IS4

```
import hashlib
from Crypto import Random
from Crypto.Cipher import AES
from base64 import b64encode, b64decode

class AESCipher(object):
 def __init__(self, key):
 self.block_size = AES.block_size
 self.key = hashlib.sha256(key.encode()).digest()
 def encrypt(self, plain_text):
 plain_text = self.__pad(plain_text)
 iv = Random.new().read(self.block_size)
 cipher = AES.new(self.key, AES.MODE_CBC, iv)
 encrypted_text = cipher.encrypt(plain_text.encode())
 return b64encode(iv + encrypted_text).decode("uɳ-8")

 def decrypt(self, encrypted_text):
 encrypted_text = b64decode(encrypted_text)
 iv = encrypted_text[:self.block_size]
 cipher = AES.new(self.key, AES.MODE_CBC, iv)
 plain_text = cipher.decrypt(encrypted_text[self.block_size:]).decode("uɳ-8")
 return self.__unpad(plain_text)


 def __pad(self, plain_text):
 number_of_bytes_to_pad = self.block_size - len(plain_text) % self.block_size
 ascii_string = chr(number_of_bytes_to_pad)
 padding_str = number_of_bytes_to_pad * ascii_string
 padded_plain_text = plain_text + padding_str
 return padded_plain_text
 @staϴcmethod
 def __unpad(plain_text):
 last_character = plain_text[len(plain_text) - 1:]
 return plain_text[:-ord(last_character)]

# Example usage
def main():
 key = "mysecretpassword" # This is the secret key
 message = "Hello, world!" # This is the message to be encrypted
 # Create an instance of AESCipher with the secret key
 cipher = AESCipher(key)
 # Encrypt the message
 encrypted_text = cipher.encrypt(message)
 print("Encrypted text:", encrypted_text)
 # Decrypt the encrypted text
```

```
 decrypted_text = cipher.decrypt(encrypted_text)
 print("Decrypted text:", decrypted_text)

if __name__ == "__main__":
 main()
```

OUTPUT

```
Encrypted text: klkXa5ArwD/HZ29pjQpYX21Tt/BrLJDT2EO6D2hKUzI=
Decrypted text: Hello, world!
```