Ceaser Cipher

```
plaintext = ""
# cipher = ""
words = []
cipher_list = []
def encrypt(key, plaintext):
    for letters in plaintext:
        z = ord(letters) - 97 + key
        words.append(z)

    for word in words:
        z = chr((word%26)+97)
        cipher_list.append(z)
        cipher = ''.join(cipher_list)
    return cipher

def enc(key, plaintext):
    cipher = ""
    for letters in plaintext:
        asci_value = ord(letters) - 97 + key
        asci_value %= 26
        cipher += "".join(chr(asci_value+97))
    return cipher
plaintext = input("Enter thr message you want to encrypt: ")
key = int(input("Enter the key: "))
# encrypt(key,plaintext)
e = enc(key,plaintext)
print(e)
```

Railfence

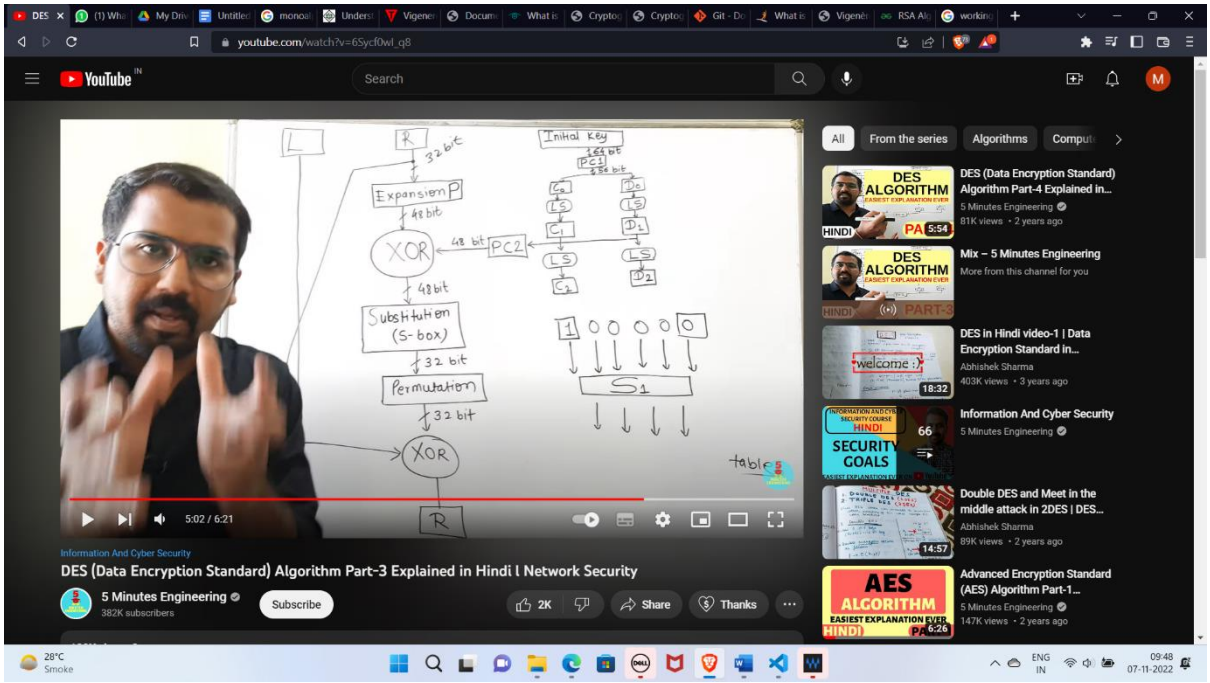
```
message = "I am tired"
key = 3
def rail():
    test = [['/n' for i in range (len(message))] for j in range (key)]

    down = False
    row, col = 0, 0

    for i in range (len(message)):
        if(row==0) or (row==key-1):
            down = not down

        test[row][col] = message[i]
        col +=1
```

```
        if down==True:
            row +=1
        else:
            row -=1

        cipher = ""

    for i in range (key):
        for j in range (len(message)):
            if test[i][j] != '/n':
                cipher += "".join(test[i][j])
    return cipher


r = rail()
print(r)
```

