

RSA

```
#On envoie le message de utilisateurCertifie à receveur
receveur.listeMessageRecu.append((ca,utilisateurCertifie.certificat))
utilisateurCertifie.listeMessageEnvoye.append((ca,utilisateurCertifie.certificat))
#on récupère ce message
message = receveur.listeMessageRecu[len(receveur.listeMessageRecu)-1][1]
return utilisateurCertifie.publicKey == (dechiffre(ca.publicKey[0],message[0],ca.n),dechiffre(ca.publ

def genererCertificat(demandeur,ca):
#certificat = (e_demandeur ** cléPrivé_CA) mod n_CA, (n_demandeur ** cléPrivé_CA) mod n_CA
return (puissance(demandeur.publicKey[0],ca.privateKey, ca.n),puissance(demandeur.publicKey[1],ca.pri

def verifierAuthenticiteMessage(demandeur,ca):
#On récupère le dernier message reçu par le CA
message = ca.listeMessageRecu[len(ca.listeMessageRecu)-1]
```



Cahier des charges

- Implémentation du chiffrement RSA
- chiffrement et déchiffrement asymétrique
- travail en équipe



Techniques

- IDE : VSCODE
- Langage : Python
- Notions abordées : chiffrement asymétrique, mathématiques liées à la cryptographie, POO



Rendu final

- chiffrement fonctionnel
- Délai respecté
- Interface textuelle
- fonctionnalité supplémentaire



Retour d'expérience :

Ce projet m'a permis d'améliorer mes compétences dans le langage python, et d'appliquer concrètement les notions découvertes en cours de cryptographie