

Practica 2

El propósito de este documento es documentar el alcance y uso de la aplicación Cryptofinanciera

Desarrollada con python 3.8, la aplicación cliente está compuesta por tres módulos principales:

- Cifrado
- Signature
- Selfsigned_cert

Por medio del método execute_mode del cliente se ejecutan las instrucciones posteriormente definidas

Cifrado				
Objetivo	Encargado de cifrar de forma simétrico o asimétrica, así como generar los mecanismos necesarios para poder llevar a cabo la encriptación.			
Metodos	Alcance	Parametros	Libreria	Referencia
encrypt_simetrico	Cifra en modo AES	FicheroEntrada, FicheroSalida, Contraseña	pyAesCrypt	https://pypi.org/project/pyAesCrypt/
decrypt_asimetrico	Descifra en modo AES	FicheroEncrptadoEntrada, FicheroDescifradoSalida, Contraseña	pyAesCrypt	https://pypi.org/project/pyAesCrypt/
encrypt_asimetrico	Cifrado asimétrico RSA con llave publica y algoritmo SHA256	FicheroEntrada, FicheroSalida	cryptography.hazmat	https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/
decrypt_asimetrico	Cifrado asimétrico RSA con llave privada y algoritmo SHA256	FicheroEncrptadoEntrada, FicheroDescifradoSalida	cryptography.hazmat	https://cryptography.io/en/latest/_modules/cryptography/hazmat/backends/
generate_keys	Se crean las llaves pública y privada con un tamaño de 2048*2 bytes para poder encriptar un archivo con un peso menor a 500 bytes		cryptography.hazmat	https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/
store_keys	Almacena las llaves en sus ficheros respectivos	Llave publica, Llave privada	cryptography.hazmat	https://cryptography.io/en/latest/_modules/cryptography/hazmat/backends/
create_keys	Encargada de orquestar la generación y guardado de las llaves pública y privada			
read_public_key	Obtiene la llave publica de su fichero respectivo			
read_private_key	Obtiene la llave privada de su fichero respectivo			

Signature				
Objetivo	Firma de fichero			
Metodos	Alcance	Parametros	Libreria	Referencia
sign_file	Se firma el fichero utilizando hash256	FicheroEntrada	hashlib RSA	https://docs.python.org/3/library/hashlib.html https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html
verify_signature	Se verifica el hash256 con el hash generado sobre la firma	FicheroEntrada, firma, tuplaRSA(modulo'exponente privado)	hashlib RSA	https://docs.python.org/3/library/hashlib.html https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html

Ejecución y funcionalidades:

El programa se invoca de la siguiente forma: CriptoFinanciera.exe -m [cs | ds | h | vh | ca | da | cert | ts | tsv] [-p contraseña] -i fichero.xml [-ad adicionales.txt] [-o salida.cpt]

-m indica el modo:

- help: despliega interfaz de ayuda
- cs/ds: cifrado/descifrado simétrico. La contraseña se establecerá con -p
- h/vh: función resumen y su verificación
- ca/da: cifrado/descifrado asimétrico. Las claves pública/privada se guardan en la misma carpeta
- cert: Creación de un certificado X.509, cuyo nombre se especificará con -o

Ejemplos:

Descifrado simétrico:

.\CriptoFinanciera.exe -m ds -p asd -i transaction.aes -o transaction_descifrada.xml

```
Windows PowerShell
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__> .\CriptoFinanciera.exe -m cs -p asd -i transaction.xml -o transaction.aes
Resultado de encriptcion en transaction.aes
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__>
```

Función resumen (calculo sha256):

.\CriptoFinanciera.exe -m h -i transaction.xml

Función verificación sha256:

.\CriptoFinanciera.exe -m vh -i transaction.xml -ad c68ab63028b793c18891cc289d575c1577cb67be0a5e0ce416adbe0921f1a752

```
Windows PowerShell
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__> .\CriptoFinanciera.exe -m h -i transaction.xml
sha256 para fichero transaction.xml: c68ab63028b793c18891cc289d575c1577cb67be0a5e0ce416adbe0921f1a752
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__> .\CriptoFinanciera.exe -m vh -i transaction.xml -ad c68ab63028b793c18891cc289d575c1577cb67be0a5e0ce416adbe0921f1a752
sha256 para fichero transaction.xml: c68ab63028b793c18891cc289d575c1577cb67be0a5e0ce416adbe0921f1a752
Verificacion valida: True
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__>
```

Cifrado asimétrico:

```
.\CriptoFinanciera.exe -m ca -i transaction.xml -o transaction.ca
```

Descifrado asimétrico:

```
.\CriptoFinanciera.exe -m da -i transaction.ca -o transaction_ca.xml
```

```
Windows PowerShell
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__> .\CriptoFinanciera.exe -m ca -i transaction.xml -o transaction.ca
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__> .\CriptoFinanciera.exe -m da -i transaction.ca -o transaction_ca.xml
PS D:\DEV\GIT_REPO\uc3m\my\uc3m-Blockchain\practica2\dist\__main__>
```

base_library.zip	01/05/2020 21:44	Compressed (zipp...	764 KB
CriptoFinanciera.exe	01/05/2020 21:44	Application	2.463 KB
libcrypto-1_1.dll	01/05/2020 19:54	Application exten...	3.303 KB
libffi-7.dll	01/05/2020 19:54	Application exten...	33 KB
libssl-1_1.dll	01/05/2020 19:54	Application exten...	671 KB
private_key.pem	01/05/2020 21:54	PEM File	4 KB
public_key.pem	01/05/2020 21:54	PEM File	1 KB
pyexpat.pyd	01/05/2020 19:54	Python Extension ...	186 KB
python38.dll	01/05/2020 19:54	Application exten...	4.098 KB
select.pyd	01/05/2020 19:54	Python Extension ...	27 KB
tcl86t.dll	01/05/2020 19:54	Application exten...	1.666 KB
tk86t.dll	01/05/2020 19:54	Application exten...	1.434 KB
transaction.aes	01/05/2020 21:47	AES File	1 KB
transaction.ca	01/05/2020 21:54	CA File	1 KB
transaction.xml	13/04/2020 19:41	XML Document	1 KB
transaction_ca.xml	01/05/2020 21:54	XML Document	1 KB
unicodedata.pyd	01/05/2020 19:54	Python Extension ...	1.071 KB

Creación certificado X509:

```
.\CriptoFinanciera.exe -m cert -o mi_cert.crt
```

libssl-1_1.dll	01/05/2020 19:54
mi_cert.crt	01/05/2020 21:56

 Recoverable Signature

X Alvaro Suarez

Alvaro Suarez

Signed by: 991c1c1a7dd38f9a