Application de RSA Standard et RSA Unidirectionnel

Introduction :

Pour réussir cette phase nous allons utiliser les outils et technologies nécessaires, ci-dessous, à l'implémentation de notre application.

• Python



Figure 12 – Logo Python

Python est un langage de programmation open source interprété côté serveur et non compilé. Créé par Guido van Rossum, il est utilisé pour le développement web, de jeux vidéo et autres logiciels ainsi que pour les interfaces utilisateurs graphiques. Il a notamment été utilisé dans la création d'Instagram, de YouTube et de Spotify, et est l'un des langages de programmation officiels de Google. L'interpréteur Python peut être facilement étendu par de nouvelles fonctions et types de données implémentés en C ou C++ (ou tout autre langage appelable depuis le C). Python est également adapté comme langage d'extension pour personnaliser des applications.

• Django



Figure 13 – Logo de django

Django est un framework python open-source consacré au développement web 2.0. Les concepteurs de Django lui ont attribué le slogan suivant : " Le framework web pour les perfectionnistes sous pression ". Il est donc clairement orienté pour les développeurs ayant comme besoin de produire un projet solide rapidement et sans surprise ... c'est à dire à tous les développeurs !

Comme il est toujours compliqué de partir de rien, Django nous propose une base de projet solide. Django est donc une belle boîte à outils qui aide et oriente le développeur dans la construction de ses projets.

Django a été créer en 2003 et a été publié sous licence BSD en juillet 2005.



• Pycharm

Figure 14 – Logo pycharm

PyCharm est un environnement de développement intégré dédié aux langages Python et Django. Ce logiciel existe en deux versions disponibles en téléchargement, une version libre et gratuite appelée Community Edition et une version complète payante Professional Edition.

nous disposerons d'une foule de fonctionnalités d'édition comme la coloration syntaxique, la complétion automatique du code ainsi que sa factorisation. Le logiciel s'avère utile pour analyser et réparer automatiquement le code source Python et Django.

3.1 Implémentation de RSA Standard

Montrons comment les algorithmes RSA fonctionnent en Python. Le code ci-dessous générera une paire de clés RSA aléatoire, cryptera un message qu'on veut chiffré et le décryptera à sa forme d'origine, en utilisant le schéma de remplissage RSA-OAEP.

Tout d'abord, installons le package pycryptodome, qui est une puissante bibliothèque Python de primitives cryptographiques de bas niveau (hachages, codes MAC, dérivation de clé, chiffrements symétriques et asymétriques, signatures numériques) : (myenv) mamy@mamy-ThinkPad-X240:~/Documents/appliRSA\$ pip install pycryptodome

Figure 15 – Installation du bibliothèque pycryptodome

Importation des packages pour l'algorithme RSA en python



```
from Crypto.Cipher import PKCS1_0AEP
from Crypto.Cipher import PKCS1_v1_5
import binascii
from hashlib import sha512
```

Figure 16 – Importation des packages

3.1.1 Chiffrement/déchiffrement

Maintenant, écrivons le code Python. Tout d'abord, générons les clés RSA (3072 bits) et récuperons la clé publique dans un fichier qu'on nomme **"public.pem"** et la clé privée dans un fichier appelé **"private.pem"** au format PEM





Figure 17 – Génération des clés

Voici la clé publique dans le fichier "public.pem"



Figure 18 – La clé publique

La clé privée est stockée dans le fichier "private.pem"