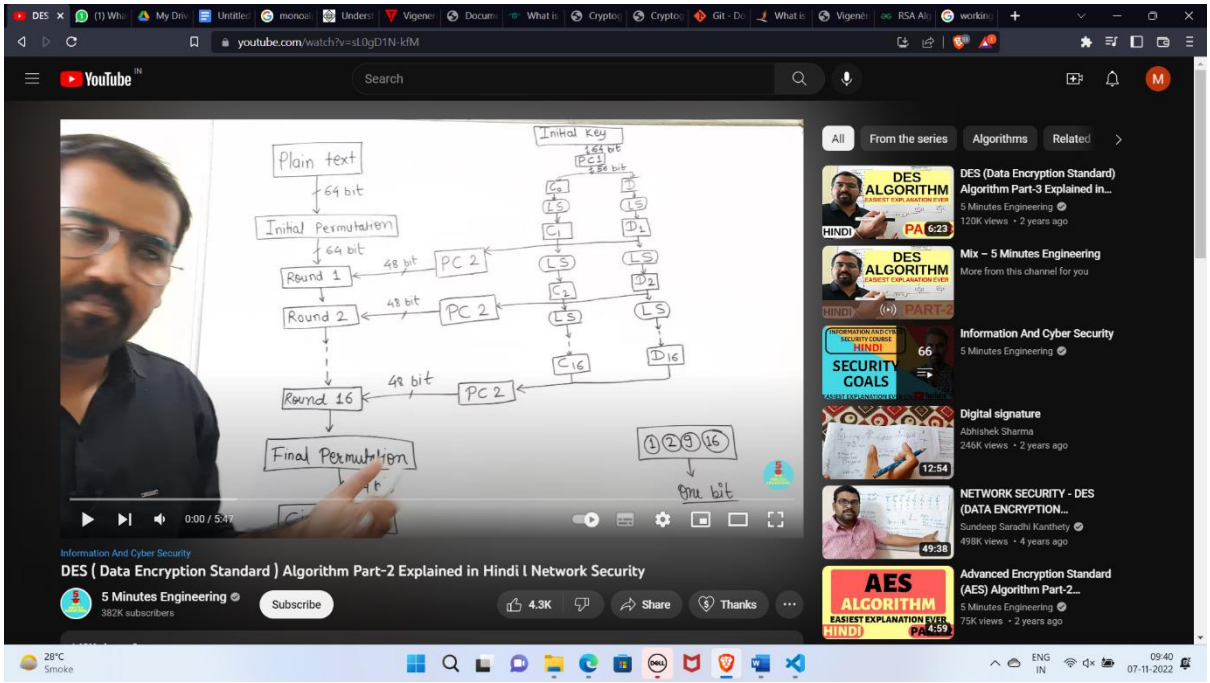
RSA

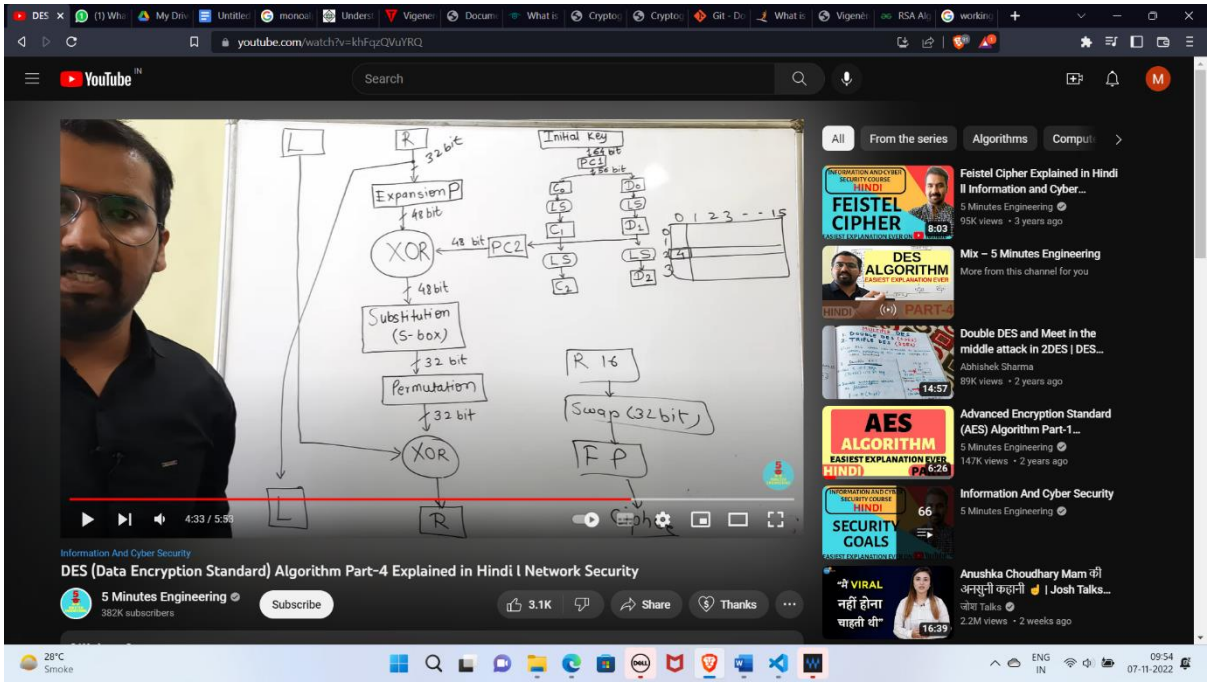
```
import math
import random

list1 = []
def prime_num():
    for num in range(2,250):
        isPrime = True
        for i in range(2,num):
            if num%i == 0:
                isPrime = False
        if isPrime == True:
            list1.append(num)
prime_num()

p = random.choice(list1)
q = random.choice(list1)

# print("p:",p,"q:",q)

n = p * q
m = (p-1)*(q-1)
e=0
def get_key(m):
    for i in range(2,m):
        if math.gcd(i, m) == 1:
            break
    return i
```

```
e = get_key(m)
print("Public Key:",e)

def mod_inverse(a,b):
    for x in range(1,m):
        if((a%m)*(x%m) % m==1):
            return x

d = mod_inverse(e,m)
print("Private Key:",d)

# encrypt c = m^e mod n
def encryption(m):
    return pow(m,e) % n

#  decrypt m = c^d mod n
def decryption(c):
    return pow(c,d) % n

choice = 0
while(choice!=3):
    choice = int(input("Press\n1 to encrypt\n2 to decrypt\n3 to exit\n"))
    if choice == 1:
        m = int(input("Enter a number you want to encrypt: "))
        print(encryption(m))
    if choice == 2:
        c = int(input("Enter a number you want to decrypt: "))
        print(decryption(c))
    if choice == 3:
        print("Bye")
```


youtube.com/watch?v=6Sycf0wl_q8
Search
All
From the series
Algorithms
DES (Data Encryption Standard) Algorithm Part-4 Explained in...
5 Minutes Engineering
81K views • 2 years ago
Mix – 5 Minutes Engineering
More from this channel for you
DES in Hindi video-1 | Data Encryption Standard in...
Abhishek Sharma
403K views • 3 years ago
Information And Cyber Security
5 Minutes Engineering
