

CryptoMessenger



Projet de spécialité ISN pour le baccalauréat 2018

Par Julien MARZAL-HILAIRE, Paul NADAL, et Donatien PHILIPPO T°SL et T°S4

[I. Présentation du projet 3](#_Toc516071028)

[1. CryptoMessenger 3](#_Toc516071029)

[2. Origine de l’idée 3](#_Toc516071030)

[II. Démarche de projet 3](#_Toc516071031)

[1. Analyse du besoin 3](#_Toc516071032)

[2. Cahier des charges, contraintes et moyens. 4](#_Toc516071033)

[3. Répartition des tâches 5](#_Toc516071034)

[III. Présentation de l’interface Tkinter 6](#_Toc516071035)

[IV. Collaboration et difficultés rencontrées 8](#_Toc516071036)

[Mot de fin 9](#_Toc516071037)

1. Présentation du projet
2. CryptoMessenger

Le projet CryptoMessenger est un projet de réalisation d’un logiciel de communication entre plusieurs ordinateurs avec comme particularité le fait de proposer la possibilité de crypter les messages avec des méthodes de cryptage simples à utiliser tout en étant compliquées à cracker, sous peine d’être inutile.

1. Origine de l’idée

L’idée du projet CryptoMessenger n’est pas arrivée du premier coup. Il nous a fallu un certain temps avant de choisir, puisque notre passion commune pour les jeux vidéo nous conduisait à penser que réaliser un jeu pourrait être une bonne idée. Cependant, beaucoup d’autres groupes avaient déjà choisi de faire des jeux, et bien qu’ils fussent tous différents les uns des autres et qu’il n’aurait pas été difficile de trouver un autre type de jeu, nous nous sommes dit qu’il serait préférable de trouver quelque chose qui pourrait s’avérer utile. C’est alors que Paul, qui adore la sécurité informatique et le cryptage, nous a proposé de faire quelque chose en rapport avec le cryptage. Après avoir décidé que nous partirions sur un logiciel de cryptage de texte, nous avons cependant jugé que ce n’était pas suffisant, et donc nous avons tourné cela en un chat de messages cryptés.

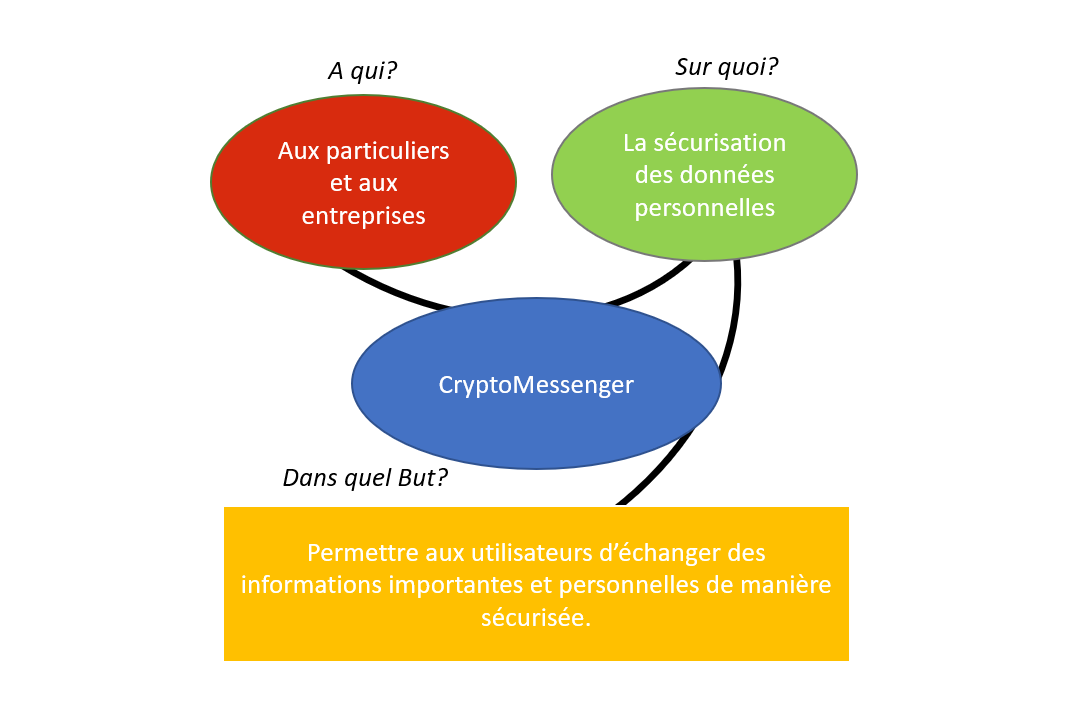
1. Démarche de projet
2. Analyse du besoin

La protection des données et de la confidentialité représente un enjeu important de la communication dans la société actuelle. La communication tenant elle-même une place prépondérante, le cryptage est nécessaire, notamment avec l’essor d’internet, des ordinateurs et des programmes en tout genre, certains servant de logiciel espion pour des sociétés peu scrupuleuses qui vendent ensuite ces informations confidentielles à des publicitaires ou tout autre fournisseur de service nécessitant des informations sur les utilisateurs.