----BEGIN RSA PRIVATE KEY-----

MIIG4gIBAAKCAYEA54do46Z16k3mupDADaxUQcYwFi4eKPr5W3MGQwD0hy1TMV0P h++QZgND0lEZvQvtIejfUwokNu3y4q59dzwXmzUR6JSlwukMjwYGZjge4ZNF1mm0 nNKgDjPEmdVUOnWATIxONJsQVjAy9Jfrh5/7ee43lqUrLWFg5tahAwdvuVoHXmR2 4DK6blF825xGAYLc6OH+LhBdy7CbAB5+aSjKSRd7MANIK6t4nPZ+bgnqL7pUvApa lyALyZIDCBUwczCFCpLjJ/nCiEmVJoIlg1EWegvora36Is6CkQ0ic5PKrzXWNWYd iusxRVbDtdjo13XlKc5+b8FPRDaX+rFGX4fhIQBse/BN9H6e1KCPdB1wHwpieLeo 8gldgs+i9wcy9/Xq0R1b7yjtxR17I0z/Ama5tnox++3ba7C0hllVNx1GiTQ7EQV8 zM7WhpVt0VXQc6DTYJ/NI7jNeBo6p/kf20S8Vw3cvnTbbxVf2Nr551UK8dwe2t2+ z/RWIfwDedrFqzxTAgMBAAECggGADNE9WGD5cf0tfF8rRENQIZk830JFVcQMtIHW wAELbt8hyTLoLoCi9oRjjeVpSELWD8Sddf9MbiTYVErc098ZRXSq9zyHb3wvjEye lqxKFV3UUqB+dbIS8N/vy3GSaW9I0rQJV+mZSwdj3F0ur+iGgPag0C+roL/QoAB6 YGym3dSvx/HfqapVRNDKpm1T0mox6MvEb3Ax6Roi5cV9mDyNJxe4/TlcfPWcrNlx uicrwxTZjQf06EMUM/mbCyibionBSl9bi7KqaNz4rH393gxGLJ1uh++Pu2V5e80s UeAjMdBqieJIY8K0IPSXCqK5tV4LP0uS+6w7NZIZpFasKXJUA0phVuSVWYQpSFNy fdi+eyaV/Ns2BDA3sW6fdMaxAMiCoLiMsK+kUSHBY3o6gYBRUh1r/3zVlUSLLakI nUeOSCOQqmnGiF7fuOxWrjlpqB1E+MY6yETvDkPiruUaQZ1YpHd/IrgmDYRK7E1K cgHupUB+hr3r3tp4xWs8k8M+OTddAoHBAOvn0j5DHPpoKvaYNAGqEuAxPn+UCmNA 4ZdyWdEW/SY0EP5TQVJG9ztV6t3JlgUooZ03xq67HGaPqXiBT7LbRYFLK1uTN17K or850LzEngmbAeRVDPJ9zC6mblaeiv1mHtJNRodmKHZDruC/tqb4uF3GirU03bXW gm4bTqnW39vRfB3Fdt3e5b8LMrN8baTiYWll8c/p6V80B04wdql/JlBfcQYkaALk eq1D6wgNyDcDmM/3YZtaTEZdNZFfdmEPRQKBwQD7QCbYoxY64bj/o44j/NeUQ+41 qKsTK0QmcRTAdR73P/nU/HkqZjZHB5X1QMTjzdwWzS7e5DJqArsWe6dT+wgS5Cdc E8XkWU7EYcFQMThgHIXIeuCDKoNju4U04raCSJnGfj83hDYzCa+NvzDvogyMlr5y lnQuWIQwKVFjIaRRtd2cNg01GwSkc/sMpB0/TpgJeQofgfiuLI5f/LINqWeVFV4/

Figure 19 – La clé Privée

On récupére les nombres (n, e) qui forment la clé publique

Figure 20 – Le nombre n



Figure 21 – Le nombre e

Et les nombres (n, d) qui forment la clé privée.





Figure 23 – Le nombre d

Ensuite, on crypte le message au moyen de la clé publique RSA (ici supposée être disponible localement dans le fichier **"public.pem"**) à l'aide du schéma de cryptage RSA-OAEP (RSA avec remplissage OAEP PKCS1) et le text chiffré est stocké dans le fichier **"encrypt.txt"**.



Figure 24 – Le chiffrement du text en claire

Enfin, on décrypte le message en utilisant RSA-OAEP avec la clé privée RSA qui est stockée dans le fichier **"private.pem"**, le message décrypté se trouve dans le fichier **decrypt.txt**.



Figure 25 – Le déchiffrement du message chiffré

3.1.2 Signature/Vérification

Maintenant, signons un message, en utilisant la clé privée RSA (n, d). Calculons son hachage et augmentons le hachage à la puissance d modulo n (chiffrons le hachage avec la clé privée). Nous utiliserons le hachage SHA-512, il s'adaptera à la taille de clé RSA actuelle (3072). En Python, nous avons l'exponentiation modulaire comme fonction intégrée pow (x, y, n).

