecter. On utilise pour cela le port 110. Il existe également le POP3S ou POP3 over SSL qui est plus sécurisé que l'original. Tout l'intérêt de ce protocole est de permettre aux utilisateurs de pouvoir consulter les mails qu'ils ont reçus lorsqu'ils n'étaient pas connectés. Le POP3 et le SMTP fonctionnent sur le même principe, à savoir que des commandes textuelles sont utilisées pour faire transiter les mails. La différence fondamentale entre ces deux systèmes est l'authentification.
De fait, si le **SMTP** ne permet pas d'identifier l'expéditeur, le protocole de messagerie POP3 est à même de gérer cela grâce à l'utilisation d'un identifiant et d'un mot de passe. Les utilisateurs de gestionnaires de courrier électronique, à l'image de Microsoft Outlook qui peut paramétrer leurs comptes de messagerie, sont familiers avec ce système. En revanche, la sécurité n'est pas sans faille puisque les mots de passe ne sont pas cryptés.

# III-La chiffrement et la Déchiffrement en utilise RSA .

## . Chiffrement RSA .

Le chiffrement RSA (nommé par les initiales de ses trois inventeurs) est un algorithme de cryptographie asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. RSA a été breveté[1] par le Massachusetts Institute of Technology (MIT) en 1983 aux États-Unis. Le brevet a expiré le 21 septembre 2000.

## . *Fonctionnement général*

**Le chiffrement RSA est *asymétrique* : il utilise une paire de clés (des nombres entiers) composée d'une *clé publique* pour chiffrer et d'une *clé privée* pour déchiffrer des données confidentielles. Les deux clés sont créées par une personne, souvent nommée par convention *Alice*, qui souhaite que lui soient envoyées des données confidentielles. Alice rend la clé publique accessible. Cette clé est utilisée par ses correspondants (*Bob*, etc.) pour chiffrer les données qui lui sont envoyées. La clé privée est quant à elle réservée à Alice, et lui permet de déchiffrer ces données. La clé privée peut aussi être utilisée par Alice pour signer une donnée qu'elle envoie, la clé publique permettant à n'importe lequel de ses correspondants de vérifier la signature.**

**Une condition indispensable est qu'il soit « calculatoirement impossible » de déchiffrer à l'aide de la seule clé publique, en particulier de reconstituer la clé privée à partir de la clé publique, c'est-à-dire que les**

**moyens de calcul disponibles et les méthodes connues au moment de l'échange (et le temps que le secret doit être conservé) ne le permettent pas. Le chiffrement RSA est souvent utilisé pour communiquer une clé de chiffrement symétrique, qui permet alors de poursuivre l'échange de façon confidentielle : Bob envoie à Alice une clé de chiffrement symétrique qui peut ensuite être utilisée par Alice et Bob pour échanger des données.**

---

## *Création des clés*

**L'étape de création des clés est à la charge d'Alice. Elle n'intervient pas à chaque chiffrement car les clés peuvent être réutilisées. La difficulté première, que ne règle pas le chiffrement, est que Bob soit bien certain que la clé publique qu'il détient est celle d'Alice. Le renouvellement des clés n'intervient que si la clé privée est compromise, ou par précaution au bout d'un certain temps (qui peut se compter en années).**

- Choisir $p$ et $q$, deux nombres premiers distincts ;
- calculer leur produit $n = pq$, appelé *module de chiffrement* ;
- calculer $\varphi(n) = (p - 1)(q - 1)$ (c'est la valeur de l'indicatrice d'Euler en $n$) ;
- choisir un entier naturel $e$ premier avec $\varphi(n)$ et strictement inférieur à $\varphi(n)$, appelé *exposant de chiffrement* ;
- calculer l'entier naturel $d$, inverse de $e$ modulo $\varphi(n)$, et strictement inférieur à $\varphi(n)$, appelé *exposant de déchiffrement* ; $d$ peut se calculer efficacement par l'algorithme d'Euclide étendu.

