# Cryptography

<https://www.youtube.com/watch?v=Q35FGQsyDZI&list=PLEiONtPux_g6-mIFgzfvqzpjVL6UxKcWi>

# Asymmetric Key creation

Brief on Symmetric and Asymmetric Keys

Digital Certificates - X.509 and GPG

Commands to generate asymmetric keys and digital certificate.

**X.509 commands:**

*Key Pair Creation:*

openssl req -new -x509 -newkey rsa:2048 -keyout Private.key -out Public.crt -days 365 -nodes -sha256

*Dump Private key contents:*

openssl rsa -in Private.key -noout –text

*Dump public key contents:*

openssl x509 -in Public.crt -noout -text

*GPG Commands:*

*Key pair creation:*

gpg --gen-key

*List gpg keys:*

gpg --list-keys

*Dump key contents:*

gpg -a --export key-name | gpg --list-packets --debug 0x02

# Creating dig signature python lib – cripto

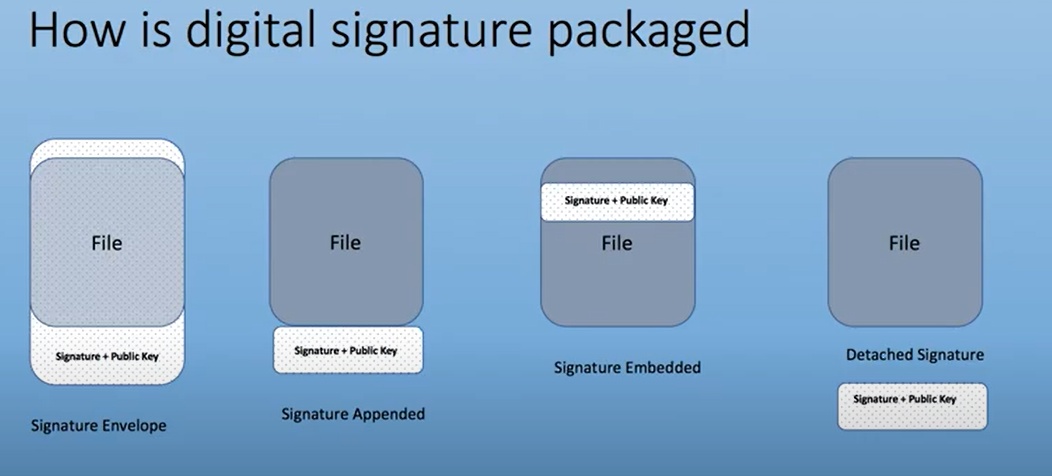
<https://medium.com/@securegns/implementing-asymmetric-encryption-to-secure-your-project-35368049cb5f>

# Encription and decription

openssl enc -aes-128-cbc -in Plaintext.txt -K ABCDEF12345 -iv ABCDEF > Cipher.txt

openssl enc -d -aes-128-cbc -in Cipher.txt -K ABCDEF12345 -iv ABCDEF

# Packing file and digital signature



# Digital signature creation with python

1. Create private and public key

import Crypto

from Crypto.PublicKey import RSA

from Crypto import Random

import base64

def rsakeys():

     length=1024

     privatekey = RSA.generate(length, Random.new().read)

     publickey = privatekey.publickey()

     return privatekey, publickey

def encrypt(rsa\_publickey,plain\_text):

     cipher\_text=rsa\_publickey.encrypt(plain\_text,32)[0]

     b64cipher=base64.b64encode(cipher\_text)

     return b64cipher

def decrypt(rsa\_privatekey,b64cipher):

     decoded\_ciphertext = base64.b64decode(b64cipher)

     plaintext = rsa\_privatekey.decrypt(decoded\_ciphertext)

     return plaintext

privatekey,publickey=rsakeys() #generating keys

try:

    with open('master\_private.pem', 'wb') as keyfile:

        keyfile.write(privatekey.export\_key('PEM'))

        keyfile.close()

        print ("[\*] Successfully created your MASTER RSA private key")

except Exception as e:

    print ("[\*] Error creating your private key", e)

try:

    with open('master\_public.pem', 'wb') as keyfile:

        keyfile.write(publickey.export\_key('PEM'))

        keyfile.close()

        print ("[\*] Successfully created your MASTER RSA publickey key")

except Exception as e:

    print ("[\*] Error creating your publickey key", e)

1. Signature creation

* python digi-sig.py -s Private.key my\_data.txt my\_data.sig

1. Verify signature

* python digi-sig.py -v Public.crt my\_data.txt my\_data.sig