La signature numérique obtenue est un entier compris entre zéro et la longueur de la clé RSA [0 ... n] qui est stocké dans le fichier "sign.txt, comme illustré ci-dessous



Figure 26 – La signature du message en claire

Ensuite, vérifions la signature en déchiffrant la signature à l'aide de la clé publique (élevons la signature à e modulo n) et comparons le hachage obtenu de la signature(hastFromSignature) au hachage du message signé à l'origine(hash)



Figure 27 – La vérification de la signature

3.2 Implémentation de RSA Unidirectionnel

3.2.1 Chiffrement/Déchiffrement

Pour l'implémentation de RSA unidirectionnel on crée d'abord deux paires de clés de taille 3072 et 4096. chaque clé est stockée dans un fichier nommé respectivement par "**publicuni1.pem**", "**privateuni1.pem**", "**publicuni2.pem**", et "**privateuni2.pem**". On a donc deux clés publiques et deux clés privées

```
def rsa_uni(request):
    new_key = RSA.generate(3072)
    private_key_uni1 = new_key.exportKey("PEM")
    public_key_uni1 = new_key.publickey().exportKey("PEM")
    private_key_uni1_path = Path('privateuni1.pem')
    private_key_uni1_path.touch(made=00600)
    private_key_uni1_path.write_bytes(private_key_uni1)
    public_key_uni1_path = Path('publicuni1.pem')
    public_key_uni1_path.touch(made=00604)
    public_key_uni1_path.touch(made=00604)
    public_key_uni1_path.write_bytes(public_key_uni1)
    private_key_uni1 = Path('privateuni1.pem')
    public_key_uni1 = Path('privateuni1.pem')
    rsa_public_key_uni1 = RSA.importKey(public_key_uni1.read_bytes())
    rsa_public_key_uni1 = RSA.importKey(private_key_uni1.read_bytes())
    rsa_private_key_uni1 = RSA.importKey(private_key_uni1.read_bytes())
    rsa_private_key_uni1 = PKCS1_0AEP.new(rsa_private_key_uni1)
    print(f"Public key: (n={hex(new_key.n)}, e={hex(new_key.e)}]*)
    print(f"Private key: (n={hex(new_key.n)}, d={hex(new_key.d)})*)
```



Figure 28 – Génération des clés

On récupére les nombres n, e et d pour chaque clé

- Pour les clés de tailles 3072 on a :
- les nombres (n,e) qui forment la clé publique ci-dessous



Figure 29 – Le nombre n

Figure 30 – Le nombre e

La clé publique

----BEGIN PUBLIC KEY-----

MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAnLIUSKE3uACborOQxet9 wIXqzuaLncHNMxcbx2cxuDhIfArM4YMjpmIYoHcFn3dkJ72NC4tLufLQjyjea21j 6ukw4RByBM06PU95tayQvWhWT2w4NTPLZLxftf9pGwJWmQ+7nnma2Gn15iAlzwBe x3BDBtz/77QnzhdJ6dCuViKyVNR4xca3rT4H2EIGl6s0IydQLA8HBm+bfmjDHpaQ 1NRYCR8ot7MJkWMsp1lI5CDXnHzgJdgB+xDV3qjFs7DZ5UgSD2SX4gmI/7ECnM34 9+rKgk0/FVwmAu6/mcyAx80nMKZt9iV+0FVrQeaclXJ3fi5MNXI5YmHdtkL6SNKQ HBGKJvqWFymbGPAVeVwt3TVQvP5IzxwcjM0rNDmVHI0jY/b9Ir2kk/rAeuOWy0F9 ldpeJqp02Vb3xlEX5TF2CEMPohNXz8ciGh3a2y65Gx1Njkdv3Rr9N33hgYSrZw7k BePj3FXisU/wgt5Rn3nqZiXGoGtW6HnWkhz4qlqWTHRBAgMBAAE= ----END PUBLIC KEY----

Figure 31 – La clé publique 1

- Les nombres (n,d) qui forment la clé privée ci-dessous



Figure 33 – Le nombre d

bb82fb888c0d54498303fcfb48d96b09a38368c7fc65e2418384ee43a2&6de 97b6db56c7c662dba7c74ca665567b8d55fab735aff94f18af36a78arefc91f