**Comme $e$ est premier avec $\varphi(n)$, d'après le théorème de Bachet-Bézout il existe deux entiers $d$ et $k$ tels que $ed = 1 + k\varphi(n)$, c'est-à-dire que $ed \equiv 1 \pmod{\varphi(n)}$ : $e$ est bien inversible modulo $\varphi(n)$.**

Dans tout le paragraphe précédent, on peut utiliser l'indicatrice de Carmichael, qui divise φ(*n*).

Le couple (*n*, *e*) — ou (*e*, *n*)[3] — est la *clé publique* du chiffrement, alors que sa *clé privée* est le nombre *d*, sachant que l'opération de déchiffrement ne demande que la clef privée *d* et l'entier *n*, connu par la clé publique (la clé privée est parfois aussi définie comme le couple (*d*, *n*)[3] ou le triplet (*p, q*, *d*)[5]).

- *Chiffrement du message*

Si $M$ est un entier naturel strictement inférieur

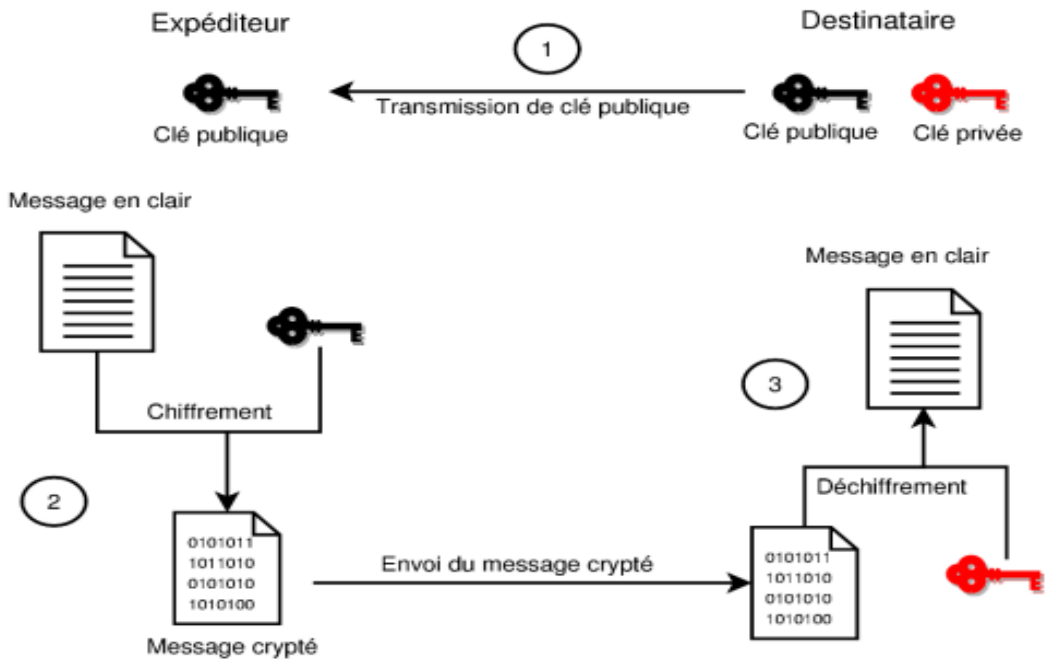
à *n* représentant un message, alors le message chiffré sera représenté par

L'entier naturel *C* étant choisi strictement inférieur à *n*.

- *Déchiffrement du message*

Pour déchiffrer *C*, on utilise *d*, l'inverse de *e* modulo (*p* – 1)(*q* – 1), et l'on retrouve le message clair *M* par

## 1.2 Une schéma de RSA :




---

## IV – Exemple d'exécution de Code JAVA

- Voici les bibliothèques que nous devons importer

```
5   package javaapplication1;
6
7   import com.sun.mail.pop3.POP3Store;
8   import java.io.IOException;
9   import java.math.BigInteger;
10  import java.util.Properties;
11  import java.util.Random;
12  import javax.mail.Folder;
13  import javax.mail.Message;
14  import javax.mail.MessagingException;
15  import javax.mail.NoSuchProviderException;
16  import javax.mail.PasswordAuthentication;
17  import javax.mail.Session;
18  import javax.mail.Transport;
19  import javax.mail.internet.InternetAddress;
20  import javax.mail.internet.MimeMessage;
21  import javax.swing.JOptionPane;
22
23  /**
```

