```python
from Crypto.Random import get_random_bytes
from Crypto.Cipher import DES, AES
from Crypto.Util.Padding import pad,unpad
from Crypto.Util import Counter

class AES_CIPHER:

    BLOCK_SIZE_AES = 16 # AES: Bloque de 128 bits

    def __init__(self, key):
        """Inicializa las variables locales"""
        self.key=key
    def cifrar(self, cadena, IV, mac_length):
        """Cifra el parámetro cadena (de tipo String) con una IV específica, y
           devuelve el texto cifrado binario"""
        # Crea un mecanismo de cifrado AES en modo GCM
        cadena = cadena.encode("utf-8")
        cypher = AES.new(self.key, AES.MODE_GCM, nonce=IV, mac_len=mac_length)

        # Cifro haciendo que el bloque sea múltiplo del tamaño del bloque
        cyphertext = cypher.encrypt(pad(cadena, self.BLOCK_SIZE_AES))
        return cyphertext

    def descifrar(self, cifrado, IV, mac_length):
        """Descifra el parámetro cifrado (de tipo binario) con una IV específica, y
           devuelve la cadena en claro de tipo String"""
        # Creo un mecanismo de descifrado AES en modo GCM
        decipher_aes = AES.new(self.key, AES.MODE_GCM, nonce=IV, mac_len=mac_length)

        # Descifro, elimino el padding y recupero la cadena
        new_data = unpad(decipher_aes.decrypt(cifrado),self.BLOCK_SIZE_AES).decode("utf-8",
"ignore")
        return new_data

key = get_random_bytes(16) # Clave aleatoria de 128 bits
IV = get_random_bytes(16)  # IV aleatorio de 64 bits (tamaño del bloque / 2)
datos = "Hola Mundo con AES en modo GCM"
print(datos)
d = AES_CIPHER(key)
mac_length = 16
cifrado = d.cifrar(datos, IV, mac_length)
print("Texto cifrado")
print(cifrado)
descifrado = d.descifrar(cifrado, IV, mac_length)
print("Texto descifrado")
print(descifrado)

############################
############################
############################

```