La clé privée

----BEGIN RSA PRIVATE KEY-----

MIIG4wIBAAKCAYEAnLIUSkE3uACbor0Qxet9wIXqzuaLncHNMxcbx2cxuDhIfArM 4YMjpmIYoHcFn3dkJ72NC4tLufLQjyjea21j6ukw4RByBM06PU95tayQvWhWT2w4 NTPLZLxftf9pGwJWmQ+7nnma2Gn15iAlzwBex3BDBtz/77QnzhdJ6dCuViKyVNR4 xca3rT4H2EIGl6s0IydQLA8HBm+bfmjDHpaQ1NRYCR8ot7MJkWMsp1lI5CDXnHzg JdgB+xDV3qjFs7DZ5UgSD2SX4gmI/7ECnM349+rKgk0/FVwmAu6/mcyAx80nMKZt 9iV+OFVrQeaclXJ3fi5MNXI5YmHdtkLGSNKQHBGKJvqWFymbGPAVeVwt3TVQvP5I zxwcjM0rNDmVHI0jY/b9Ir2kk/rAeu0Wy0F9ldpeJqp02Vb3xlEX5TF2CEMPohNX z8ciGh3a2y65Gx1Njkdv3Rr9N33hgYSrZw7kBePj3FXisU/wgt5Rn3nqZiXGoGtW 6HnWkhz4qlqWTHRBAgMBAAECggGADK3/ix99gFvSoqo58nB8tVhUCKukBWWdUtKk rzbgHkwccXORe94kw5Eq9P35bBufjlrkT77VDeL8K9l4X7EMzAGHfW7bPk91G2j6 LFCE4ghpJsKKRgPX6mFKTWvkxKHxZ/y0uuNhCFbgswh+7SIJvJHBzfH5ALVo7MbH Q5UpxlNgb6tFBDWkWO+s3gimh+qVExMnUnYucPLiRKrHJaTn0cRGq2/kcTKa35m3 qOl5ehxX1i3n5XTcepIa/Fcj4fqGRuR90Pe8f3XZn+6UjX50FFfrSlPE3qX0EEj6 qisPAjed1uJePjfkaru3Pzynob8fTQXC+6nwuuZbdi+MM9s2nJHJEas1Johls8lu BPbTy6ECEyfptcPLy0Zp3z8sz1mpjzouB3900PRyLFxILavm7W4vy7yLUNE9Pgy6 RWWAmuYg7xw8tiElzZGmMrE+RQlt9KDh7/9v24DI/85tla9Z0F47Rb/gV+vYezbY 5D9TmhYp8ggSrh790QgmEdk5BptRAoHBAMK6mmi2fYIuIVPnZ+F6qLfR05RZEK2F tQDkkQo6HJgTgnpOgkxrDOe2+gySe5a4XGX+WS2QOi+9+VA7Co8qR84Vl/WUr351 79wLwiNnSU+kY8r4zNvw2sXpwebnPX4TTthfaSbITWJ50D1nQlxKhyApl49UstlI ZAOTybKCOn0iwYnZXBEJGxESHlKf3vMDmzLzFUs7rcLCHV4PBgk132nQ00FRELho DkgSMHezJCjvaOV/7Hv2Cmp/wTNSM4U42wKBwQDN/+XCI9FY/Es9QJowAdSVIx1c VTqgobtcón0JGOs/+Pl5brplgLJz8ro1aR1Tg/OA/AHyIepEGd9j/h4xF876DDPU 50VQuFSR2u5tPbqIn3DlVf/7Umrz0zZcmj8hCl9zWjr3L0MWXR5jLKJqdjll4WoB +z/8DW9gyS8zbUcIlKGQ4/Nd1jSqsH4v6uj957wa5Kl2gL3M+887ZFEE3WfzAHo/

Figure 34 – La clé privée 1

- Pour les clés de tailles 4092 on a :
- les nombres (n,e) qui forment la clé publique ci-dessous





