Exp No: 1
01/08/23

CAESAR CIPHER

AIM: To implement caesar cipher

Encryption:

$$C = E(P, K)$$

$$C = (P + K) \% 26$$

Decryption:

$$P = D(C, K)$$

$$P = (C - K) \% 26$$

$-3 : Q(26) + R$

$R = -(3 + 26Q)$

$3 + 26Q < 0$

$26Q < -3$

CODE:

```
#include <bits/stdc++.h>
using namespace std;
unordered_map<char, int> m;
unordered_map<int, char> m2;

string cipher(string p, int key, unordered_map
<char, int> &m, unordered_map<int, char>&m2){
    string ans = "";
    for(auto &i: p){
        int t = m[i];
        int temp = (t+key) %26;
        ans += m2[temp];
    }
    return ans;
}
```

```
string decrypt (string c, int key, unordered_map
<char, int> &m, unordered_map <int, char> &m2)
{
 string ans = "";
 for (auto &i : c){
      int t = m[i];
      int temp = (t - key) % 26;
      if (temp < 0)
      ans += m2[temp + 26];
      else
       ans += m2[temp];
    }
 return ans;
}
int main () {
    for (int i = 0; i < 26; i++) {
      m[char ('a' + i)] = i;
      m2[i] = char ('a' + i);
    }
    string s;
    int key;
    cin >> s >> key;
    transform (s.begin(), s.end(), s.begin(),
                                  ::tolower);
    string p = cipher (s, key, m, m2);
    cout << "Encrypted: " << p << endl;
    string d = decrypt (p, key, m, m2);
    cout << "Decrypted : " << d << endl;
```

```
return 0;
}
```

OUTPUT:

```
hello 100
Encrypted : dahhk
Decrypted : hello
```

dp verified 04/08/2023

RESULT:

Thus Caesar Cipher has been implemented

HILL CIPHER

AIM:

To perform encryption and decryption for given plaintext with key using Hill cipher.

ALGORITHM:

Encryption:

1. Square key matrix [k] is given.
2. Add filler to the plain text [p] if block size is not compatible.
3. Divide the plaintext into blocks of the same size as the key matrix.
4. Encrypt plaintext into cipher text by matrix multiplication with the key matrix
5. $C = (P \cdot K) \bmod 26$
6. Perform modular arithmetic to keep the values within the range [0, 25]

Decryption:

1. Compute the inverse of the square matrix $[K^{-1}]$
2. Divide the ciphertext into blocks of the same size as the key matrix.
3. Decrypt ciphertext into plaintext by matrix multiplication with the inverse of the key matrix $[K^{-1}]$.
4. Perform modular arithmetic to keep the values within the range [0, 25]
5. $P = (C \cdot K^{-1}) \bmod 26$

AFFINE CIPHER

AIM:

To perform encryption and decryption on the given plaintext with key pair using Affine cipher.

ALGORITHM:

Encryption:

1. Given 'a' and 'b' as the key, where 'a' should be coprime with 26 and 'b' is any integer.
2. Encrypt each letter in the plaintext using the formula:

$$c = (a * p + b) \bmod 26$$

Decryption:

1. Compute a^{-1} (i.e) $(a \times a^{-1}) \bmod 26 = 1$ and a^{-1} should be positive.
2. Decrypt each letter in the cipher text using the formula:

$$p = a^{-1}(c - b) \bmod 26$$

VIGNERE CIPHER

AIM:

To perform encryption and decryption on the given plaintext with key using Vignere Cipher.

ALGORITHM:

Encryption:

1. Convert plaintext 'pt' and key 'k' to uppercase & remove any non-alphabetic characters
2. Create the full key by repeating keyword to match length of plain text.
3. Initialize empty string to store cipher text.
4. Iterate through each character in plain text.
   a) If character is alphabetic
      → Calculate Shift value based on case of characters.
      → Calc encrypted character using formula (character + key - character) mod 26.
      → Convert it to appropriate case & append to cipher text.
   b) If non-alphabetic - append to cipher text as it is.
5. Send cipher text to server.

Decryption:

1. Create full key by repeating keyword to match length of cipher text.
2. Initialize empty string to store decrypted text.
3. Iterate through each character in CT.
   a) If character - alphabetic:
      -> Calculate shift val. based on case.
      -> Calculate decrypted character using (character - key - character + 26) mod 26
      -> Convert decrypted character to appropriate case & append to decrypted text.
   b) If character is not alphabetic, append it to decrypted text as it is.
4. Display decrypted text.