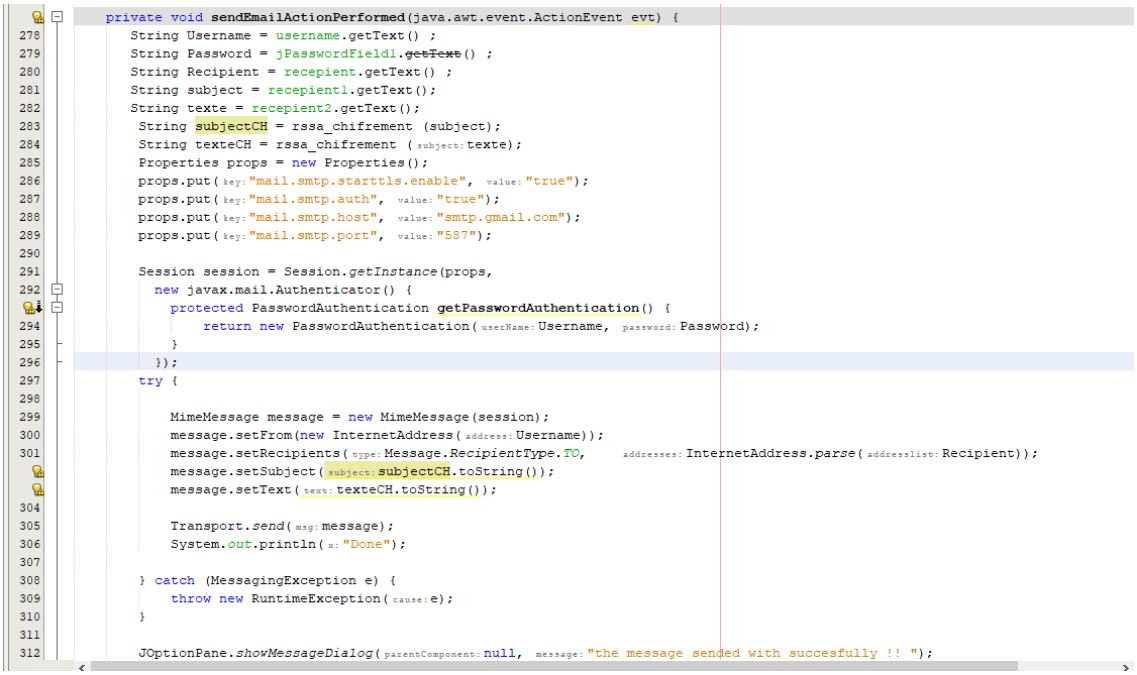
*Ç'est ça le fonction  qui le chiffrer le message :*

```
public String rssa_chifrement (String subject) {
    //___________//generation p,q,n,f,e,d_____________________________________________________
            BigInteger p= new BigInteger( val: "11");
            BigInteger q= new BigInteger( val: "13");
            BigInteger n=p.multiply( val: q);
            BigInteger f=p.subtract( val: BigInteger.ONE).multiply( val: q.subtract( val: BigInteger.ONE));

            BigInteger e=new BigInteger( val: "47");
            BigInteger d=e.modInverse( m: f);
//__________Convertir du message en tableau_____________________________________________________
            char[] charmsg=subject.toCharArray();
            int msgascii[]=new int[subject.length()];
            for(int i=0;i<subject.length();i++) {
                msgascii[i]=(int)charmsg[i];
            }
//_____________//Chiffrement_____________________________________________________
            int[] msgcrypt=new int[subject.length()];
            for(int i=0;i<subject.length();i++) {
                BigInteger nb=BigInteger.valueOf(msgascii[i]);
                msgcrypt[i]=nb.modPow( exponent: e,  m: n).intValue();
            }
            String msg="";
            for(int i  = 0;i<msgcrypt.length;i++){
                msg += (char)msgcrypt[i];
            }

//      System.out.println("le message crypte est : "+msgcrypt.toString());
    return msg;
```