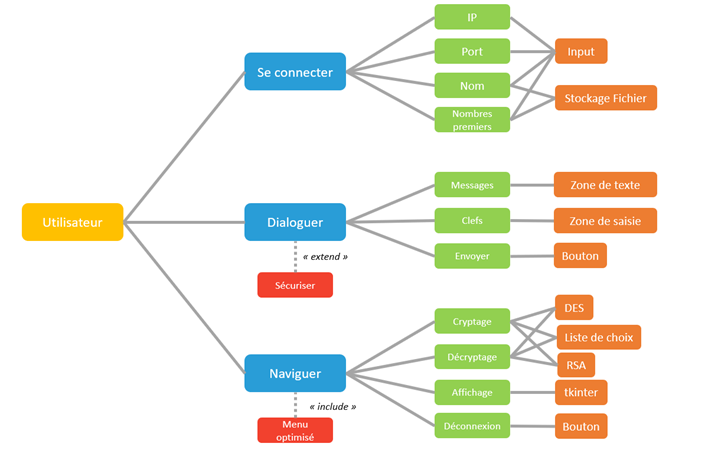
Pour la sécurisation des données conservées sur l’ordinateur, il existe déjà des solutions comme les antivirus qui repèrent les logiciels malveillants.

Cependant, il reste un problème majeur, celui du transport des données, c’est-à-dire qu’il est simple de récupérer des données pendant leur transit entre les deux communicants. Notre programme répond donc à ce besoin de confidentialité en proposant à l’utilisateur de crypter lui-même ses données avant même qu’elles ne soient envoyées. Elles seront ainsi plus compliquées à récupérer.

1. Cahier des charges, contraintes et moyens.

A partir de la définition du besoin, nous avons créé le diagramme « bête à cornes » suivant :

Pour notre projet, nous nous sommes donnés pour contrainte d’utiliser le python comme langage de programmation, de faire une interface graphique assez garnie pour rendre l’utilisation de notre logiciel agréable, et de permettre à 2 personnes ou plus de communiquer.

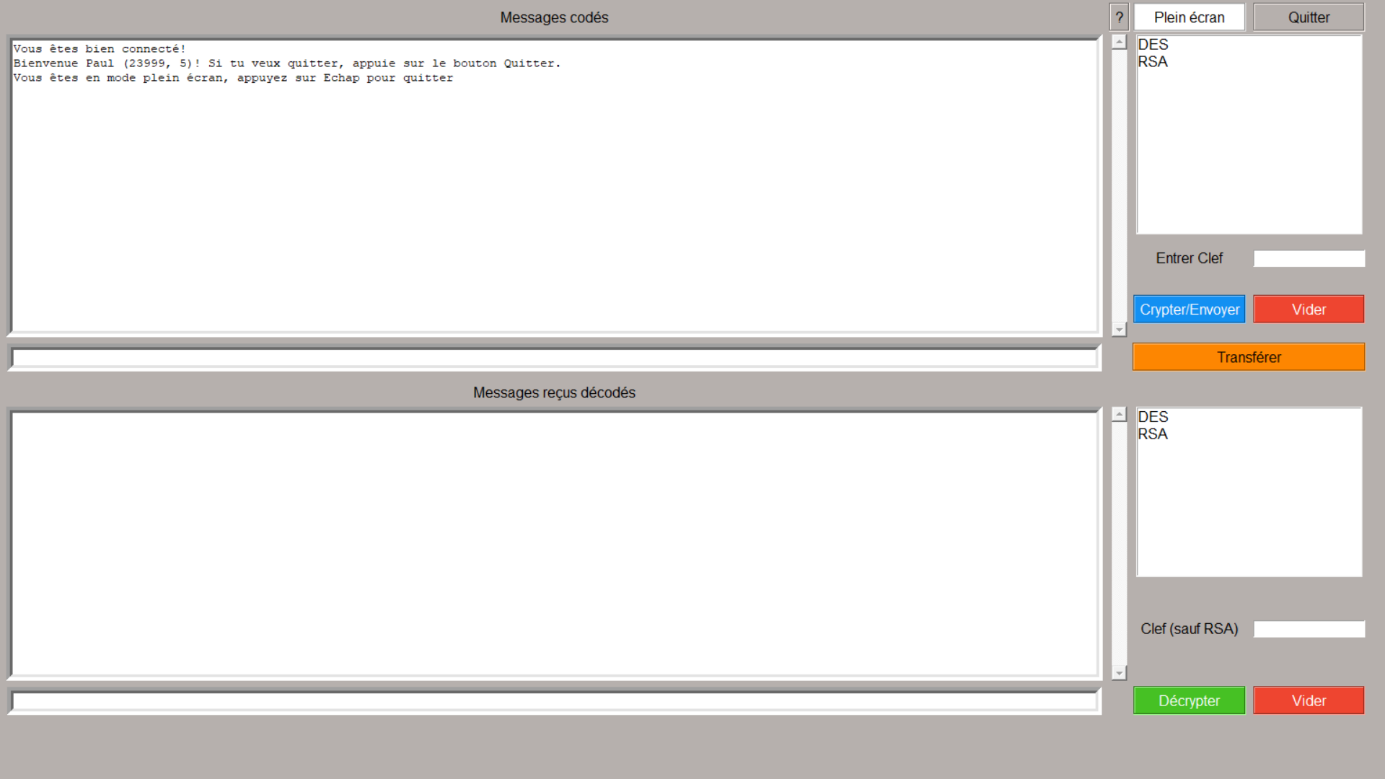
Nous avons choisi de nous servir de Tkinter pour faire l’interface graphique car c’est un module très utilisé donc pour lequel il est simple de se documenter en cas de souci, et de créer un module contenant nos méthodes de cryptage, par souci de lisibilité du code. Pour la communication entre les utilisateurs, nous pensions faire un client qui fasse aussi office de serveur mais nous sommes finalement partis sur un serveur séparé du client. Les utilisateurs n’ont qu’à se mettre d’accord sur qui héberge le serveur et se connectent à lui grâce à son adresse IP. Pour le stockage des données à utiliser sur le long terme comme les clefs de cryptage du RSA et le nom de l’utilisateur nous nous sommes penchés sur le module Pickle qui permet de stocker dans un fichier les objets en tant qu’objets et non en tant que texte. Nous avons ainsi pu faire le diagramme utilisateur ci-dessous.