Figure 36 – Le nombre e

La clé publique

-----BEGIN PUBLIC KEY-----MIICIJANBgkqhkiG9w0BAQEFAADCAg8AMIICCgKCAgEA3vzuJd4WtbfUQJwiD24D gxCkTkccDml0riX/V1y8AAHABK5/XEPxhse6ySbpEezAx8n1MeGEwTzRE4YTT+UH +60xgP7Y0QNft5CFHUzo9VFvX94VpX7IRVKM7l+dzM6ypbzI2YNdmqFd8pI70Uc4 jlnkIprr5gblhk0gDLCwatZwWt9pN3LUkP45D5rivGZUqLtHVg5nJErz/kNgLdJP 8iPf9QPnnsvFLi84u14pXRDcvREQ3FFKHTTmeGo4rhYyslWG4xSlAVq8LJPhI/y9 DF8eS2eVcrCTJzk8+tYY8112fqhZK8Yvo34NgwkQ+hlo6n+FdXCnXue834NN8y3 dSpXqcFBU5d73YD6FeaZ7yugylUrtSC1VuvZg/0nGLH+9+kodTQJ5Kr3U+l62mCr P08J5aJ6pAWUKHGlNtqUGYAusx05dds1xCm1KORSVQcziH2uAHiUzn83vUtXv7A1 F8QHcxaEi2XkkjFcntw7cVmMKQtTipRxj7SYr0gVbsCR9b0s1MmKnW0KdyD7HiLK 0aKqUFxH7oMsVuU5JKmDBsQNfrI11/o02jPqUmd0ZTxA42FtnT8VKRG6LMea8rax Gmdfw57bYoK20kw7GwplcbqUkzh0VmBDXiw3V76eyiz/wPtU5NHwbEUYKnFvQZ0T hsUj8belybJZ0NJacLU/uM0CAwEAAQ== -----END PUBLIC KEY-----

Figure 37 – La clé publique 2

- Les nombres (n,d) qui forment la clé privée ci-dessous



Figure 39 – Le nombre d

La clé privée

BEGIN RSA PRIVATE KEY
MIIJKQIBAAKCAgEA3vzuJd4WtbfUQJwiD24DgxCkTkccDml0riX/V1y8AAHABK5/
XEPxhse6ySbpEezAx8n1MeGEwTzRE4YTT+UH+G0xgP7Y0QNft5CFHUzo9VFvX94V
pX7IRVKM7l+dzM6ypbzI2YNdmqFd8pI70Uc4jlnkIprr5gblhk0gDLCwatZwWt9p
N3LUkP45D5rivGZUqLtHVg5nJErz/kNgLdJP8iPf9QPnnsvFLi84uL4pXRDcvREQ
3FFKHTTmeGo4rhYyslWG4xSlAVq8LJPhI/y9DF8eS2eVcrCTJzk8+tYYy8112fqh
ZK8Yvo34NgwkQ+hlo6n+FdXCnXue834NN8y3dSpXqcFBU5d73YD6FeaZ7yugylUr
tSC1VuvZg/0nGLH+9+kodTQJ5Kr3U+l62mCrP08J5aJ6pAWUKHGlNtqUGYAusx05
dds1xCm1K0RSVQcziH2uAHiUzn83vUtXv7AlF8QHcxaEi2XkkjFcntw7cVmMKQtT
ipRxj7SYr0gVbsCR9b0s1MmKnW0KdyD7HilK0aKqUFxH7oMsVuU5JKmDBsQNfrI1
1/o02jPqUmd0ZTxA42FtnT8VKRG6LMea8raxGmdfw57bYoK20kw7GwplcbqUkzh0
VmBDXiw3V76eyiz/wPtU5NHwbEUYKnFvQZ0ThsUj8belybJZ0NJacLU/uM0CAwEA
AQKCAgBXgM6dSoUD4zLl9b8B6gxzwrSPSSVP3H8elaeZKzlVEVq+G5G77aeaWcdJ
Bz1fvNW7sNtIqzQytkBfVnY+nGQfy1NNCPRljxbhDFPINzWVitLXPmmlK1bAqU/H
fPsJ8zS0iBwTCxNQyjD5LvxN0BMP/WNiVwPqHNu18473m+Bm1GZ7qn759NRIX5KZ
tpByGP/wxKkOMpGz8rvDk8HqyUR5Ay3v0ovySwlCAeLTc37s1+CfStKgej/IrUkh
iKFcYzaat6g6dQ/5M5dzMMahHh07fRoAwhGp030zaSkuT5SkaShtlQX/YhymLkTw
RJx4j3pPGrar79myfd/WsijStEFq++BC0orALGcqooILywGFnmuc0Zd07hwNj/pU
FBqRVlVQ9nPX/+Yu5LHQc7DeTK4hKDFQ7v0g28u73KVEAPHYaTtT+g9i8NLa2xlo
rcROUIouJIkNzH3WHfSxlvlel+FzMwPkA4H53S5Bzqxa0+MWYiN+pRNXJLnqiCWt
NlX+UIrJVjTPBKtULJ0U/tAfcy4ExWbIxXViymW/nGWMgf16H3gq3nZALXtEnR3Q
l0I+sjY+4YDkT4mk+6yga1uKL9JfDv+ombo2sV2+sgyNIxEA2bD4HmlsepBStUMJ
94fwiv/1VMv++SzN9kCbmv9QsX5mL+5cv9a2jQEcsQWMvPr/qwKCAQEA70ttW60o
4D50UGdf+CRCPrrsYXxUD8D+0Vs/oJvssnUkUujfIiBDm6Wm66nk0Z0M0GCjLDr0
jaLgQlCJ6glFUSMHImYc1wKYqDxtnmmVETh6iDNHLpI26GMfjAGBJDsmtqJmySJI

