# CHAPTER 1

# A Simple Cipher

## 1.1 FAL3014 Introduction

Cipher Name: FAL3014

The FAL3014 Cipher is a polyalphabetic substitution cipher. It is similar to the Vigenère cipher, but uses a different method of generating the key. It is more secure than Vigenère cipher.

The encryption of the original text is done using the **Vigenère table** but there is a little change in it.

- The table consists of the alphabets written out 51 times in different rows instead of 26 as in Vigenère , each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 51 possible Caesar Ciphers.
- 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.

51= 6+1+12+3+15+14 => FALCON

**Encryption:**

The first letter of the plaintext the first letter of the key. So use row and column of the Vigenère square. Similarly, for the second letter of the plaintext, the second letter of the key is used. The rest of the plaintext is enciphered in a similar fashion.

**Decryption:**

Decryption is performed by going to the row in the table corresponding to the key, finding the position of the ciphertext letter in this row, and then using the column's label as the plaintext.

**Transposition**: Here the encrypted test is passed to a columnar transposition cipher where:

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "3 1 2 4".
4. Any spare spaces are filled with nulls or left blank or placed by a character
5. Finally, the message is read off in columns, in the order specified by the keyword.

**Transposition Encryption**:

- A matrix of Key's length is done containing the no of rows same as the length of key.
- The words of the output of the FAL3014 are placed accordingly below the key in the matrix.
- Then then columns get shifted or transposed w.r.t the alphabetical order of the key.
  Ex: HACK will have the order 3124.
- This is the final output.

**Transposition Decryption**:

- Decipher it, the recipient has to work out the column lengths by dividing the message length by the key length.
- Then, write the message out in columns again, then re-order the columns by reforming the key word.

| H | A | C | K |
|---|---|---|---|
| 3 | 1 | 2 | 4 |
| G | e | e | k |
| s | _ | f | o |
| r | _ | G | e |
| e | k | s | _ |

**Steps**:

1) User inputs a String.
2) The String gets converted into encrypted form of FAL3014 Cipher.
3) This output is passed to Columnar Transposition Cipher.
4) This is the final output
5) To decrypt the output of Columnar Transposition Cipher is passed to the decrypter function.
6) This output is passed to the decrypter function of FAL3014 Cipher.

## 1.2 FAL3014 Cipher Pseudocode

key <- "HACK" # key for transposition

**FUNCTION FOR GENERATE KEY**

```
FUNCTION generateKey(string, key): #generate key with length same as input
   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)])
   ENDIF
      ENDFOR
   RETURN("" . join(key))
ENDFUNCTION
```

**FAL3014 ENCRYPTER FUNCTION**

```
FUNCTION cipherText(string, key): #ecrypting using FAL3014 CIPHER
   cipher_text <- []
   for i in range(len(string)):
      x <- (ord(string[i]) +
          ord(key[i])) % 51
      x += ord('A')
      cipher_text.append(chr(x))
   ENDFOR
   RETURN("" . join(cipher_text))
ENDFUNCTION
```

**COLUMNAL ENCRPTER FUNCTION**

```
FUNCTION encryptMessage(msg): #DOING COLUMNAR transformation
   cipher <- ""
   # track key indices
   k_indx <- 0
   msg_len <- float(len(msg))
   msg_lst <- list(msg)
   key_lst <- sorted(list(key))
   # calculate column of the matrix
   col <- len(key)
   # calculate maximum row of the matrix
   row <- int(math.ceil(msg_len / col))
   fill_null <- int((row * col) - msg_len)
   msg_lst.extend('_' * fill_null)
   matrix <- [msg_lst[i: i + col]
           for i in range(0, len(msg_lst), col)]
           ENDFOR
   for _ in range(col):
      curr_idx <- key.index(key_lst[k_indx])
      cipher += ''.join([row[curr_idx]
                  for row in matrix])
                  ENDFOR k_indx += 1

ENDFOR, RETURN cipher ENDFUNCTION
```

**COLUMNAR DECRYPTER FUNTION**

```
FUNCTION decryptMessage(cipher): #reverse of columnar transformation

    msg <- ""
    k_indx <- 0
    msg_indx <- 0
    msg_len <- float(len(cipher))
    msg_lst <- list(cipher)
    col <- len(key)
    row <- int(math.ceil(msg_len / col))
    key_lst <- sorted(list(key))
    dec_cipher <- []
    for _ in range(row):
        dec_cipher += [[None] * col]
    ENDFOR
    for _ in range(col):
        curr_idx <- key.index(key_lst[k_indx])
        for j in range(row):
            dec_cipher[j][curr_idx] <- msg_lst[msg_indx]
            msg_indx += 1
        ENDFOR
        k_indx += 1
    ENDFOR
    try:
        msg <- ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")
    null_count <- msg.count('_')
    IF null_count > 0:
        RETURN msg[: -null_count]
    ENDIF
    RETURN msg
ENDFUNCTION
```

