TEACHNOOK

CYBER SECURITY

SEPTEMBER 2022 Batch

CS-MINOR-SEPTEMBER.

MINOR PROJECT

Different Types Of Ciphers

STUDENT DETAILS

NAME: AKSHITHA R

UNIVERSITY: SRMIST

DEGREE: BTech computer science.

YEAR OF STUDY: 3rd

E mail: 2407akshi@gmail.com

Phn no: +919884972919

TABLE OF CONTENTS:

INTRODUCTION:

CRYPTOGRAPHY:

Cryptography is a technique used for securing information and communications.Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”. In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

CIPHER AND CIPHERTEXT:

Ciphertext is encrypted text transformed from plaintext using an encryption algorithm. Ciphertext can't be read until it has been converted into plaintext (decrypted) with a key. The decryption cipher is an algorithm that transforms the ciphertext back into plaintext.

The term cipher is sometimes used as a synonym for ciphertext. However, it refers to the method of encryption rather than the result.

Types of ciphers:

Substitution ciphers:

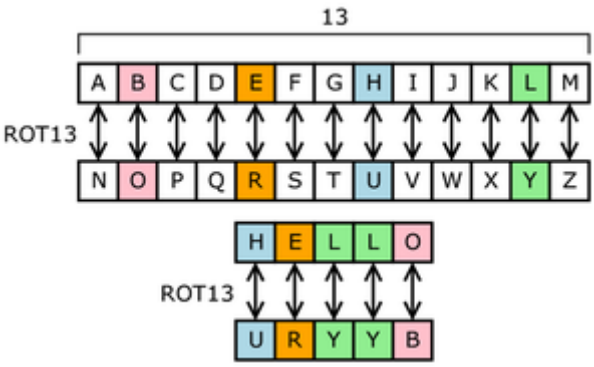
Description:

1. Replace bits, characters, or character blocks in plaintext with alternate bits, characters or character blocks to produce ciphertext
2. A substitution cipher may be monoalphabetic or polyalphabetic:

(i) Caesar Cipher:

Description:

1. The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique.
2. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet.



Algorithm:

Step 1: Choose a shift value.

Step 2: The shift value remains same for each character.

Step 3: Shift each character in the plain text by the shift value.

For example:
Shift = 2 (ROT-2)
a=c, b=d, c=e, ……, y=a, z=b.

step 4: Number of characters in both plain text and the cipher are the same.

Program:

Simple implementation using python program :

```
def encrypt(string,shift):
    cipher=""
    for i in string:
        if i == ' ':
            cipher+=i
        else:
            cipher+=chr((ord(i)-97+shift)%26+97)
    return cipher

text = input("Enter text to be encrypted : ")
s = int(input("Enter the shift value : "))
print(encrypt(text,s))
```

OUTPUT:

```
Enter text to be encrypted : hello world
Enter the shift value : 2
jgnnq yqtnf
```

Online tool implementation:

https://planetcalc.com/7984/

Step 1: Enter the key (rotated order of alphabets)

Key: ABCDEFGHIJKLMNOPQRSTUVWXYZ
CDEFGHIJKLMNOPQRSTUVWXYZAB

Step 2: Enter the text to be encrypted and select the encode option.

Option
(•) Encode ( ) Decode

Plain text
HELLO WORLD

Output cipher:

Transformed text
JGNNQ YQTNF

(ii) Monoalphabetic Cipher:

Description:

1. Caesar cipher is easy to break.
2. In monoalphabetic, each alphabet in plain text can be replaced by any other alphabet except the original alphabet.
3. That is, A can be replaced by any other alphabet from B to Z. B can be replaced by A or C to Z. C can be replaced by A, B, and D to z, etc.
4. Mono alphabetic cipher causes difficulty to crack the message as there are random substitutions and a large number of permutation and combination are available.

Plain Alpha.	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher Alpha.	L	O	R	F	W	S	E	V	A	M	C	P	N	D	B	Q	G	J	T	Y	I	U	X	H	Z	K

Algorithm:

Step 1: Choose a substitution alphabet of length 26, which corresponds to the character to be substituted for characters a-z.

Step 2: Character substitution for blank spaces and punctuations can also be included.

Step 3: Substitute the characters in plain text one at a time.