Ç'est ça le fonction  qui le déchiffrer le message :

```
public String dechifrer_rssa (String message) {
    //___________//generation p,q,n,f,e,d_____________________________________________________

            BigInteger p= new BigInteger( val: "11");
            BigInteger q= new BigInteger( val: "13");
            BigInteger n=p.multiply( val: q);
            BigInteger f=p.subtract( val: BigInteger.ONE).multiply( val: q.subtract( val: BigInteger.ONE));
            Random rnd=new Random() ;
            BigInteger e=new BigInteger( val: "47");
            BigInteger d=e.modInverse( m: f);
            char[] msgcrypt=message.toCharArray();
           String msg="";
           //on va dechifrer chaque valeur de le message crpter a laide de le cle privé
    for(int i=0;i<message.length();i++) {
                    BigInteger nb=BigInteger.valueOf(msgcrypt[i]);
                    int decode=nb.modPow( exponent: d,  m: n).intValue();
                    msg += (char)decode ;
                    //System.out.print((char)decode);
            }
          //  System.out.println();
            return msg ;
}
```

+ Ç'est ça le fonction qui l'envoyer un message chiffré à un destinataire a l'aide de protocole SMTP :

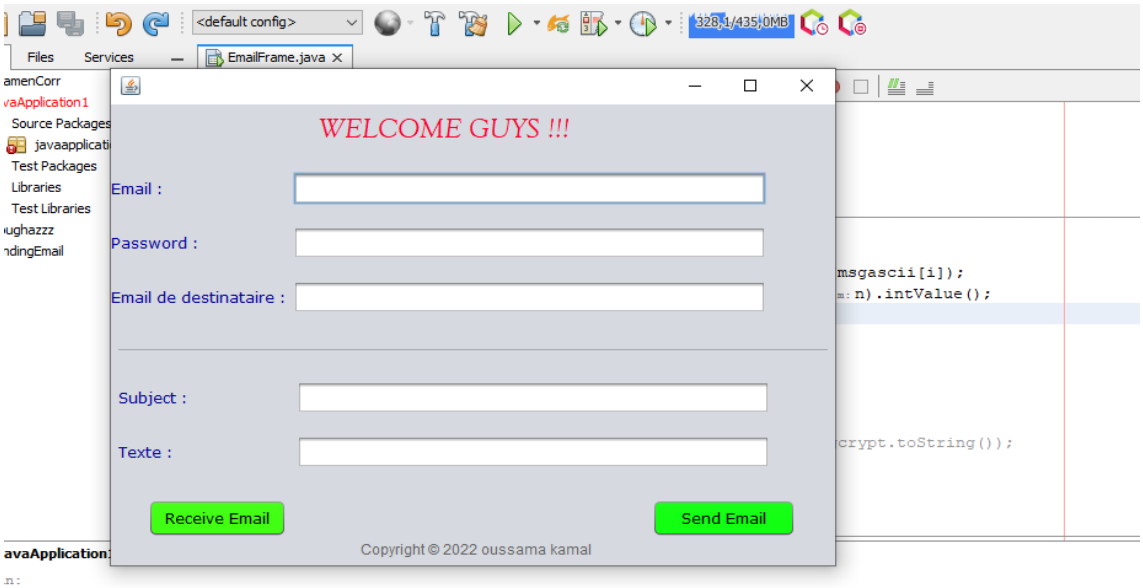


```
278     private void sendEmailActionPerformed(java.awt.event.ActionEvent evt) {
279         String Username = username.getText() ;
280         String Password = jPasswordField1.getText() ;
281         String Recipient = recepient.getText() ;
282         String subject = recepient1.getText();
283         String texte = recepient2.getText();
284          String subjectCH = rssa_chifrement (subject);
285          String texteCH = rssa_chifrement ( subject: texte);
286          Properties props = new Properties();
287          props.put( key: "mail.smtp.starttls.enable",  value: "true");
288          props.put( key: "mail.smtp.auth",  value: "true");
289          props.put( key: "mail.smtp.host",  value: "smtp.gmail.com");
290          props.put( key: "mail.smtp.port",  value: "587");
291 
292          Session session = Session.getInstance(props,
293            new javax.mail.Authenticator() {
294              protected PasswordAuthentication getPasswordAuthentication() {
295                  return new PasswordAuthentication( userName: Username,  password: Password);
296              }
297            });
298          try {
299 
300              MimeMessage message = new MimeMessage(session);
301              message.setFrom(new InternetAddress( address: Username));
302              message.setRecipients( type: Message.RecipientType.TO,     addresses: InternetAddress.parse( addressList: Recipient));
303              message.setSubject( subject: subjectCH.toString());
304              message.setText( text: texteCH.toString());
305 
306              Transport.send( msg: message);
307              System.out.println( x: "Done");
308 
309          } catch (MessagingException e) {
310              throw new RuntimeException( cause: e);
311          }
312 
          JOptionPane.showMessageDialog( parentComponent: null,  message: "the message sended with succesfully !! ");
```

