



APPLICATION DE CHAT SECURISE AVEC L'ALGORITHME RSA

Réalisée par : **Haoua Amamatou Oumar MAIROU**

Encadre par : **Mme. Saïida LAAZAR**

Présentation du projet

Contexte

Dans le cadre d'un travail pratique de cryptographie, nous avons développé une application de chat sécurisé. Le langage utilisé est **python** et l'algorithme est **RSA**.

Outils utilisés

Python

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Nous avons utilisé les sockets pour connecter deux nœuds sur un réseau afin qu'ils communiquent entre eux et tkinter pour l'interface graphique.



Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOS. Nous avons également utilisé le terminal pour exécuter le code python.



Visual Studio Code

Définition des concepts

Chiffrement

procédé de cryptographie qui consiste à protéger des données qui sont alors incompréhensibles pour celui qui ne dispose pas de la clé du chiffrement.

Déchiffrement

Opération inverse du chiffrement, il s'agit du processus effectué pour déverrouiller vos fichiers cryptés.

Algorithme RSA

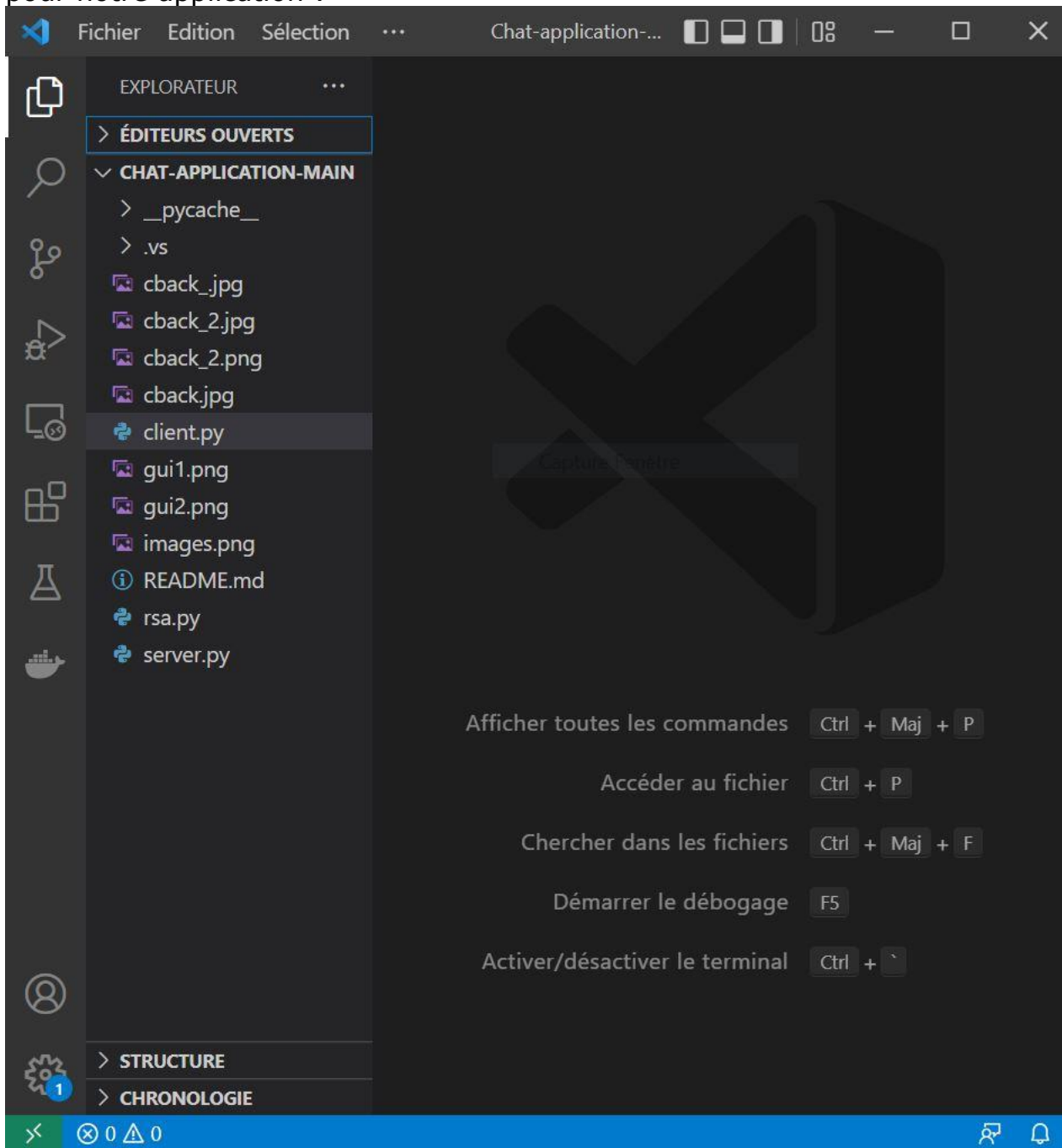
L'algorithme RSA est un algorithme de cryptographie asymétrique. Asymétrique signifie en fait qu'il fonctionne sur deux clés différentes, à savoir **la clé publique** et **la clé privée**. Comme son nom l'indique, la clé publique est donnée à tout le monde et la clé privée reste privée.

Exemple du processus de cryptage avec RSA

1. Un client (par exemple un navigateur) envoie sa clé publique au serveur et demande des données.
2. Le serveur crypte les données à l'aide de la clé publique du client et envoie les données cryptées.
3. Le client reçoit ces données et les décrypte.