Program:

Simple implementation using python

```python
def encrypt(plaintext,substitution):
    ciphertext=''
    for i in plaintext:
        if i==" ":
            ciphertext+=i
        elif i.isupper():
            ciphertext+=substitution[ord(i)-65].toupper()
        else:
            ciphertext+=substitution[ord(i)-97]
    return ciphertext

subs_alph = "qwertyuiopasdfghjklzxcvbnm"
plain_text = input("Enter the plain text to be encrypted : ")
print("cipher text : ",encrypt(plain_text,subs_alph))
```

OUTPUT:

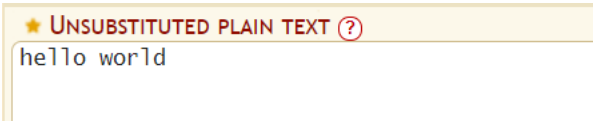
```
Enter the plain text to be encrypted : hello world
cipher text :  itssg vgksr
```

Online tool implementation:

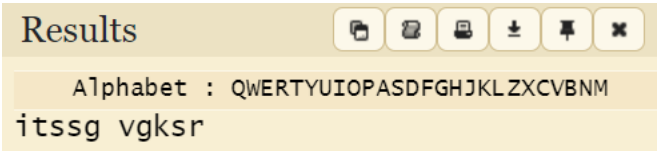
https://www.dcode.fr/monoalphabetic-substitution

Step 1: Enter the substitution alphabet: string of length 26.

Step 2: Enter the plain-text.



Output cipher:



(iii) Polyalphabetic cipher:

Description:

1. Polyalphabetic Cipher is also known as Vigenère Cipher.
2. It uses multiple substitution alphabets for encryption.
3. Vigenère square or Vigenère table is used to encrypt the text. The table contains 26 alphabets written in different rows.
4. Each alphabet is cyclically shifted to the left according to the previous alphabet, equivalent to the 26 possible Caesar Ciphers.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Algorithm:

Step 1: Choose a keyword.

Step 2: If the length of the keyword is less than the length of plain text, the key is generated by repeating the keyword until the length is reached.

Step 3: At different points in the encryption process, the cipher uses a different alphabet from one of the rows.

The alphabet used at each point depends on a repeating keyword.

Program:

Simple implementation using python

```
def generateKey(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)
    else:
        for i in range(len(string) -
                       len(key)):
            key.append(key[i % len(key)])
    return("" . join(key))
def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) +
             ord(key[i])) % 26
        x += 97
        cipher_text.append(chr(x))
    return("" . join(cipher_text))
plain_text = input("Enter the plain text to be encrypted : ")
keyword = input("Enter the keyword : ")
key = generateKey(plain_text, keyword)
print("Ciphertext :", cipherText(plain_text,key))
```

OUTPUT:

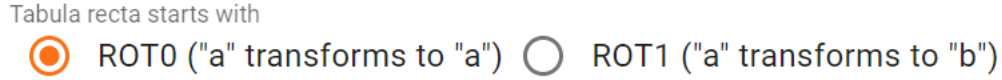
```
Enter the plain text to be encrypted : hello world
Enter the keyword : www
Ciphertext : pmttwvewztl
```

Online tool implementation:

https://planetcalc.com/2468/

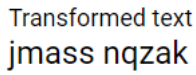
Step 1: Enter the keyword.

Step 2: Choose whether or not to add rotation.



Step 3: Enter the plain text to be encrypted.

Output cipher:



Transposition ciphers:

Description:

1. Unlike substitution ciphers that replace letters with other letters, transposition ciphers keep the letters the same, but rearrange their order according to a specific algorithm.
2. For example : In a simple columnar transposition cipher, a message might be read horizontally but would be written vertically to produce the ciphertext.

h	e	l	l
o	w	o	r
l	d		

Algorithm:

Step1: Choose the mode in which the plain text characters should be written and read to get the cipher text.

Step 2: Choose a key.

Key: A string of fixed length, which is a permutation of numbers from 1 to length of the key. For example: 213, 3421 ,35412,etc.

Step 3: According to the chosen mode write the characters of the plain text in either rows or columns.

Step 4: According to the chosen mode and the order of row/columns in the key read the rows/columns to get the cipher text.