CRYPTOGRAPHIC ATTACKS
- CAESAR ATTACK

ALGORITHM:

→ Receive ciphertext

→ Convert ciphertext to uppercase & remove any spaces or special characters.

→ For each possible key (0 to 25):

→ Initialize an empty text string to store possible decrypted text.

→ Iterate through each character in ciphertext.

→ If character is an alphabetic letter

→ Determine ASCII shift based on 'A' or 'a'

→ Apply formula

decrypted_char = (ord(char) - shift_key) % 26 + shift

→ Append decrypted character to the possible decrypted text.

→ Examine displayed decrypted texts & their contexts to determine correct key that yields meaningful texts

AFFINE ATTACK

AIM:

To find the key of affine cipher text.

ALGORITHM:

→ Receive the ciphertext and known original plaintext.

→ Convert both the given plaintext & ciphertext to uppercase & remove spaces / special chara

→ For each possible 'a' value from 1 to 2

→ For each possible 'b' value

→ Calculate modular inverse of 'a' using the function.

→ If modular inverse exists:

→ Initialize an empty string to store decrypted text.

→ Iterate through each character in cipher text.

→ If the character is an alphabetic letter.

→ Determine ASCII shift based on 'A' or 'a'.

→ Apply decryption formula:

decryp_char = ca - inv *

c Cord (char) - shift) - b)) %.26 + shift

→ Append decrypt_char to t

text.

→ If character not an alphabet → simple append to text.

→ Compare decrypt and plain text, if they match ⇒ we have found 'a' & 'b'

→ Else key not found in given range.

HILL ATTACK

AIM: To find the key of hill cipher te

ALGORITHM:

-> Receive the known plain text & cipher text &
size of key matrix.
-> Truncate plain text & cipher text to match
size of key matrix.
-> Convert the characters in plain text & ciphe
text to numerical values (A=0, 1, ... 25)
-> Reshape PT to CT to nxn matrices.
-> Calculate inverse of Plain text matrix
(PT.
-> Calculate determinant of plain text matrix
& find its modular inverse modulo 26
-> If modular inverse is not found, raise a
exception.
-> Calculate final plain text inverse using
modular inverse and determinant.
-> Calculate key matrix as matrix product
of cipher text matrix & plaintext inverse
mod 26.
-> Displayed recovered key matrix.

VIGNERE

AIM:

To find the key of vigenere cipher text.

ALGORITHM:

→ Initialize a variable "recovered_key" as an empty string.

→ For each possible key length from 1 to length of ciphertext:

→ Divide the known plain text & ciphertext into columns of current key length.

→ For each character pair (pt_char, ct_char) in corresponding columns:

i) Calculate the shift as ~~ord~~ (ord (ct_char) - ord (pt_char)) % 26

ii) Calculate corresponding key character by shifting 'A' by shift amount & taking modulo 26.

→ Append the key character to "recovered_key" string.

* Display "recovered_key" as result.

DATA ENCRYPTION STANDARD

Algorithm:

Encryption:

* Permute the 64-bits in the plain text & divide them into 2 equal halves [LPT & RPT]
* These 32-bit chunk of data will undergo multiple rounds of operation.

Key transformation:

* Initially 64-bit key is transformed into 56-bit key by discarding every 8th bit of initial key.
* From this 56-bit key, 48-bit key is generated during each round. [after left shift of each 28 bit & then permuted contraction).

[In round 1, 2, 9, 16 -> 1 shift, Other rounds -> 2 shifts]

* The RPT is expanded from 32 bits to 48 bits (expansion permutation)
* This 48-bit RPT is xored with 48-bit key & resulting output is given to s-box substitution, which then gives 32 bits as output.

* The 32 bit is again permuted & xored with the LPT.
* This is combined with the original RPT & swapped.
* This is one round
* This is repeated for 16 rounds [where will have separated key during key transformation phase.
* Then in the last round (final round), swap doesn't occur, & final permutation takes place to give 64-bit cipher text as output. [Inverse permutation should be applied].

Decryption:

→ The order of 16 - 48-bit keys is reversed such that K_{16} becomes K_1, K_{15} becomes K_2 & so on.

→ The other steps of encryption is followed as it is to get the decrypted text.