Implémentation

Le dossier chat-application-main contient plusieurs fichiers et images que nous utilisons pour notre application :



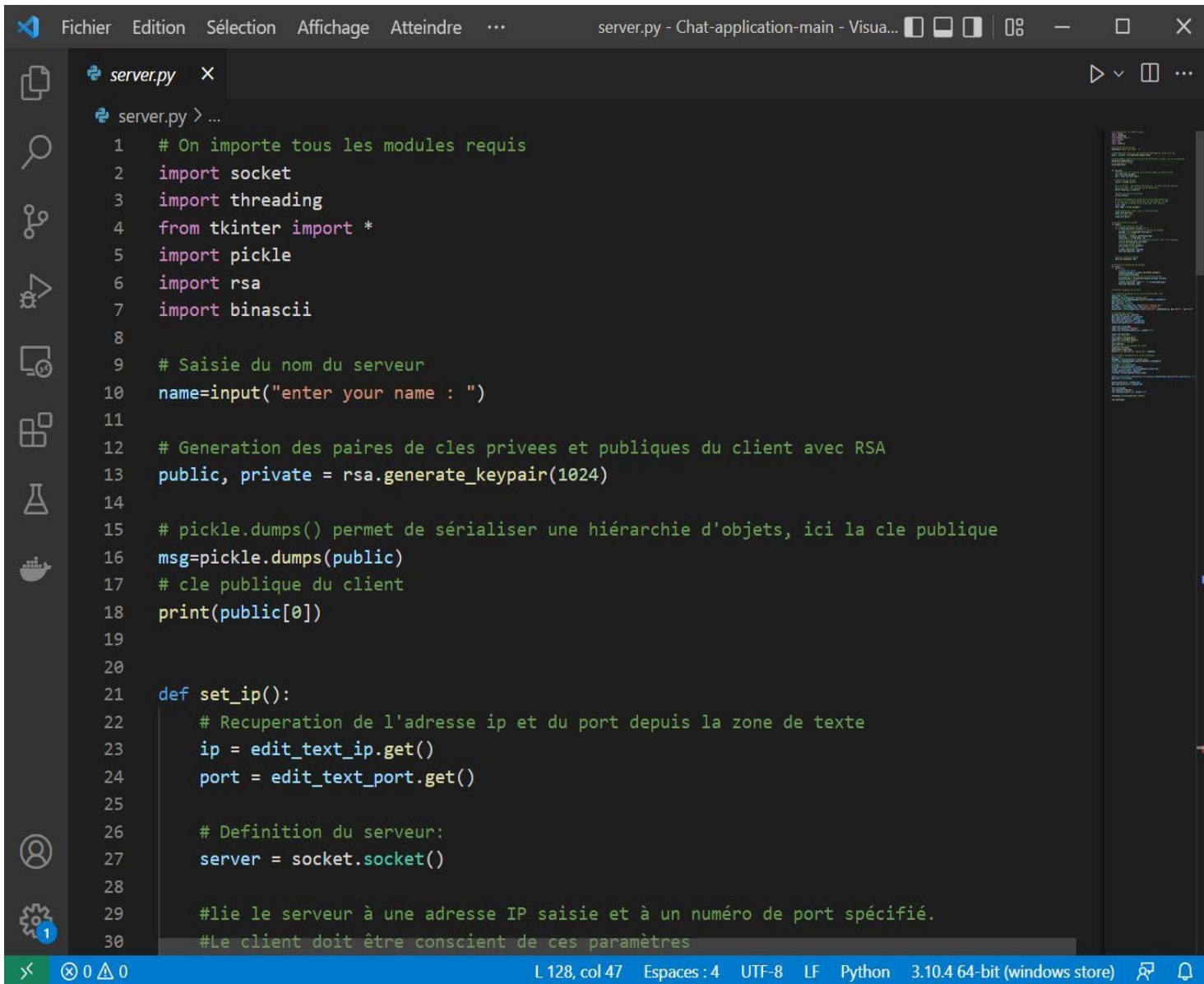
L'algorithme RSA est utilisé en important rsa du fichier rsa

```
1  from __future__ import unicode_literals
2  from math import sqrt
3  import random
4  from random import randint as rand
5  from random import getrandbits
6  import sympy
7  import time
8  import binascii
9
10 def gcd(a, b):
11     if b == 0:
12         return a
13     else:
14         return gcd(b, a % b)
15
16 def egcd(a, b):
17     x,y, u,v = 0,1, 1,0
18     while a != 0:
19         q, r = b//a, b%a
20         m, n = x-u*q, y-v*q
21         b,a, x,y, u,v = a,r, u,v, m,n
22     return b, x, y
23
24 def mod_inverse(a, m):
25     g, x, y = egcd(a, m)
26     if g != 1:
27         return None
28     else:
29         return x % m
30
```

```
47     return p
48
49
50 def generate_keypair(keysize):
51     p = generate_prime(keysize)
52     q = generate_prime(keysize)
53     n = p * q
54     phi = (p-1)*(q-1)//gcd(p-1, q-1)
55     e = sympy.randprime(1,phi)
56     d = mod_inverse(e,phi)
57     if e != d:
58         return ((e, n), (d, n))
59
60
61 def encrypt(plain_text, package):
62     e, n = package
63     if plain_text > n:
64         print('Message is too large for key to handle')
65     msg_ciphertext = pow(plain_text, e, n)
66     return msg_ciphertext
67
68
69 def decrypt(msg_ciphertext, package):
70     d, n = package
71     msg_plaintext = pow(msg_ciphertext, d, n)
72     return binascii.unhexlify(hex(msg_plaintext)[2:]).decode()
73
74
```

Programme Python pour implémenter le côté serveur de la salle de discussion

server.py avec explication en commentaires :



```
server.py  X
server.py > ...
1  # On importe tous les modules requis
2  import socket
3  import threading
4  from tkinter import *
5  import pickle
6  import rsa
7  import binascii
8
9  # Saisie du nom du serveur
10 name=input("enter your name : ")
11
12 # Generation des paires de cles privees et publiques du client avec RSA
13 public, private = rsa.generate_keypair(1024)
14
15 # pickle.dumps() permet de sérialiser une hiérarchie d'objets, ici la cle publique
16 msg=pickle.dumps(public)
17 # cle publique du client
18 print(public[0])
19
20
21 def set_ip():
22     # Recuperation de l'adresse ip et du port depuis la zone de texte
23     ip = edit_text_ip.get()
24     port = edit_text_port.get()
25
26     # Definition du serveur:
27     server = socket.socket()
28
29     #lie le serveur à une adresse IP saisie et à un numéro de port spécifié.
30     #Le client doit être conscient de ces paramètres
```