Program:

Simple implementation using python :

```
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]
def encode(key, plaintext):
    plaintext=plaintext+'.'*(len(key)-len(plaintext)%len(key))
    order = {
    num: int(val) for num, val in enumerate(key)
    }
    ciphertext = ''
    for index in sorted(order.keys()):
        for part in split_len(plaintext, len(key)):
            if part[order[index]]==".":
                continue
            else:
                ciphertext+=part[order[index]]
    return ciphertext
key=input("Enter key : ")
plain_text=input("Enter the text to be encrypted : ")
print("cipher : ",encode(key,plain_text))
```

Mode Used: Write by rows and read by columns.

OUTPUT:

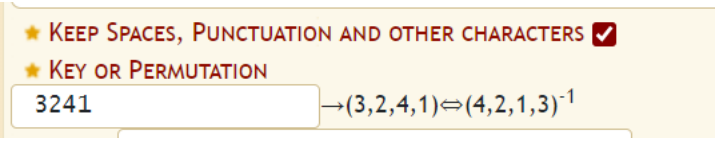
```
Enter key : 2130
Enter the text to be encrypted : hello world
cipher :  lwde llohor
```

Online tool implementation:

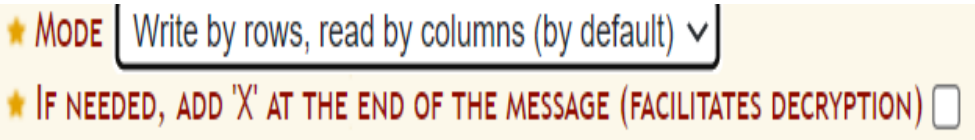
https://www.dcode.fr/transposition-cipher

Step 1: Choose whether or not to include blank spaces and punctuations.

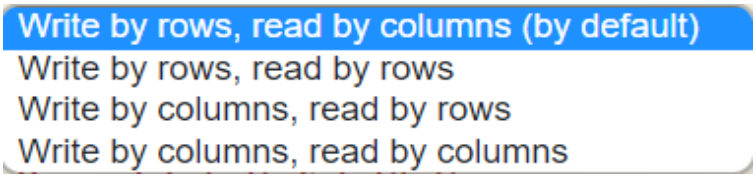
Step 2: Choose a key, which specifies the order in which the columns/rows need to be read.



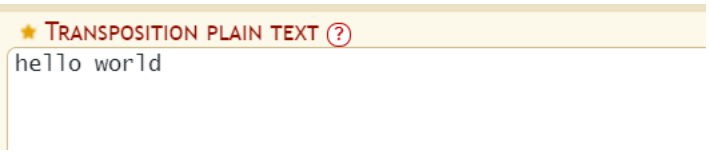
Step 3: Choose the write and read mode. This is the important step as it changes the algorithm according to the mode chosen.



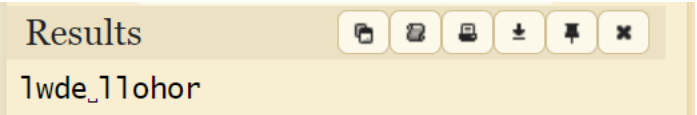
Available modes:



Step 4: Enter the plain text to be encrypted



Output cipher:



Polygraphic ciphers:

Description:

1. Instead of substituting one letter for another letter, a polygraphic cipher performs substitutions with two or more groups of letters. This masks the frequency distribution of letters, making frequency analysis attacks much more difficult.
2. When the length of the block is specifically known, more precise terms are used: for instance, a cipher in which pairs of letters are substituted is **bigraphic**.
3. There are different types of polygraphic ciphers. Let's consider two of the popular ones : Hill cipher and Playfair cipher.
4. Hill cipher: Hill cipher is a polygraphic substitution cipher based on linear algebra.
5. Playfair cipher: Instead of encrypting single letters, the Playfair cipher encrypts pairs of letter(digrams or bigrams). This makes frequency analysis much harder, since there are around 600 combinations instead of 26.

Algorithm: (Hill cipher)

Step 1: Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used.

Step 2: To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26.