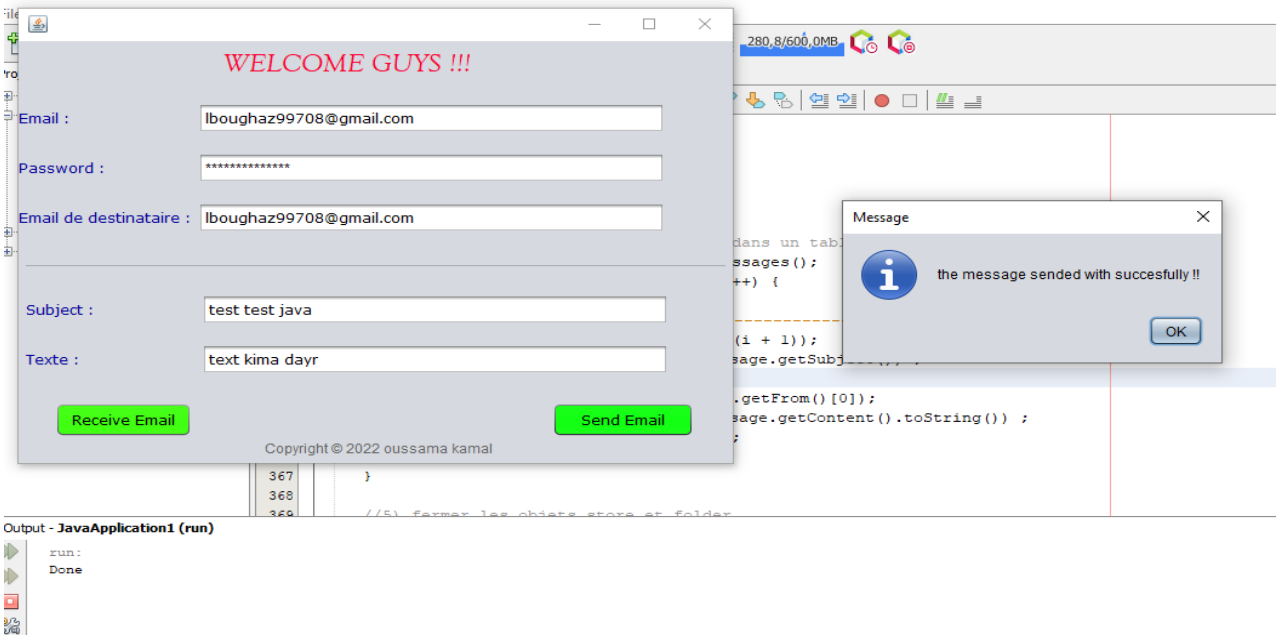
Ç'est ça le fonction qui le recevoirr un message déchiffré à partir d'un  expiditeur a l'aide de protocole pop3 :

```
    private void receiveEmailActionPerformed(java.awt.event.ActionEvent evt) {
    String pop3Host = "pop.gmail.com"; String storeType = "pop3s";
    String user = username.getText() ;  String Pass = jPasswordField1.getText() ;
try {

  //1) get the session object
  Properties properties = new Properties();
  properties.put( key: "mail.pop3.host",  value: pop3Host); properties.put( key: "mail.pop3.port",  value: "995");
  properties.put( key: "mail.pop3.starttls.enable",  value: "true");
  Session emailSession = Session.getDefaultInstance( props: properties);

  //2) create the POP3 store object and connect with the pop server
  POP3Store emailStore = (POP3Store) emailSession.getStore( protocol: storeType);
  emailStore.connect( host: pop3Host,user,  password: Pass);

  //3) create the folder object and open it
  Folder emailFolder = emailStore.getFolder( name: "INBOX");
  emailFolder.open( i: Folder.READ_ONLY);
  //4)récupérer les messages du dossier dans un tableau et l'imprimer
  Message[] messages = emailFolder.getMessages();
  for (int i = 0; i < messages.length; i++) {
   Message message = messages[i];
   System.out.println( x: "---------------------------------");
   System.out.println("Email Number " + (i + 1));
   String x = dechifrer_rssa ( message: message.getSubject()) ;
   System.out.println("Subject: " +x );
   System.out.println("From: " + message.getFrom()[0]);
   String y = dechifrer_rssa ( message: message.getContent().toString()) ;
   System.out.println("Text: " + y+"\n");       }
  //5) fermer les objets store et folder
  emailFolder.close( bln: false);
  emailStore.close();
 } catch (NoSuchProviderException e) {e.printStackTrace();}
 catch (MessagingException e) {e.printStackTrace();}
 catch (IOException e) {e.printStackTrace();}
JOptionPane.showMessageDialog( parentComponent: null,  message: "the messages received  succesfully !! ");
```