L 128, col 47 Espaces : 4 UTF-8 LF Python 3.10.4 64-bit (windows store)

```

31     server.bind((ip, int(port)))
32
33     #serveur en ecoute de connexion
34     server.listen()
35
36     #Accepte une demande de connexion et stocke deux paramètres,
37     # conn qui est un objet socket pour cet utilisateur, et addr
38     # qui contient l'adresse IP du client qui s'est connecté
39     global conn
40     conn, addr = server.accept()
41
42     # Destruction de l'input root ou racine d'entree
43     input_root.destroy()
44     # fin de input root:
45     input_root.quit()
46
47
48 # fonction d'envoi de message
49 def send():
50     # si la zone texte est non vide
51     if str(edit_text.get()).strip() != "":
52         # recuperation et conversion en octects du message
53         message = str.encode(edit_text.get())
54         #conversion en numb
55         hex_data = binascii.hexlify(message)
56         plain_text = int(hex_data, 16)
57
58         # chiffrement du texte brut(message en entier) avec la cle publique
59         ctt=rsa.encrypt(plain_text,pkey)
60         # envoi du message chiffré
61         conn.send(str(ctt).encode())
62         # barre de défilement:
63         listbox.insert(END, message)
64         edit_text.delete(0, END)
65
66     # Apres l'envoi du message
67     edit_text.delete(0, END)
68
69
70 # fonction de recuperation de message
71 def rcv():
72     while True:
73         # reponse du client
74         response_message =int(conn.recv(1024).decode())
75         print(response_message)
76         # decryptage du msg avec la clé privée du serveur
77         decrypted_msg = rsa.decrypt(response_message, private)
78         # barre de défilement::
79         listbox.insert(END, name1 +" : "+ str(decrypted_msg))
80         edit_text.delete(0, END)
81
82
83 # Interface graphique du serveur:
84

```



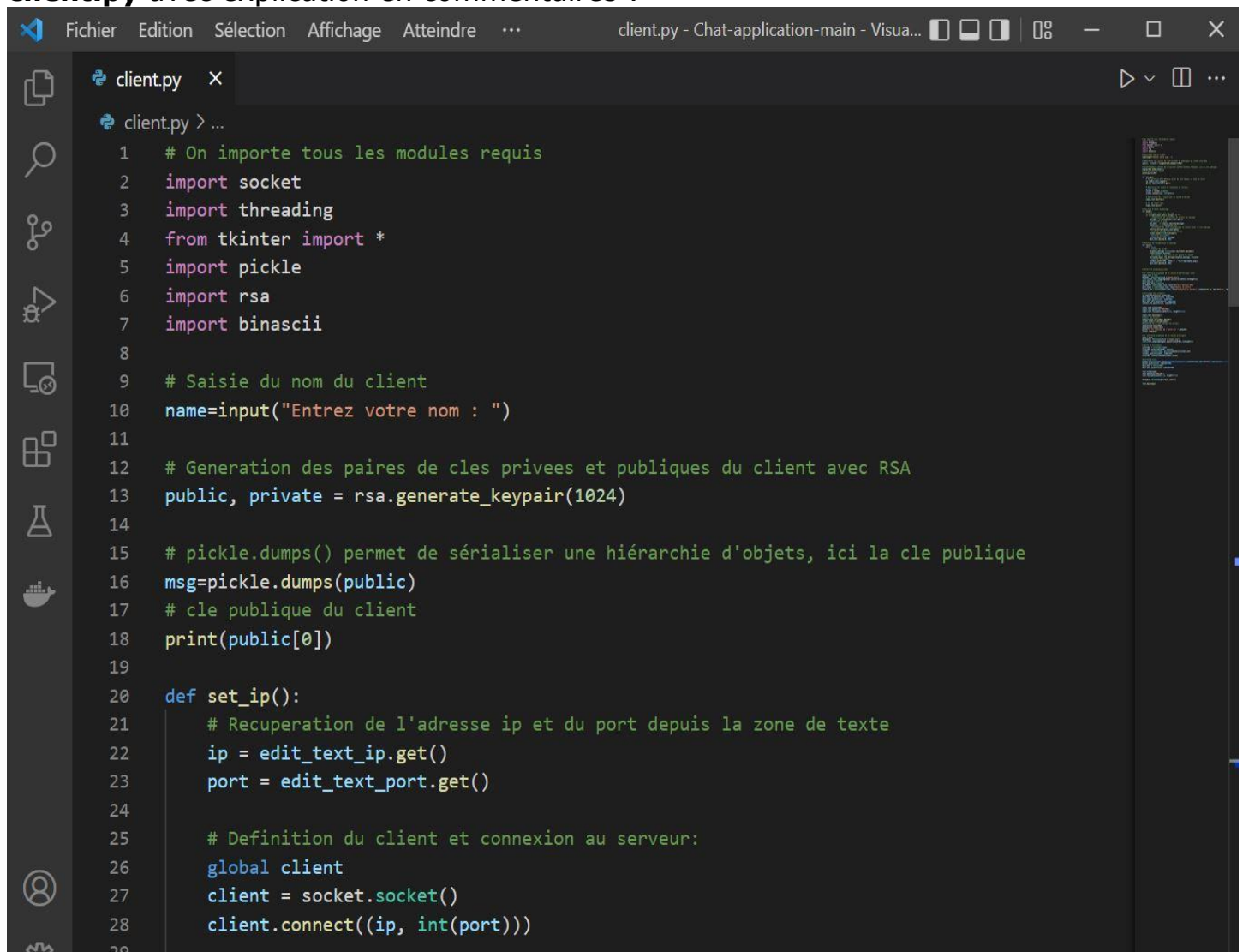
```

85 # 1: Interface graphique de la racine d'entrée(input root)
86 input_root = Tk()
87 bgimage = PhotoImage(file="images.png")
88 Label(input_root,image=bgimage).place(relwidth=1,relheight=1)
89 edit_text_ip = Entry()
90 edit_text_port = Entry()
91 ip_label = Label(input_root, text="Entrer l'adresse IP:")
92 port_label = Label(input_root, text="Entrez le port:")
93 connect_btn = Button(input_root, text="Connexion", command=set_ip, bg='#668cff', fg="white")
94
95 # affichage des éléments:
96 ip_label.pack(fill=X, side=TOP)
97 edit_text_ip.pack(fill=X, side=TOP)
98 port_label.pack(fill=X, side=TOP)
99 edit_text_port.pack(fill=X, side=TOP)
100 connect_btn.pack(fill=X, side=BOTTOM)
101
102 input_root.title(name)
103 input_root.geometry("400x500")
104 input_root.resizable(width=False, height=False)
105
106 input_root.mainloop()
107 #envoi des détails-----
108 conn.send(str.encode(name))
109 name1=conn.recv(1024).decode()
110 #Envoi de la cle publique
111 conn.send(msg)
112 #Reception de la cle publique du client
113 rmsg=conn.recv(1024)
114 pkey=pickle.loads(rmsg)
115 print("la clé publique de l'autre est :",pkey[0])
116
117 # 2: Interface graphique de la racine principale
118 root = Tk()
119 bgimage2 = PhotoImage(file="images.png")
120 Label(root,image=bgimage2).place(relwidth=1,relheight=1)
121 # barre de défilement:
122 scrollbar = Scrollbar(root)
123 scrollbar.pack(side=RIGHT, fill=Y)
124 listbox = Listbox(root, yscrollcommand=scrollbar.set)
125 listbox.pack(fill=BOTH, side=TOP)
126 scrollbar.config(command=listbox.yview)
127
128 button = Button(root, text="Envoyer le message", command=send, bg='#a33429', fg="white")
129 edit_text = Entry(root)
130
131 button.pack(fill=X, side=BOTTOM)
132 edit_text.pack(fill=X, side=BOTTOM)
133
134 root.title(name)
135 root.geometry("400x500")
136 root.resizable(width=True, height=True)
137
138 threading.Thread(target=recv).start()
139
140 root.mainloop()
141