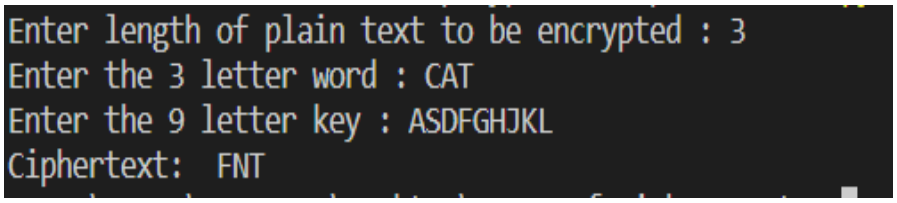
Step 3: To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

Program: (Hill cipher**)**

Simple implementation using python.

```
def getKeyMatrix(key):
    k = 0
    for i in range(n):
        for j in range(n):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1
def encrypt(messageVector):
    for i in range(n):
        for j in range(1):
            cipherMatrix[i][j] = 0
            for x in range(n):
                cipherMatrix[i][j] += (keyMatrix[i][x] *
                                       messageVector[x][j])
            cipherMatrix[i][j] = cipherMatrix[i][j] % 26
def HillCipher(message, key):
    getKeyMatrix(key)
    for i in range(n):
        messageVector[i][0] = ord(message[i]) % 65
    encrypt(messageVector)
    CipherText = []
    for i in range(n):
        CipherText.append(chr(cipherMatrix[i][0] + 65))
    print("Ciphertext: ", "".join(CipherText))
n = int(input("Enter length of plain text to be encrypted : "))
message = input(f"Enter the {n} letter word : ")
key = input(f"Enter the {n*n} letter key : ")
keyMatrix = [[0] * n for i in range(n)]
messageVector = [[0] for i in range(n)]
cipherMatrix = [[0] for i in range(n)]
HillCipher(message, key)
```

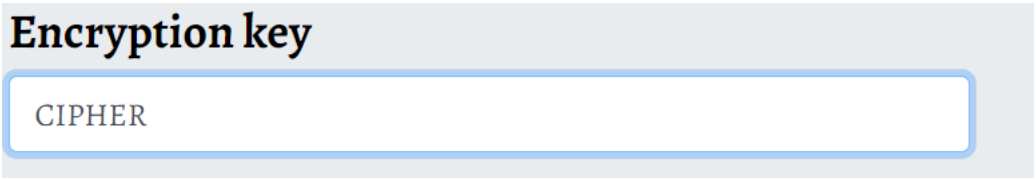
OUTPUT:



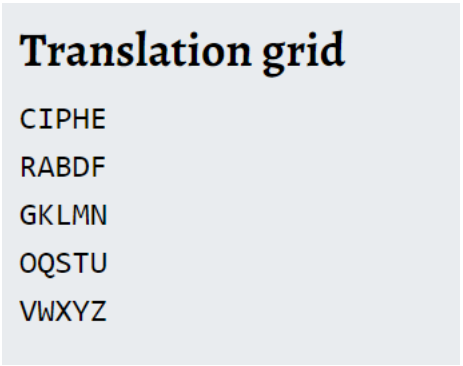
Online tool implementation: (Playfair cipher**)**

https://www.boxentriq.com/code-breaking/playfair-cipher

Step 1: Choose an encryption key , a string.



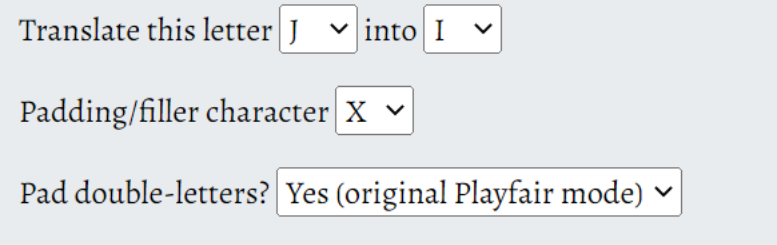
The grid is formed by first taking a code word (with duplicate letters removed) and then adding any alphabet characters missing



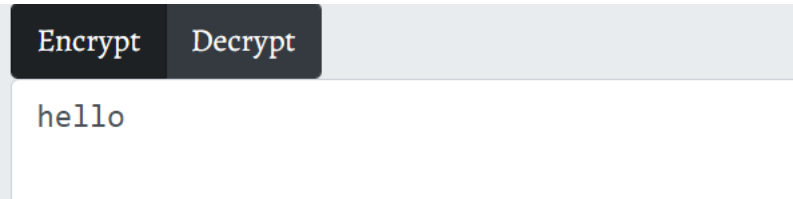
Step 2: A grid of 5x5 letters is used for encryption. Since there are only 25 spots, one character has to be omitted (for instance J, which is replaced by I).