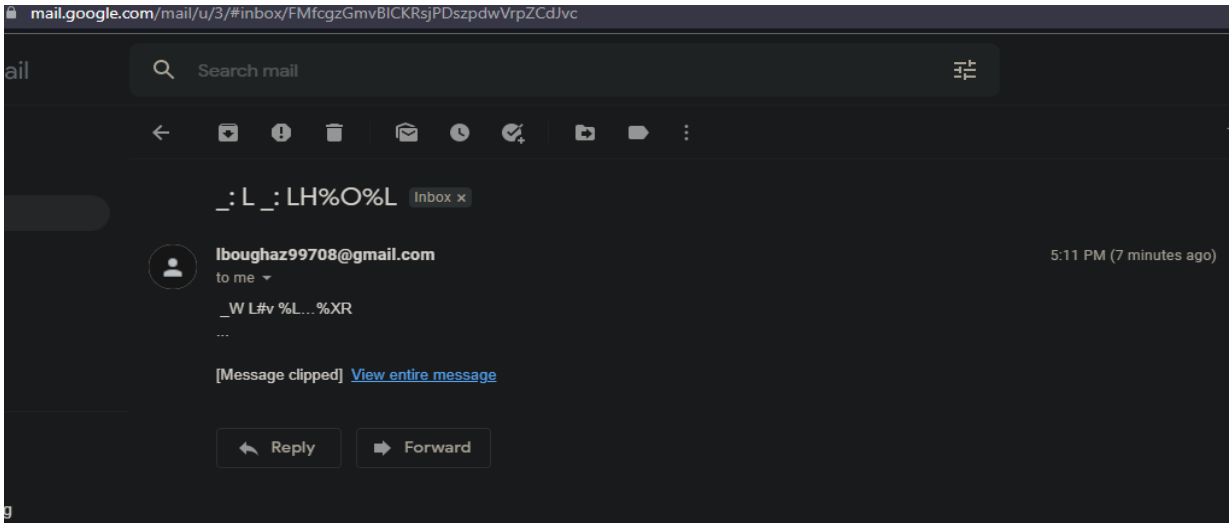
Ceci est l'interface graphique qui apparaît lorsque nous faisons RUN :