1. Répartition des tâches

Nous avons commencé dès la première idée, c’est-à-dire que nous avons directement cherché chacun de notre côté une méthode de cryptage que nous pourrions implémenter dans notre programme. Cependant, nous nous sommes rendu compte que celui que j’avais choisi était trop restrictif et mon travail est devenu l’interface Tkinter. Le travail sur le client et le serveur est donc un travail collectif, car il s’agit là d’un code compliqué à comprendre, et il nous a été nécessaire de s’y mettre à plusieurs pour en saisir l’essence.

1. Présentation de l’interface Tkinter

Notre programme est donc composé, pour l’utilisateur, d’un fichier python de serveur nommé "Serveur.py" et un fichier python de client nommé "TkinterV2". Lorsque vous êtes celui qui doit lancer le serveur, lancez celui-ci, il n’y a rien à faire, si ce n’est noter pour vous-même que de base dans le programme le port du serveur est le port 33000.

Lors du lancement du client, l’interface Tkinter ne se lance pas directement. Tout d’abord, un input vous demandera l’adresse IP du serveur auquel le client doit se connecter (localhost si l’utilisateur héberge le serveur sur sa machine), ainsi que le port (33000 par défaut). S’il il s’agit de votre premier lancement du client, d’autres inputs vous demanderont de rentrer votre nom ainsi que 2 nombres premiers nécessaires au bon fonctionnement de notre programme - si vous voulez les changer, il suffit de supprimer le fichier "cles\_ordi" qui se trouve dans le même répertoire que le fichier python. Après ces inputs, l’interface se lance dès que la connexion au serveur est réussie.

L’interface Tkinter de notre programme le rend très simple à utiliser. La toute première ligne contient les boutons utilitaires, comme le « ? » qui ouvre la page d’aide (qui affiche, sous forme de questions-réponses, comment se servir de notre programme), le bouton de passage en mode plein écran et le bouton qui permet de quitter. Lorsque le mode plein écran est activé, il suffit d’appuyer sur la touche Echap pour en sortir.

En dessous de la première ligne se trouve la partie essentielle de notre programme. La partie supérieure est consacrée au chat avec les autres personnes et donc au cryptage des massages, alors que la partie inférieure est locale (donc personne ne sait ce qu’il s’y passe), et donc consacrée au décryptage des messages cryptés envoyés par les autres utilisateurs. Chaque partie est disposée de la même manière que l’autre :

- une grande zone de texte contenant les messages (cryptés ou non) ;

- une zone de saisie située juste en dessous pour écrire les messages à envoyer ou les messages cryptés envoyés par les autres utilisateurs ;

- sur la droite, une liste des différentes méthodes de cryptage utilisables pour crypter les messages. Il faut sélectionner celle que l’on souhaite utiliser pour que le programme comprenne.