Step 3: Choose a padding character.

Options:



Step 4: Enter the plain text to encrypt.



Output cipher:

Private-key cryptography:

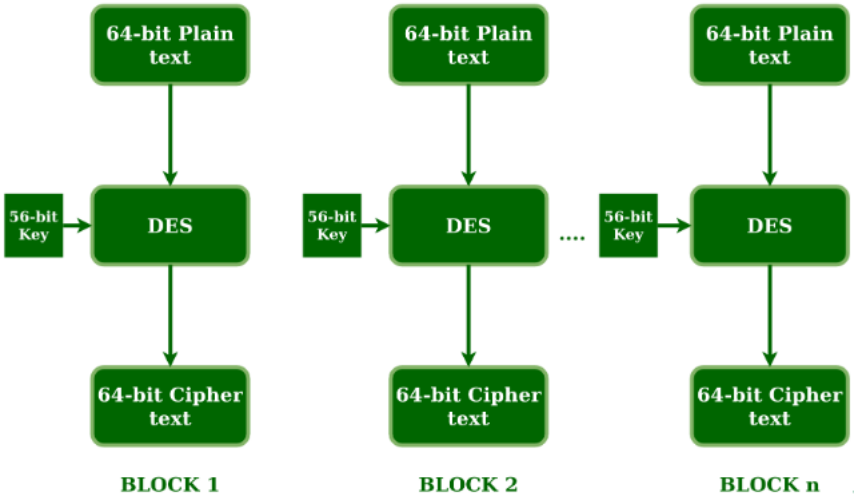
Description:

1. The private-key cryptography is known as symmetric cryptography.
2. It is symmetrical because there is only one key that is called a secret key.
3. The same key (secret key) is used for encryption and decryption.
4. The only key is copied or shared by another party to decrypt the cipher text.
5. It is faster than public-key cryptography.
6. It is less secure than public-key cryptography.
7. Examples: Data Encryption Standard (DES), Advanced Encryption Standard (AES).

(i) DES:

Description:

Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline.



Algorithm:

Step 1: 64-bit plain text block is handed over to an initial Permutation function. The initial permutation is performed on plain text.

Step 2: The initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).

Step 3: Each LPT and RPT go through 16 rounds of the encryption process.

Step 4: In the end, LPT and RPT are joined and a Final Permutation (FP) is performed on the combined block.

Key transformation:

Step 1: 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available.

Step 2: From this 56-bit key, a different 48-bit Sub Key is generated

Step 3: For this, the 56-bit key is divided into two halves, each of 28 bits.

Step 4: These halves are circularly shifted left by one or two positions, depending on the round.

The result of this process produces 64-bit ciphertext.

Program:

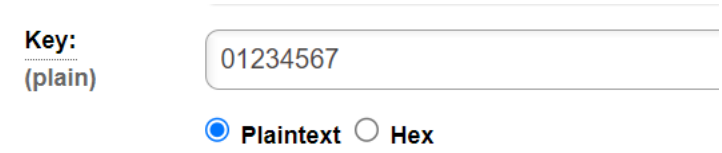
```
from crypto.Cipher import DES

def DES_encrypt(plain_text,key):
    key = bytes(key,'utf-8')
    plain_text = bytes(key,'utf-8')
    cipher_text = DES.new(key, DES.MODE_ECB)
    return cipher_text.encrypt(plain_text)
```

Online tool implementation:

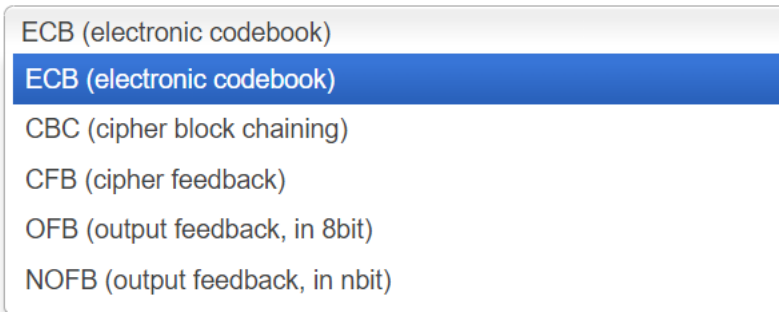
http://des.online-domain-tools.com/

Step 1: Choose a key of length 8 bytes.