Figure 40 – La clé privée 2

Ensuite on crypte le message en claire avec la première clé publique rsa_public_key_uni1 qui est stockée dans le fichier **publicuni1.pem** et on obtient un message chiffré nommé "encrypted" se trouvant dans le fichier "encryptuni1.txt". Ce dernier sera transformé avec la deuxiéme clé publique **rsa_public_key** qui se trouve dans le fichier **publicuni2.pem** on obtient un second chiffré nommé rsa_encrypted stocké dans le fichier "encryptuni2.txt"



Figure 41 – Le chiffrement des messages

En fin on déchiffre le message "rsa_encrypted" avec la clé privée *rsa_private_key_uni2* on obtient un déchiffré nommé "decrypted" qu'on stocke dans le fichier "decrypt1.txt". On decrypt le message "decrypted" avec la clé privée *rsa_private_key_uni1*, on obtient "rsa_decrypted" se trouvant dans le fichier "decrypt2.txt" qui contient le message en claire.



Figure 42 – Le déchiffrement des messages chiffrés

3.3 Test et Validation

3.3.1 RSA standard

a. Page de gérération des clés

On clic sur le bouton "Générer la paire de clé" et les clés sont générées dans les fichiers **public.pem** et **private.pem** ci-dessus.

← → C ③ 127.0.0.1:8000/cr	yptosystem/	<u></u>	☆	\$ ÷
***••				
 RSA standard RSA Unidirectionel 	RSA Standard			
■ RSA bidirectionel	Génération des clés Chiffrement Dechiffrement Signatu Généreration de la paire de clé Générer la paire de clé	re Verifcatio	n	

Figure 43 – Page de génération des clés

b. Page de chiffrement du message

On veut chiffrer le message stocké dans le fichier "plaintext.txt" ci-dessous avec la clé public générée dans le fichier "public.pem"



 $Figure \ 44 - Le \ fichier \ plaintext.txt$

D'abord on clic sur le bouton "Upload" et on choisit le fichier "plaintext.txt" contenant le text a chiffré. Ensuite on clic sur "Chiffrer" et le message est chiffré et gardé dans le fichier "encrypt.txt" comme illustré ci-dessous.



Figure 45 – Page de chiffrement du message



Figure 46 – Le fichier encrypt.txt

c. Page de déchiffrement du message

On clic sur le bouton "Upload", ensuite on choisit le fichier contenant le chiffré "encrypt.txt" et enfin on clic sur le bouton "Déchiffrer". Le message est déchiffré à l'aide de la clé privée stockée dans "private.pem" et gardée dans "decrypt.txt".



Figure 47 – Page de déchiffrement du message

Ouvrir	+	.Fì		decrytp.txt ~/Bureau/test/memoire_rechearche_rsa
1 þ'bonj	jour	ceci	est un message	secret'
			Figure $48 - Le$	e fichier decrypt.txt

d. Page de signature du message

on veut signer le message en claire, se trouvant dans le fichier "plain-text.txt", avec la clé privée.

on clic sur "Uplaod" on choisit le fichier "plaintext.txt" et on clic sur le bouton "Signer". Le message signé est gardé dans le fichier **sign.txt**.