```

Programme Python pour implémenter le côté client de la salle de discussion

client.py avec explication en commentaires :

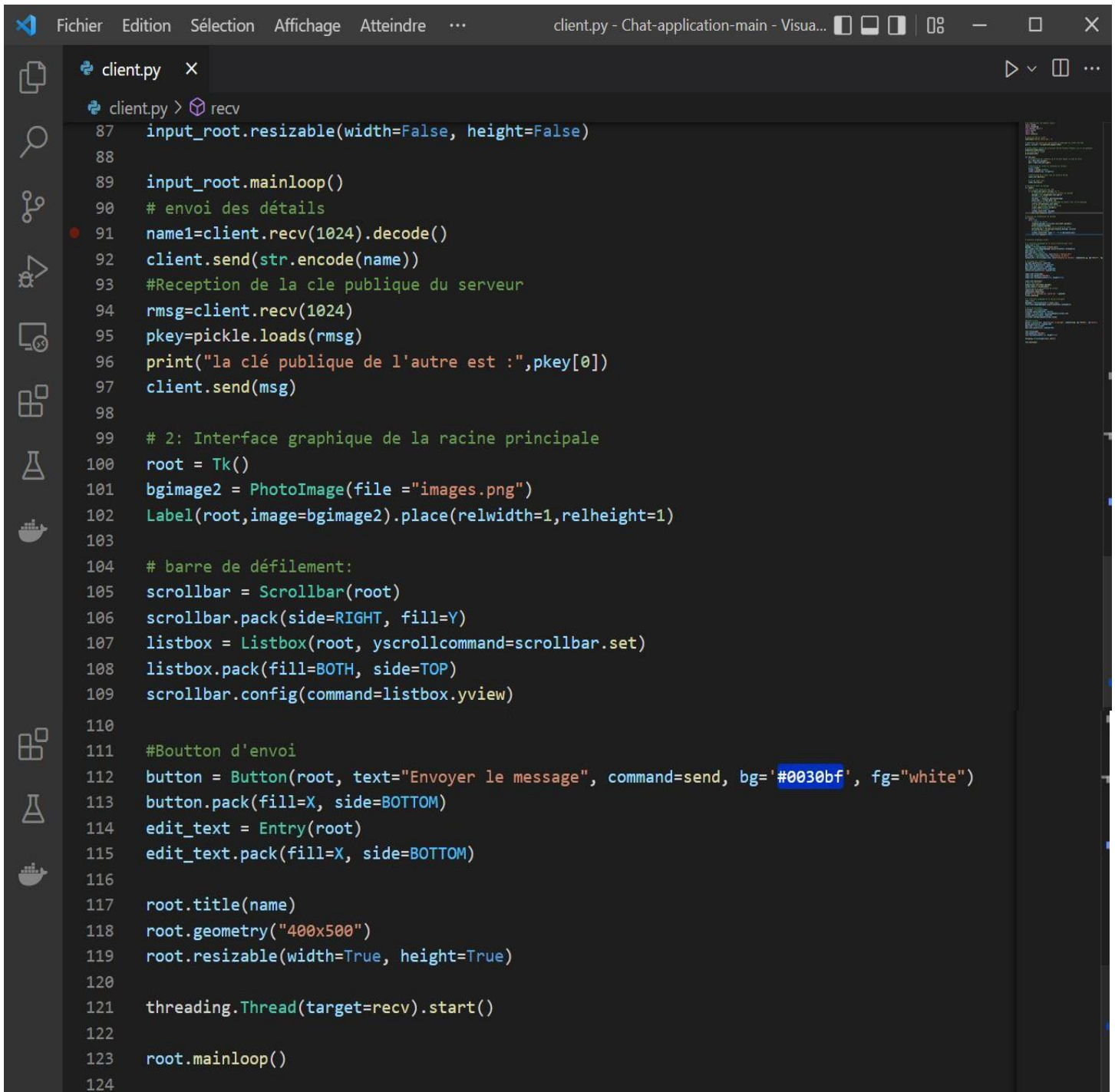


```
1  # On importe tous les modules requis
2  import socket
3  import threading
4  from tkinter import *
5  import pickle
6  import rsa
7  import binascii
8
9  # Saisie du nom du client
10 name=input("Entrez votre nom : ")
11
12 # Generation des paires de cles privees et publiques du client avec RSA
13 public, private = rsa.generate_keypair(1024)
14
15 # pickle.dumps() permet de sérialiser une hiérarchie d'objets, ici la cle publique
16 msg=pickle.dumps(public)
17 # cle publique du client
18 print(public[0])
19
20 def set_ip():
21     # Recuperation de l'adresse ip et du port depuis la zone de texte
22     ip = edit_text_ip.get()
23     port = edit_text_port.get()
24
25     # Definition du client et connexion au serveur:
26     global client
27     client = socket.socket()
28     client.connect((ip, int(port)))
29
```

```

30     # Destruction de l'input root ou racine d'entree
31     input_root.destroy()
32
33     # fin de input root:
34     input_root.quit()
35
36 # fonction d'envoi de message
37 def send():
38     # si la zone texte est non vide
39     if str(edit_text.get()).strip() != "":
40         # recuperation et conversion en octets du message
41         message = str.encode(edit_text.get())
42         #conversion en numb
43         hex_data = binascii.hexlify(message)
44         plain_text = int(hex_data, 16)
45         # chiffrement du texte brut(message en entier) avec la cle publique
46         ctt=rsa.encrypt(plain_text,pkey)
47         # envoi du message chiffré au server
48         client.send(str(ctt).encode())
49         # barre de défilement:
50         listbox.insert(END, message)
51         edit_text.delete(0, END)
52
53
54
55
56
57
58     print(response_message)
59     # decryptage du msg avec la clé privée du client
60     decrypted_msg = rsa.decrypt(response_message, private)
61     # barre de défilement:
62     listbox.insert(END, name1 + " : " + str(decrypted_msg))
63     edit_text.delete(0, END)
64
65
66 # Interface graphique client
67
68 # 1: Interface graphique de la racine d'entrée(input root)
69 input_root = Tk()
70 bgimage = PhotoImage(file ="images.png")
71 Label(input_root,image=bgimage).place(relwidth=1,relheight=1)
72 edit_text_ip = Entry()
73 edit_text_port = Entry()
74 ip_label = Label(input_root, text="Entrez l'adresse IP")
75 port_label = Label(input_root, text="Entrez le port")
76 connect_btn = Button(input_root, text="Connexion au serveur", command=set_ip, bg='#668cff',
77
78 # affichage des éléments:
79 ip_label.pack(fill=X, side=TOP)
80 edit_text_ip.pack(fill=X, side=TOP)
81 port_label.pack(fill=X, side=TOP)
82 edit_text_port.pack(fill=X, side=TOP)
83 connect_btn.pack(fill=X, side=BOTTOM)
84
85 input_root.title(name)
86 input_root.geometry("400x500")