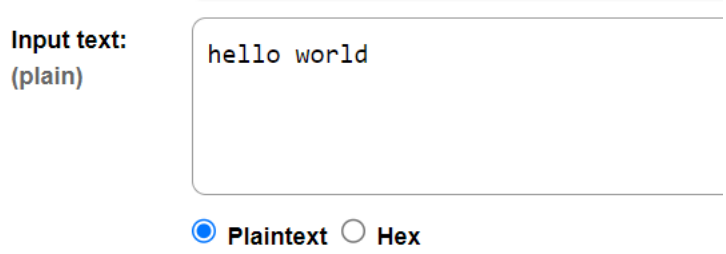
Key can either be in plain text or its hexadecimal equivalent.



Step 2: Choose one of the following modes.



Step 3: Enter plain text or upload an input file to be encrypted.



Output cipher:

Output as hex bytes.

Encrypted text:

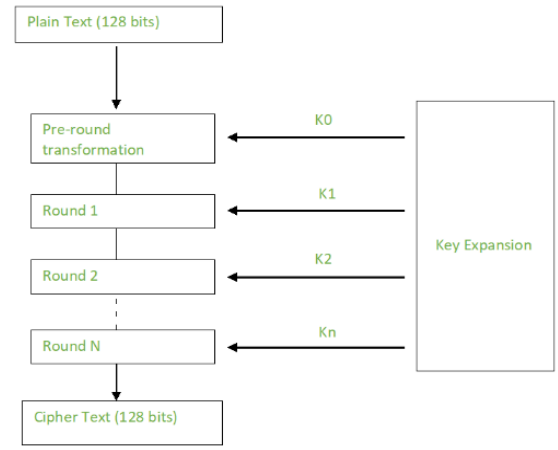
00000000 9a 02 b4 3b 24 40 34 e9 60 cf b7 d6 fa 05 4a 28 | . . ´ ; $ @ 4 é ` Ï · Ö ú . J (

[Download as a binary file] [?] Inactive

(ii) AES:

Description:

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.



Algorithm:

1. AES is a block cipher.

2. The key size can be 128/192/256 bits.
3. Encrypts data in blocks of 128 bits each.
4. The number of rounds depends on the key length:
    - 128 bit key – 10 rounds
    - 192 bit key – 12 rounds
    - 256 bit key – 14 rounds

Step 1: The bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).

Step 2: Next comes the permutation step. In this step, all rows except the first are shifted by one.

Step 3: The Hill cipher is used to jumble up the message more by mixing the block's columns.

Step 4: In the final step, the message is XORed with the respective round key.

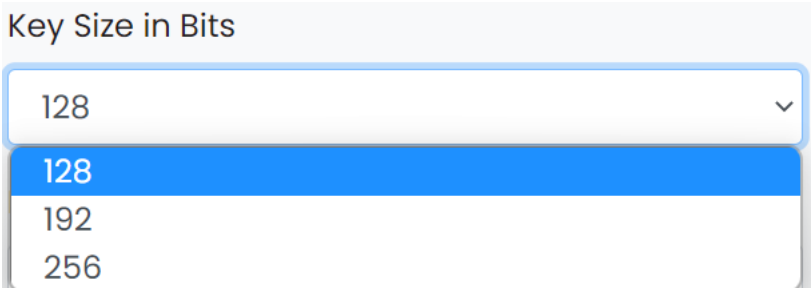
Program:

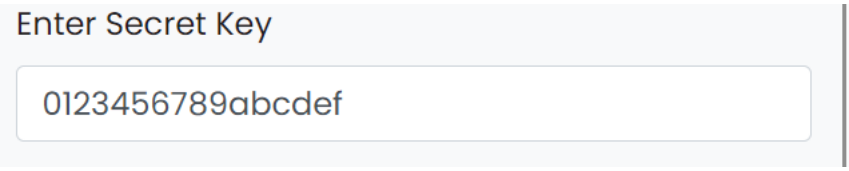
```
from Crypto.Cipher import AES

def AES_encrypt(plain_text,key):
    key = bytes(key,'utf-8')
    plain_text = bytes(key,'utf-8')
    cipher_text = AES.new(key, AES.MODE_ECB)
    return cipher_text.encrypt_and_digest(data)
```

Online tool implementation:

https://www.devglan.com/online-tools/aes-encryption-decryption

Step 1: Choose the size of key in bits (ideal 128 bits). Enter the key in either base64 format or as hex bytes.