Figure 49 – Page Signature du message

Ouvrir	•	ſŦ	sign.txt ~/Bureau/test/memoire_rechearche_rsa	Enregistrer	Ξ	-		×
1 223110	00998	86545	5963461318565731274388483992744090569673286745856723	275952760500	40509	09776	7646	87903

Figure 50 - Le fichier sign.txt

e. Page de verification de la signature

Maintenant on va vérifier la signature, on clic le bouton "Uplaod" on choisit le fichier **sign.txt**. Ensuite on clic sur le bouton "Verifier" et on obtient le resultat ci-dessous :



Figure 51 – Page Vérification de la signature

	derr eine sollter arten sonlinge st
ų	Signature valid: True
5	[10/400/0001 07.7/.00] HDDET /0000tooustom/ HTTD/1 18 000 1/017

Figure 52 – Résultat de la signature

La signature est valide et retourne **true** car le message reçu est effectivement celui qui a été signé. Dans le cas contraire le résultat serait **false**

3.3.2 RSA Unidirectionnel

a. Page de gérération des clés

On clic sur le bouton "Générer la paire de clé" deux fois et les clés sont générée dans les fichiers **publicuni1.pem**, **privateuni1.pem**, **publicuni2.pem** et **privateuni2.pem** ci-dessus.

÷	→ C ① 127.0.0.	1:8000/cr	yptosystem/uni	☆	٩	÷
*	8	F				
	RSA standard RSA Unidirectionnel		RSA Unidirectionnel			
	RSA bidirectionnel		Génération des clés Chiffrement Dechiffrement Signature Verifcation Généreration de la paire de clé Générer la paire de clé	E.		

Figure 53 – Page de génération des clés

b. Page de chiffrement du message

On veut chiffrer le message se trouvant dans le fichier "plaintext.txt" avec les deux clés publiques

Maintenant on choisit le fichier "plaintext.txt" en cliquant sur le bouton "Uplaod", on clic sur le bouton "Chiffrer" et le message est chiffré et gardé dans le fichier "encrypt1.txt" ci-dessous avec la première clé public qui se trouve dans le fichier **publicuni1.pem**