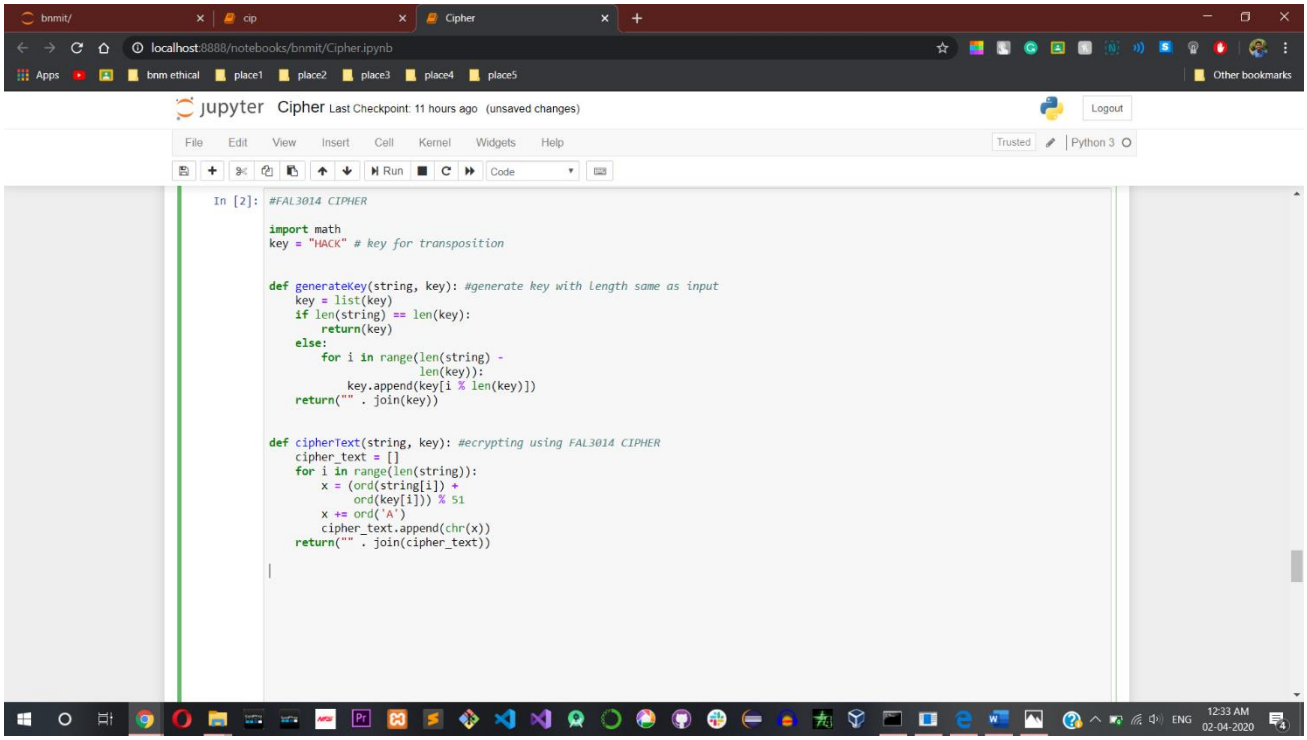
**FAL3014 DECRYPTER FUNTION**

```
FUNCTION originalText(cipher_text, key):  #reverse of FAL3014 CIPHER
    orig_text <- []
    for i in range(len(cipher_text)):
        x <- (ord(cipher_text[i]) -
             ord(key[i]) + 23) % 51
        x += ord('A')
        orig_text.append(chr(x))
    ENDFOR
    RETURN("" . join(orig_text))
ENDFUNCTION
```