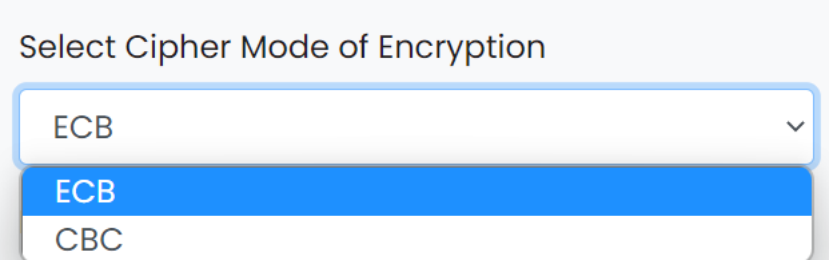




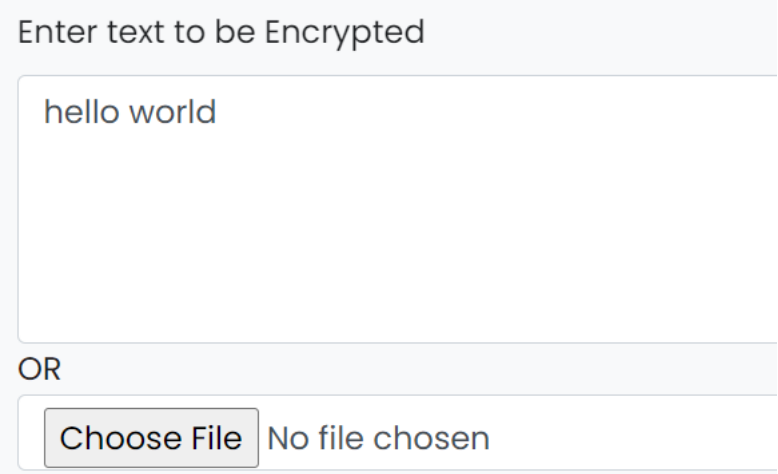
Step 2: Choose one of the two modes.

ECB - Electronic Code Book.

CBC - Cipher block chaining.

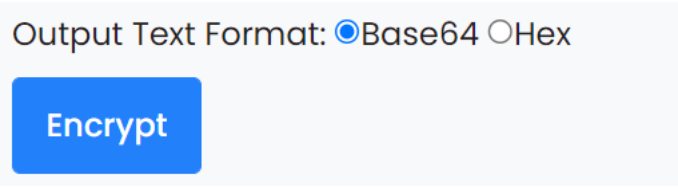


Step 3: Enter the plain text or upload a file to be encrypted.



Output cipher:

Encrypted output can be represented in either hex or base64 format



AES Encrypted Output:

gWm+1O9JqIdFWcWyANqt5w==

Public-key cryptography:

Description:

1. Public-key cryptography is also known as asymmetric cryptography.
2. It is a field of cryptographic systems that use pairs of related keys.
3. Each key pair consists of a public key and a corresponding private key.
4. Key pairs are generated with cryptographic algorithms based on mathematical problems termed one-way functions.
5. A one-way function is a function that is easy to compute on every input, but hard to invert given the image of a random input.
6. Security of public-key cryptography depends on keeping the private key secret.
7. The most widely used public-key cryptosystem is RSA (Rivest–Shamir–Adleman)

Algorithm: (RSA**)**

Public key generation**:**

Step 1: Select two prime numbers P and Q.

Step 2: n=P x Q is the first part of public key.

Step 3: Select a small exponent say e, an integer which is not a factor of n and 1< e < (P-1)(Q-1).

Public Key is made of n and e.