- en dessous de la liste, une zone dans laquelle il faut entrer la clef de cryptage correspondant à la méthode de cryptage sélectionnée (8 lettres maximum pour le DES et deux nombres séparés par une virgule pour le RSA) ;

- en dessous de la zone de saisie des clefs, à droite, le bouton « vider » qui comme son nom l’indique vide toutes les zones de saisie de tout texte ;

- à côté du bouton « vider », le bouton qui permet d’envoyer le message écrit dans la grande zone de saisie, en le cryptant avec la méthode de cryptage sélectionnée dans la liste (ou décrypter le message en se servant de la méthode sélectionnée).

La partie cryptage possède un bouton supplémentaire qui permet de transférer le dernier message reçu directement dans la zone de saisie de la partie décryptage (pour faire gagner du temps à l’utilisateur).

Il faut savoir que pour le décryptage, le RSA n’a pas besoin que l’utilisateur rentre une clef, il la possède déjà. La clef à utiliser pour envoyer un message crypté à l’aide du RSA à une personne, il suffit d’utiliser les nombres écrits entre parenthèses juste à côté de son nom.

Appuyer sur Entrée permet d’envoyer un message « en clair », c’est-à-dire non crypté, aux autres utilisateurs, si jamais l’utilisateur qui écrit juge que ce n’est pas nécessaire de crypter ce qu’il envoie.

1. Collaboration et difficultés rencontrées

La coopération a joué un rôle important dans notre projet, nos diverses connaissances du code mises en commun nous ont permis d’avancer sûrement vers notre objectif. Cependant, si nous avons eu recours à la coopération, c’est que de nombreux soucis ont fait surface pendant qu’on développait notre code.

Par exemple, il y a eu un souci, majeur selon moi, qui était que le serveur ne supportait pas le départ d’un des utilisateurs, ce qui est très problématique pour un programme qui est censé être un chat, et que dans un chat les gens peuvent normalement entrer et sortir à leur gré. Nous avons réglé cela en créant une fonction qui envoie un message spécial au serveur lors de sa fermeture. Ainsi, le serveur voit ce message et comprend qu’il doit couper toutes les connections avec le client, ainsi que la fonction de réception pour ce client-là, car c’était cette fonction qui posait le plus de souci (étant donné qu’elle est supposée être appelée de manière permanente).

Un autre problème que nous avons eu dans l’écriture du programme, c’est l’ouverture de la page d’aide. Nous l’avions au départ fait une fenêtre Tk normale, comme une fenêtre principale, avant de découvrir que ça pouvait créer des soucis. Après quelques recherches, nous avons trouvé qu’il existait une classe pour créer des fenêtres secondaires, la classe Toplevel.

Nous avons aussi rencontré un souci lors de la création du bouton « Transférer » et de la fonction qui y est associée. Il fallait réussir à récupérer uniquement le dernier message envoyé. Pour récupérer du texte écrit dans une zone de texte, il faut utiliser la méthode « .get » associée aux Widgets Text (comme la zone de texte justement), et y rentrer comme argument les index de départ de d’arrivée du texte à prendre. Pour cela, nous pensions qu’il fallait trouver à quel numéro d’index correspondait l’indice de fin « end », et lui soustraire 1 pour ensuite le mettre comme indice de départ. Mais cela ne fonctionnait pas, et il m’a fallu moults essais pour enfin trouver le moyen, qui n’était en fait que soustraire 2.

Enfin, nous avons rencontré un problème avec la barre de défilement vertical située à droite de la zone de texte, car nous n’arrivions pas à faire que la dernière ligne de la zone de texte soit toujours affichée. Il suffisait d’utiliser la méthode «.see » associée au Widget Text avec comme argument « end » (l’indice de fin).

Mot de fin

Pour moi, ce projet fut à la fois très intéressant et très enrichissant. Je suis heureux d’avoir pu participer à cette première année de spécialité ISN, je ne me serai sûrement pas autant plu dans une autre spécialité. Je remercie donc mes camarades Paul NADAL et Donatien PHILIPPO pour tout le travail qu’ils ont fait avec moi et pour m’avoir supporté malgré mon caractère, et surtout Vilboub, Nicolas Didrit de son vrai nom, pour nous avoir donné l’occasion de participer à cette spécialité, sans lui nous n’aurions rien pu faire, je lui suis donc très reconnaissant.

ANNEXES

Code du module Encryption.py

Code du fichier Serveur.py

Code du fichier client TkinterV2.py