**MAIN FUNCTION**

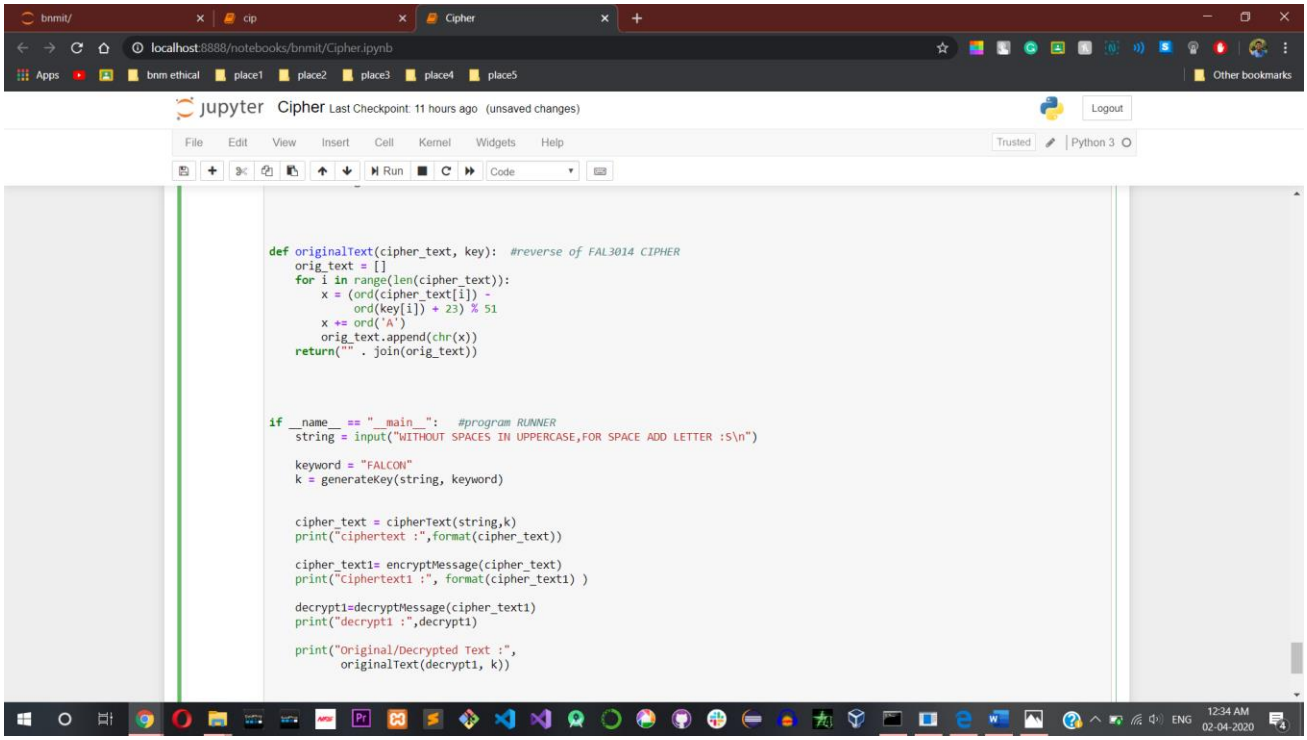
```
IF __name__ = "__main__":   #program RUNNER
   string <- input(" ENTER WITHOUT SPACES IN UPPERCASE,FOR SPACE ADD
LETTER :'S' (or) REMOVE EXTRA 'S' FROM THE FINAL OUTPUT \n")
   keyword <- "FALCON"
   k <- generateKey(string, keyword)
   cipher_text <- cipherText(string,k)
   OUTPUT "ciphertext :",format(cipher_text)
                    ENDFOR
   cipher_text1= encryptMessage(cipher_text)
   OUTPUT "Ciphertext1 :", format(cipher_text1)
                     ENDFOR
   decrypt1=decryptMessage(cipher_text1)
   OUTPUT "decrypt1 :",decrypt1
   OUTPUT "Original/Decrypted Text :",
         originalText(decrypt1, k))
```

## 1.3 Code Screenshots



```python
def encryptMessage(msg): #DOING COLUMNAR transformation
    cipher = ""

    # track key indices
    k_indx = 0

    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))

    # calculate column of the matrix
    col = len(key)

    # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))

    fill_null = int((row * col) - msg_len)
    msg_lst.extend('_' * fill_null)

    matrix = [msg_lst[i: i + col]
              for i in range(0, len(msg_lst), col)]

    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx]
                           for row in matrix])
        k_indx += 1

    return cipher
```

```python
def decryptMessage(cipher): #reverse of columnar transformation
    msg = ""

    k_indx = 0
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)


    col = len(key)
    row = int(math.ceil(msg_len / col))

    key_lst = sorted(list(key))
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]

    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])

        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1

    try:
        msg = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")
    null_count = msg.count('_')
    if null_count > 0:
        return msg[: -null_count]

    return msg
```




## 1.4 Output

ENTER WITHOUT SPACES IN UPPERCASE, FOR SPACE ADD LETTER:'S' (or) REMOVE EX TRA 'S' FROM THE FINAL OUTPUT
THIS IS SRIKRISHNA
ciphertext: BdpqJrAoGpsAseGfEj
Ciphertext1: drpejpAsG_BJGsEqoAf_
decrypt1: BdpqJrAoGpsAseGfEj
Original/Decrypted Text: THISSISSSRIKRISHNA

HERE,

S=SPACE
THE FINAL OUTPUT= THIS IS SRIKRISHNA