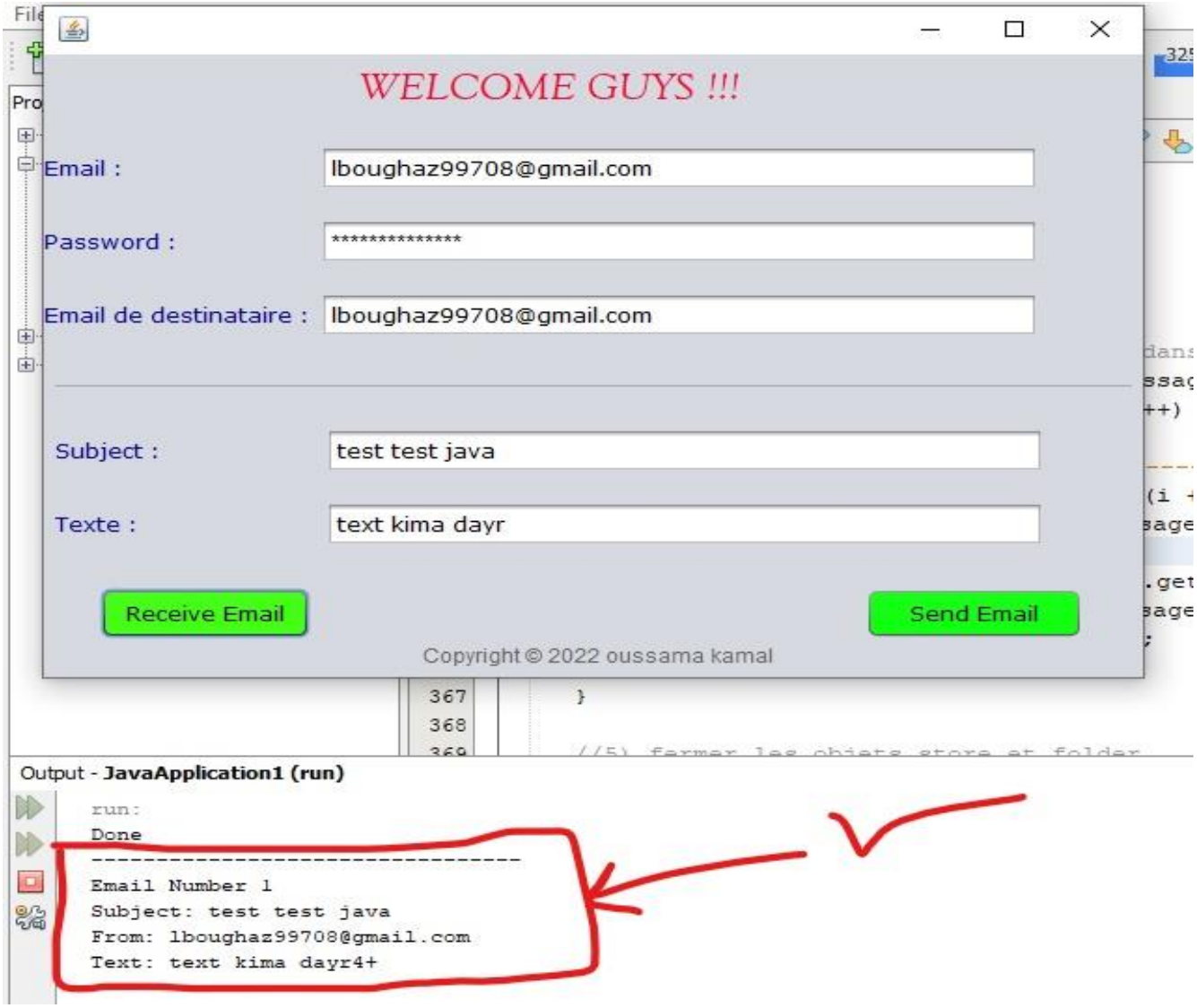

Nous remplissons les informations et cliquez sur button “send email ”. Et puis une fenêtre apparaît nous informant que le message a été envoyé :




Nous ouvrons gmail pour nous assurer que le message est déjà arrivé :



Maintenant, nous cliquons sur le button "receive




email " pour lire le contenu du message crypté que nous avons envo