Double DES and Meet in the middle attack in 2DES | DES...
Abhishek Sharma
89K views • 2 years ago
Advanced Encryption Standard (AES) Algorithm Part-1...
5 Minutes Engineering
147K views • 2 years ago
Information And Cyber Security
DES (Data Encryption Standard) Algorithm Part-3 Explained in Hindi l Network Security
5 Minutes Engineering
382K subscribers
Subscribe
2K
Share
Thanks
5:02 / 6:21

DES


youtube.com/watch?v=sL0gD1N-kfM
Search
All
From the series
Algorithms
Related
DES (Data Encryption Standard) Algorithm Part-3 Explained in...
5 Minutes Engineering
120K views • 2 years ago
Mix – 5 Minutes Engineering
More from this channel for you
Information And Cyber Security
5 Minutes Engineering
Digital signature
Abhishek Sharma
246K views • 2 years ago
NETWORK SECURITY - DES (DATA ENCRYPTION...
Sundeep Saradhi Kanthety
498K views • 4 years ago
Advanced Encryption Standard (AES) Algorithm Part-2...
5 Minutes Engineering
75K views • 2 years ago
Information And Cyber Security
DES (Data Encryption Standard) Algorithm Part-2 Explained in Hindi l Network Security
5 Minutes Engineering
382K subscribers
Subscribe
4.3K
Share
Thanks
0:00 / 5:47



Commands that run on zenmap/nmap

nmap -sn 10.0.5.*

Displays the active nodes.

nmap -sn 10.0.5.237

Displays the whether specific node is active.

nmap -T5 10.0.5.237

Displays the ports of specific node.

nmap -A 10.0.5.237

Displays the operating system of specific node(OS finger printing).

Commands that run on terminal

nmap -sP 10.0.5.237

Used for ping scanning of specific node.

nmap -p T:80 10.0.5.237

Used for tcp scanning of specific node.

nmap -p U:53 10.0.5.237

Used for udp scanning of specific node.