```

```
client.py
87 input_root.resizable(width=False, height=False)
88
89 input_root.mainloop()
90 # envoi des détails
91 name1=client.recv(1024).decode()
92 client.send(str.encode(name))
93 #Reception de la cle publique du serveur
94 rmsg=client.recv(1024)
95 pkey=pickle.loads(rmsg)
96 print("la clé publique de l'autre est :",pkey[0])
97 client.send(msg)
98
99 # 2: Interface graphique de la racine principale
100 root = Tk()
101 bgimage2 = PhotoImage(file ="images.png")
102 Label(root,image=bgimage2).place(relwidth=1,relheight=1)
103
104 # barre de défilement:
105 scrollbar = Scrollbar(root)
106 scrollbar.pack(side=RIGHT, fill=Y)
107 listbox = Listbox(root, yscrollcommand=scrollbar.set)
108 listbox.pack(fill=BOTH, side=TOP)
109 scrollbar.config(command=listbox.yview)
110
111 #Boutton d'envoi
112 button = Button(root, text="Envoyer le message", command=send, bg='#0030bf', fg="white")
113 button.pack(fill=X, side=BOTTOM)
114 edit_text = Entry(root)
115 edit_text.pack(fill=X, side=BOTTOM)
116
117 root.title(name)
118 root.geometry("400x500")
119 root.resizable(width=True, height=True)
120
121 threading.Thread(target=recv).start()
122
123 root.mainloop()
124
```

Démonstration

Ayant eu de nombreux problèmes avec Ubuntu sur VirtualBox, nous avons utilisé la machine locale pour notre démonstration, et donc le serveur et le client auront la même adresse IP.

On commence par exécuter le code depuis le terminal

D'abord celui de server.py

```
Affichage  Atteindre  ...  client.py - Chat-application-main - Visual...  [ ] [ ] [ ] [ ]

PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\asus\Documents\GINF2_2021_2022\Chat_securise\Chat-application-main> python3 server.py
```

