**GitHub:**

<https://github.com/JalliTimothy/Crypto>

**AES Code**

from Crypto.Cipher import AES

from Crypto.Random import get\_random\_bytes

import base64

def pad(text):

while len(text) % 16 != 0:

text += ' '

return text

def aes\_encrypt(key, plaintext):

cipher = AES.new(key, AES.MODE\_ECB) # ECB for simplicity

padded\_text = pad(plaintext).encode('utf-8')

ciphertext = cipher.encrypt(padded\_text)

return base64.b64encode(ciphertext).decode('utf-8')

def aes\_decrypt(key, ciphertext):

cipher = AES.new(key, AES.MODE\_ECB)

decoded\_ct = base64.b64decode(ciphertext)

plaintext = cipher.decrypt(decoded\_ct).decode('utf-8').rstrip(' ')

return plaintext

key = get\_random\_bytes(16)

plaintext = "SRM University AP"

print("Original:", plaintext)

encrypted = aes\_encrypt(key, plaintext)

print("Encrypted:", encrypted)

decrypted = aes\_decrypt(key, encrypted)

print("Decrypted:", decrypted)

**OUTPUT**



**AES Code**

def KSA(key):

"""Key Scheduling Algorithm"""

key\_length = len(key)

S = list(range(256))

j = 0

for i in range(256):

j = (j + S[i] + key[i % key\_length]) % 256

S[i], S[j] = S[j], S[i]

return S

def PRGA(S):

"""Pseudo-Random Generation Algorithm"""

i = 0

j = 0

while True:

i = (i + 1) % 256

j = (j + S[i]) % 256

S[i], S[j] = S[j], S[i]

K = S[(S[i] + S[j]) % 256]

yield K

def RC4(key, plaintext):

key = [ord(c) for c in key]

S = KSA(key)

keystream = PRGA(S)

res = []

for c in plaintext:

val = ("%02X" % (ord(c) ^ next(keystream)))

res.append(val)

return ''.join(res)

def RC4\_decrypt(key, ciphertext\_hex):

key = [ord(c) for c in key]

S = KSA(key)

keystream = PRGA(S)

ciphertext\_bytes = bytes.fromhex(ciphertext\_hex)

res = ''.join(chr(b ^ next(keystream)) for b in ciphertext\_bytes)

return res

key = "secretkey"

plaintext = "SRM University AP"

print("Original:", plaintext)

encrypted = RC4(key, plaintext)

print("Encrypted (hex):", encrypted)

decrypted = RC4\_decrypt(key, encrypted)

print("Decrypted:", decrypted)

**OUTPUT**

****