Private key generation**:**

Step 1: calculate Φ(n),such that Φ(n) = (P-1)(Q-1).

Step 2: Private Key, d = (k*Φ(n) + 1) / e for some integer k.

Encryption:

Step 1: Let the ASCII equivalent of each character in the plain text be ‘a’.

Step 2: The encrypted character ASCII equivalent $c=a^e \bmod n$

Decryption:

Step 1: The original characters ASCII equivalent can be obtained by the formula $a = c^d \bmod n$.

Program:

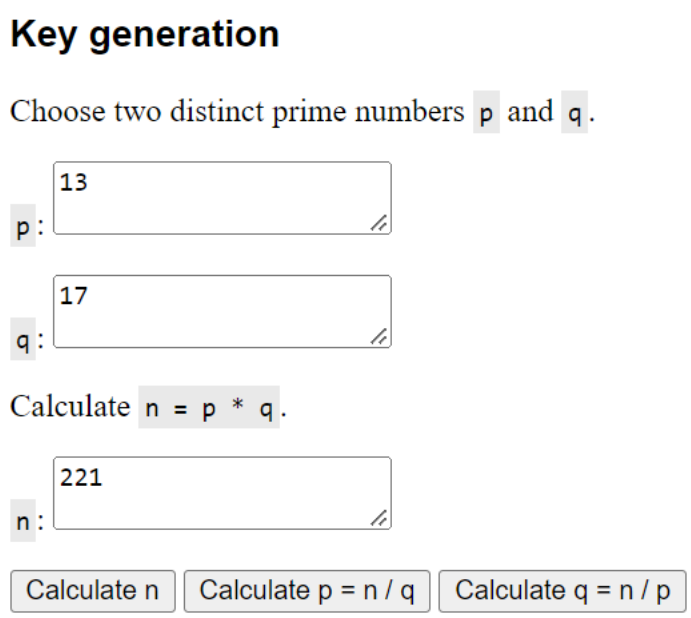
Key pair generation, encryption and decryption methods in python using the rsa module:

```python
import rsa
def generateKeys():
    (publicKey, privateKey) = rsa.newkeys(1024)
    with open('keys/publcKey.pem', 'wb') as p:
        p.write(publicKey.save_pkcs1('PEM'))
    with open('keys/privateKey.pem', 'wb') as p:
        p.write(privateKey.save_pkcs1('PEM'))
def loadKeys():
    with open('keys/publicKey.pem', 'rb') as p:
        publicKey = rsa.PublicKey.load_pkcs1(p.read())
    with open('keys/privateKey.pem', 'rb') as p:
        privateKey = rsa.PrivateKey.load_pkcs1(p.read())
    return privateKey, publicKey
def encrypt(message, key):
    return rsa.encrypt(message.encode('ascii'), key)
def decrypt(ciphertext, key):
    try:
        return rsa.decrypt(ciphertext, key).decode('ascii')
    except:
        return False
```

Online tool implementation:

https://www.tausquared.net/pages/ctf/rsa.html

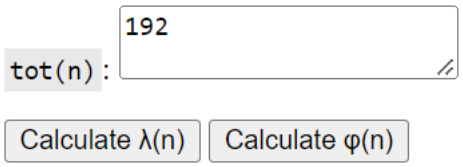
Step 1: Choose 2 prime numbers p and q and calculate n.



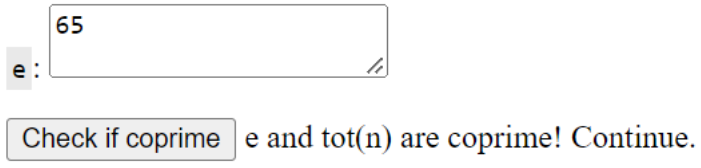
Step 2: Choose which totient function to use

Carmichael's totient function: tot(n) = λ(n) = lcm (p - 1, q - 1)

Euler's totient function: tot(n) = φ(n) = (p - 1) * (q - 1)



Step 3: Choose any number e where 1 < e < tot(n) and e is coprime to tot(n)



Step 4: Compute d, the modular multiplicative inverse of e (mod tot(n)).

d : 65

Calculate d

Step 5: Enter an integer to be encrypted/decrypted.

m : 45

c : 28

Encrypt Decrypt