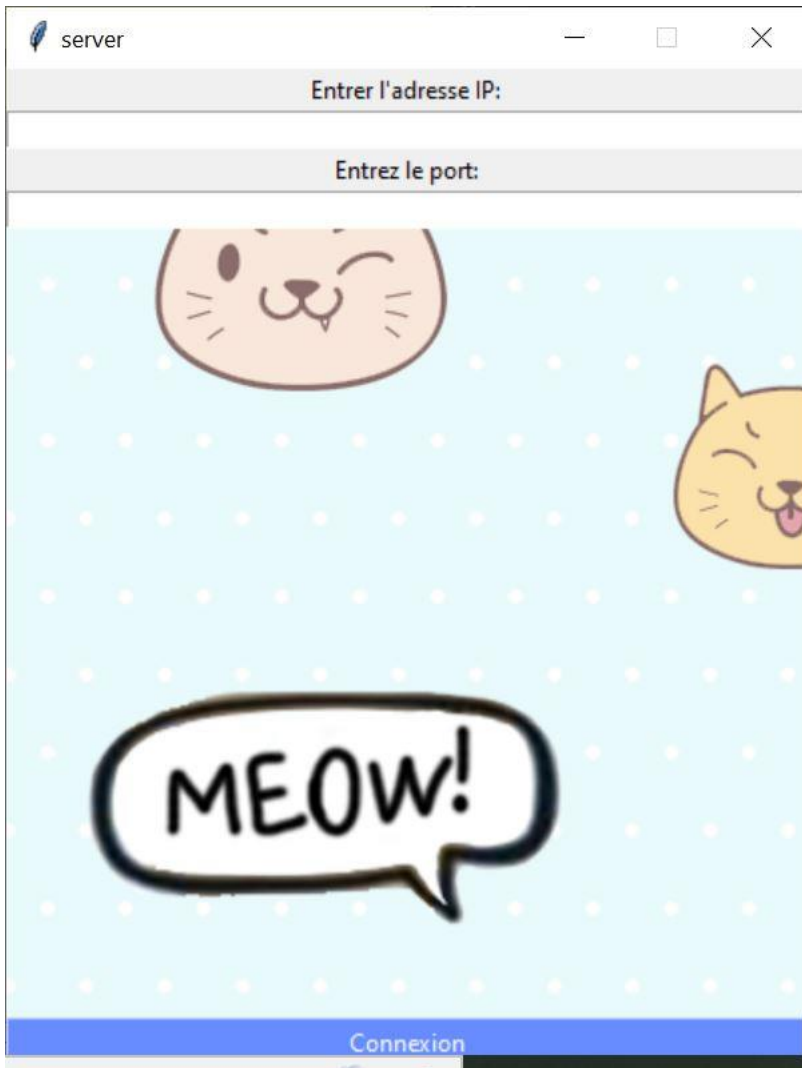
Ensuite on saisit son nom

```
enter your name : server
```

La clé publique du server est affichée sur la console

```
enter your name : server
188083288224787961535214940529958507413710570573079389913400338737020183
956941905411768845881373284209968044546430645861719816052328253556158777
104158066552529429145058389192058801528522198140781562236454541618669495
311750024363014738354075722279288031785514532435458557622655230163615729
891698337463562195909323655738705358012271176460035726205754416847648499
106627161684850586529339608882545712839318919324297003021829208626308850
482350488399687026009379627424161335457597374616284050088271663335596942
354979481018798234817264403462903493234587035078445254962044536465456789
1662247382771345545768388838962268037733
```

Et la fenêtre de chat s'ouvre également



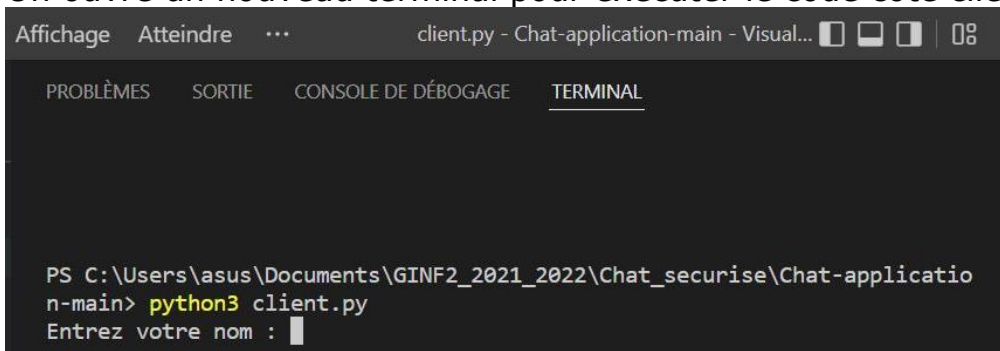
On entre ensuite l'adresse IP et le port du server



The screenshot shows a web browser window titled 'server'. It contains two input fields: 'Entrez l'adresse IP:' with the value '192.168.11.112' and 'Entrez le port:' with the value '5000'. Below these fields is a large, light blue area with a pattern of small white dots. In this area, there are two cartoon cat faces: one orange cat with a winking eye and one yellow cat with its tongue out. A large speech bubble with the word 'MEOW!' is centered in the area. At the bottom of the window is a blue bar with the text 'Connexion'.

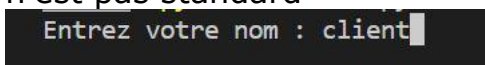
Puis on clique sur Connexion, le server est en attente écoute d'une connexion cliente

On ouvre un nouveau terminal pour executer le code cote client(client.py)



The screenshot shows a terminal window titled 'client.py - Chat-application-main - Visual...'. It has tabs for 'PROBLÈMES', 'SORTIE', 'CONSOLE DE DÉBOGAGE', and 'TERMINAL'. The terminal output shows the command 'python3 client.py' being executed, followed by the prompt 'Entrez votre nom : ' with a cursor.