Figure 54 – Page de chiffrement du message stocké dans plaintext.txt

Ouvrir	+	Fl			~/Bu	reau/bes	crypt st/mem	tuni1. noire_r	txt echearc	he_rsa			Enreg	istrer					8
1 b'\x05 *FD\xf \x00\x 0\xd6T \x18[\ \x03\x C\xf3\ \x08&^ \xee\x \x08&^ \xee\x \x19\x \x3f\\ \x04\x \x04\x \x9ex\	\xd7: b\xba 15\x1 wA\xc x0e\> aeehc xb3&\ \xd2\ ec\xf h4d\> 9ff\> xde\> xfe\> bd\xt xa8[\	uJh 88\xf 19\xd 16<\x 89\x 5\xcf xac\ xd5\ xd5\ xd5\ xd5\ xd5\ xd5\ xd5\ xd5	F\xdf\x93 d\xf0\x10 b\xd2\x0e (ce\xe0\x (fb\xc6\xal (fb\xc6\xal (fb\xc6\xal (xb7\xf0\x) (xb7\xf0\x (xb7\xf0\x) (xb7\xf0\x (xb7\xf0\x) (xc1\xa8SU)	P\x1fR\x x19\xa1 xb6\xd1 cb8\x965 8\xffP\x 2\xa0x5 x41\x72\ c2\xa0x7 x14\xf2\ c2\xa0x7 x41\x72\ c0\xf6\ c0\xf6\ c0\xf6\ c0\xf6\ x89\xd x81\xa	<pre><c3k0\x 5hru\xb="" _\="" c-="" c<4\x95="" d\xfa="" l\x81\x="" l\xe44\="" td="" x6a="" x<=""><th>xa1\x1 \xfdC\ xe7x\x be\x8f 9\xe4a 0>\xef \x14\x F\x16\ (\x05\]\x95\ xaa?- \xff\x 4\'\x0</th><th>12i\x (xa2) (ce7H) f%- a\xbf f\xa1 (d6Q5 (xb1) (x16a (x92k (x92k (c6\xb</th><th>(1f[% (xed) (x927 52\xd (\x98 5\x0b (xf5) (xf5) (xf5) (xab (xf4) xd (xab</th><th>(xda) xc8 H7 Xa (x18) x14 (x18) x14 (x18) x2 x12 (x18) y x18) y x2 x12 (x18) y x18) x x18) y x18) y x18) x x18) y x x18) y x x18) y x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x x x x x x x x x x x x x x x x x x</th><th>xf6\ (1a\x ac\x9 (xfa x8c* af\x8 (x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x</th><th>xf9\ 9a\x 5\xa \x87 aU\x a\xe xb5q eFlL xb1K 8bnI 7f!-</th><td>xee 83U 00\x (\x1 (\x1 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2</td><th>\r\r\ D\xe3 83\xe1 3\xe1 xc9\3 8b - a\xd3 f\x8l 5\xce</th><th>(x11& 7(x04 c0(x9 F) ? (xaa(x 7(xec oT(xa 208V</th><th>%\x2 \xd@ 3\xf 000\x \xc1 \xc1 \xdc</th><th>0d\x:8 0\x8 FbV} cda\: (\xb xc8 1\x0</th><th>1c\xa 6\xf0 - xe6\x 7^\x9 !;- 7D}-</th><th>a7\xc bKj9K a2\x Df-</th><th>3}- \xde^ c1U\x</th></c3k0\x></pre>	xa1\x1 \xfdC\ xe7x\x be\x8f 9\xe4a 0>\xef \x14\x F\x16\ (\x05\]\x95\ xaa?- \xff\x 4\'\x0	12i\x (xa2) (ce7H) f%- a\xbf f\xa1 (d6Q5 (xb1) (x16a (x92k (x92k (c6\xb	(1f[% (xed) (x927 52\xd (\x98 5\x0b (xf5) (xf5) (xf5) (xab (xf4) xd (xab	(xda) xc8 H7 Xa (x18) x14 (x18) x14 (x18) x2 x12 (x18) y x18) y x2 x12 (x18) y x18) x x18) y x18) y x18) x x18) y x x18) y x x18) y x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x18) x x x x x x x x x x x x x x x x x x x	xf6\ (1a\x ac\x9 (xfa x8c* af\x8 (x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x	xf9\ 9a\x 5\xa \x87 aU\x a\xe xb5q eFlL xb1K 8bnI 7f!-	xee 83U 00\x (\x1 (\x1 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2 (\x2	\r\r\ D\xe3 83\xe1 3\xe1 xc9\3 8b - a\xd3 f\x8l 5\xce	(x11& 7(x04 c0(x9 F) ? (xaa(x 7(xec oT(xa 208V	%\x2 \xd@ 3\xf 000\x \xc1 \xc1 \xdc	0d\x:8 0\x8 FbV} cda\: (\xb xc8 1\x0	1c\xa 6\xf0 - xe6\x 7^\x9 !;- 7D}-	a7\xc bKj9K a2\x Df-	3}- \xde^ c1U\x

Figure 55 - Le fichier encryptuni1.txt

Ensuite on clic sur le bouton "Upload" et on choisit le fichier "encryptuni1.txt", on clic sur "Chiffrer" et le contenu du fichier est chiffré de nouveau avec la deuxième clé public stocké dans "publicuni2.txt" et est gardé dans le fichier "encryptuni2.txt".



Figure 56 – Page de chiffrement du message stocké dans encryptuni1.txt



Figure 57 – Le fichier encryptuni2.txt

c. Page de déchiffrement du message stocké dans "encryptuni2.txt"

On clic sur le bouton "Upload", ensuite on choisit le fichier "encryptuni2.txt" et on clic sur le bouton "Déchiffrer" pour le premier déchiffrement. Le contenu du fichier est déchiffré avec la deuxième clé privée stockée dans "privateuni2.pem" et le chiffré est gardé dans le fichier "decrypt1.txt"



Figure 58 – Page de déchiffrement du message stocké dans encryptuni2.txt



Figure 59 – Le fichier decrypt1.txt

d. Page de déchiffrement du message stocké dans decrypt1.txt

on clic sur "Uplaod", on choisit le fichier "décrypt1.txt et on clic sur le bouton "Déchiffrer". Le contenu du fichier est déchiffré avec la premiere clé(privateuni1.pem) et est gardé dans le fichier **decrypt2.txt**



Figure 60 – Page de déchiffrement du message stocké dans "décrypt1.txt"



Figure 61 – Le fichier decrypt2.txt

Conclusion :

.

Dans ce chapitre nous avons utilisé le framework Django et le langage python avec l'algorithme RSA afin de développer une application web pour le RSA standard et RSA unidirectionnel qui nous permet de faire le chiffrement, le déchiffrement, la signature et la vérification des données.