On saisit le nom du client par exemple mhao, etc... ici nous mettons client mais ce n'est pas standard

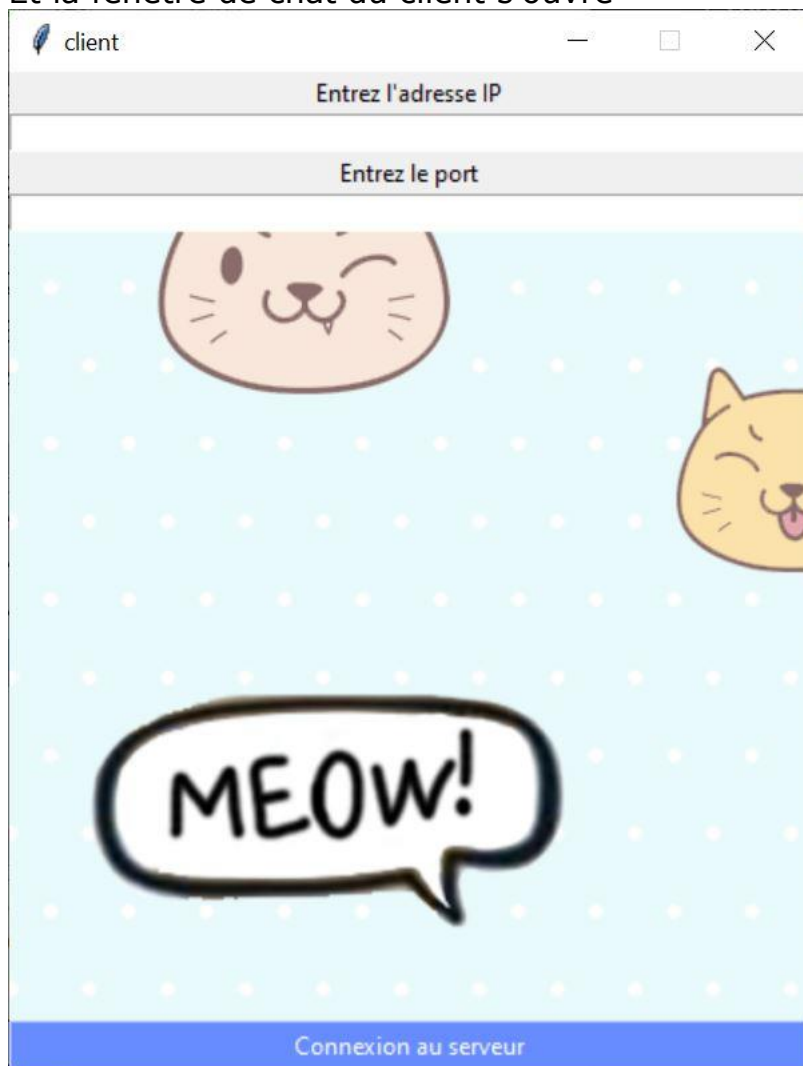


The screenshot shows the terminal prompt 'Entrez votre nom : ' with the text 'client' entered and a cursor at the end.

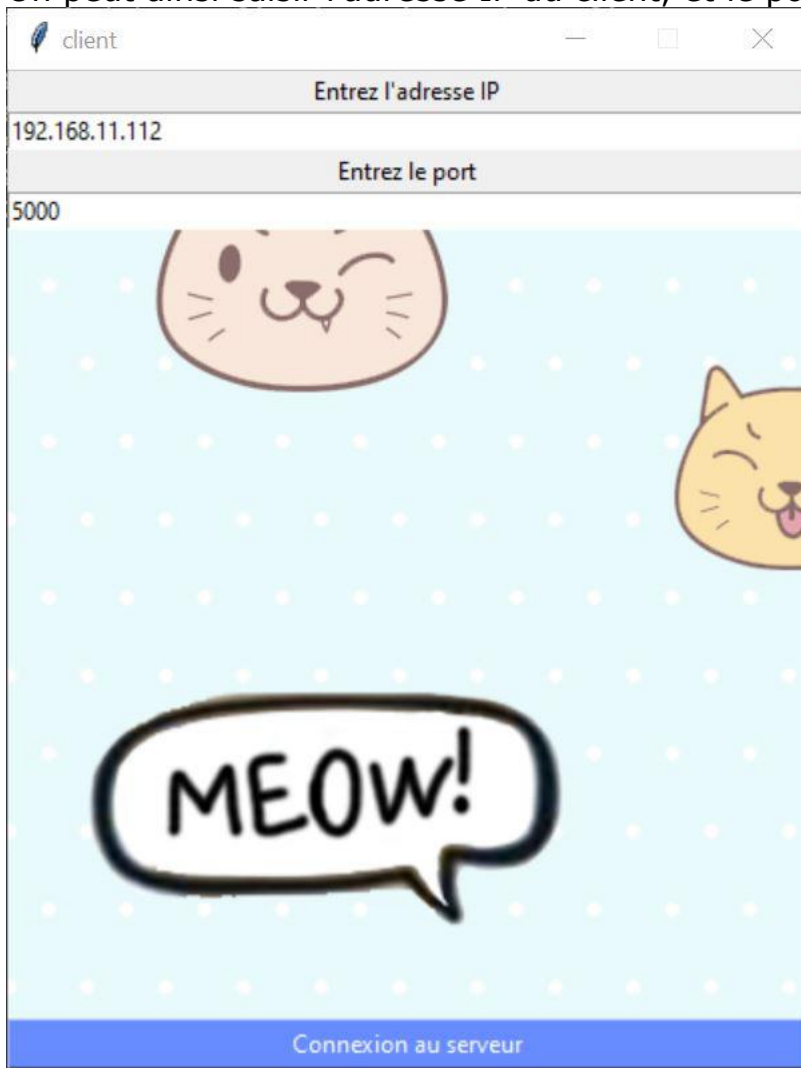
La cle publique du client s'affiche sur la console

```
820458694625502806545754810688772331253726216927409718030613001660975723
760471359697308467499446425157932011704254107783454721570446898440368889
074619932702362404764851180260261411814657505925774493216048851812148934
138726023540476823630117783555594979744940911684059835093338934835994963
806352854371641176195632162019070386644701022575857213962286467078651177
209802887930359729613631887087465149218785721583531056533902581317863261
197703701352483566327858363563632088386212090430857890094755199330354302
635018494406602049945421608074623308322546300590273630856705344849254143
5355188776172410340982079950171257005617
```

Et la fenetre de chat du client s'ouvre



On peut ainsi saisir l'adresse IP du client, et le port qui doit correspondre a celui du server



client

Entrez l'adresse IP

192.168.11.112

Entrez le port

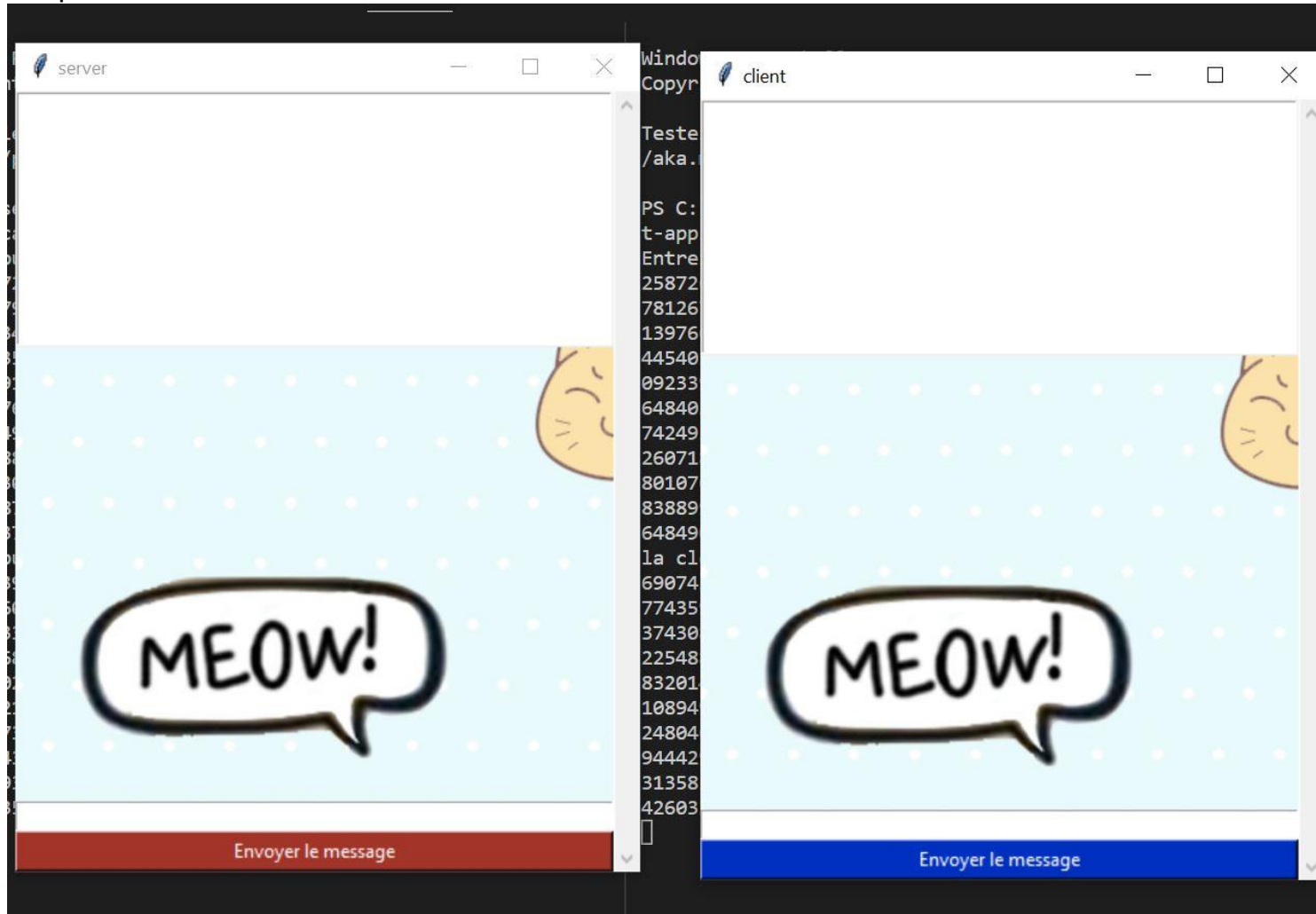
5000

MEOW!

Connexion au serveur

Puis on peut cliquer sur connexion au serveur

On peut maintenant debuter la discussion



Sur la console, on voit les cle publiques qui ont été envoyé et bien recus :

Cote serveur, la cle du client est affichée

```
la clé publique de l'autre est : 820458694625502806545754810688772331253726216927409718030613001660975723760471359697308467
499446425157932011704254107783454721570446898440368889074619932702362404764851180260261411814657505925774493216048851812148
934138726023540476823630117783555594979744940911684059835093338934835994963806352854371641176195632162019070386644701022575
857213962286467078651177209802887930359729613631887087465149218785721583531056533902581317863261197703701352483566327858363
563632088386212090430857890094755199330354302635018494406602049945421608074623308322546300590273630856705344849254143535518
8776172410340982079950171257005617
```

Cote client, la cle du serveur est affichée

```
la clé publique de l'autre est : 188083288224787961535214940529958507413710570573079389913400338737020183956941905411768845
881373284209968044546430645861719816052328253556158777104158066552529429145058389192058801528522198140781562236454541618669
495311750024363014738354075722279288031785514532435458557622655230163615729891698337463562195909323655738705358012271176460
035726205754416847648499106627161684850586529339608882545712839318919324297003021829208626308850482350488399687026009379627
424161335457597374616284050088271663335596942354979481018798234817264403462903493234587035078445254962044536465456789166224
738277134554576838838962268037733
```

Le serveur envoie un message « hello ! » au client :

On peut voir le message non déchiffré dans le terminal, et celui déchiffré avec la clé

privée du client dans la fenêtre de chat qui s'affiche normalement

```
122314078602828111231840906080585889289847642140120384378094
327432739980640838796307569630691734993174315971903076764248
542010736641054059624588587273311083040620515910096543062598
357894459890833438523310502619727846672536074153209343781874
477146130286442923699150719160619485117773795712286519691638
847011739634301383167760319886770104707560628589287069516861
712112010973062668476150876281706916888559496263596356045515
947771688160350797108706465187713607082728228008805398006128
603750313372388740785876692646509380895444051292221209371235
562950592086449232924101345729948487917114884183329590020511
13920476293429917
```

