HOMOPHONIC SUBSTITUTION CIPHER

1. PROPOSED ALGORITHM:

- Get the Plaintext from User.
- Fetch the word character by character.
- Using RSA algorithm:
    - Generating P and Q
    - Pi(n) = (P-1) * (Q-1)
    - n = (P *Q)
    - e = 1<e< pi (n) in the between range.
    - d = (e * x) % n ==1 select d when it satisfies the condition.
- Encryption:
    - ((M^e) % n) /300 - Stores the quotient values.
    - ((M^e) % n) %300 - Stores the Remainder values.
    - Take the appropriate values of remainder and fetch the values present in the keyset according to index.
- Decryption:
    - (Quotient * 300) + remainder – Getting back the original number.
    - ((Y^d) % n) % 300
    - Get back the Plaintext by converting the ASCII values to characters and store in a list.
- Result:
    - Display the result which is stored in the list.

2. ENHANCEMENT:

In the Existing system , the plaintext will be replaced by numbers. Each letter of the plaintext will be taken separately and will check in the key set of appropriate characters and will replace it with the value in the keyset and will be treated as cipher text. It is language Dependent. Only Encrypt and decrypt the language which is programmed by the developer.

In the Proposed Method, we are going to use the RSA algorithm with minor changes for better security and it is language independent. Any language can encrypt its plaintext and decrypt its Cipher text. We have added a 300 set of characters mixed with alphabets and numbers, special characters.

4. CRYPTANALYSIS OF PROPOSED ALGORITHM:

Breaking Homophonic Substitution ciphers can be very difficult if the number of homophones is high. The usual Method is some sort of hill climbing, similar to that used in breaking substitution cipher. This Proposed algorithm has neglected the previous faults.

a. Strength of the Algorithm:

It is not easily breakable, as the encrypted text would be a combination of letters, special characters, and numbers. To be able to predict the result would be difficult because of the key set and the RSA algorithm which contains the public and private keys.

b. weakness of the Algorithm:

It is Language dependent and was quiet easy to break the cipher text. Only with the particular language can be able to encode an decode. Other language couldn’t able to do the process as it does not comes in their criteria.

c. Attacks possible:

If the encrypted text is modified then the receiver couldn’t be able to decode it exactly.

d. Attacks Impossible:

To be able to predict the keys would be difficult as it is created manually without any pattern and based the frequency distribution each and every character is having the keys. The length of the cipher text also may vary from the plaintext.

e. Benefits:

- ➢ Public key and private key will be changing every time Randomly.
- ➢ Keysets is made by developer.
- ➢ Simple and Efficient method.
- ➢ Couldn’t predict without the domain knowledge.

f. Drawbacks:

For larger set of lines and process it takes longer time, As a result time complexity may appear at some part. So it has to be made more efficient.

g. Computational Power:

The minimum requirements

PROCESSOR	: P4
RAM	: 2GB
WINDOWS	: windows 7
SOFTWARE	: Python

f. Limitations of existing System:

The existing system was quite predictable and language dependent so there wasn’t any kind of security,we have provide a bit more stronger one so the secrecy is made.

5. APPLICATIONS:

Confidential Messages

6. PROGRAM:

```
 mynewchar=['wQ','hW','kK','tT','Mi','rP','sG','Bb','kC','89',

'567','53','LQQ','Hj','CF','GhS','K0q','16','lBB','Jp',

'hM','hG','Jh','tNN','SKs','rTa','jSc','wKa','jGA','Hxz',

'kZ2','33s','99T','McB','KVSD','CD3','gq)','jGa','ee3','FA<',

'<F>','K;','*ES^','kG','hg2','mxZ','KCb','hjs','67w','113d',

'uws','fhd4','670','l<','hZ!','al)','lxZ','%C','%F','%S',

'V^','&F&','*D','0)','<,','>>','||','=!',';',':',

'*','-','(',')','09','88','77','76','73','=','x%',

'~~','g#','ds!','@!','^^','0&','33','22','z.','c%',

'µ','¶','·','¸','¹','º','»','¼','½','¾',

'¡','¢','£','¤','¥','¦','§','¨','©','ª',
          'a$','nm','--','[','+','.,','%^','b','K','v',

          'x','{','-','2','2x','+c','95','au','bL','w',

          '@f','-_',']','5','u','q','an','n,','l','z',

          'J','53','8','21','L','z.','ze','r','g','vx',

          '%a','43','ar','vk','mc','00','dk','fk','ak','sm',

          '«','32','hm','bc','m','ry','aa','k.','r1','y0'

          ':)',':(','fl','sk','sy','::','h^','(.)','st','mx',

          'nl','sz','ps','ma','{}','__','n$','pi','rd','az',

          'kl','br','sM','()','@$','=+','-=','1.','jd','jv',

          'cv','0A','i7','aa','§a',';;','v©','sc','bm','*%',

          'rm','1v','ad','¶¶','º3','}}','w§','!!','($','~f',

          'te','4E','es','kr','¨¨','[¤','07','f©','2¾','11',

          'xx','-1','sb','aw','67','¤]','150','44','1(','#4',

          'nu','-5.5','mr','jn','º2','d©','i7','3¾','9Z','Lq',

          'yy','C3','in','&^','¤;','pH','s©','bL','cK','dW',

          'rr','C#','nz','dg','¤:','º1','csk','mI','4¾','gL',

          'vr','p4','hz','§i','f@','mk','*5#','ry','hi','1¾',

          'cr','q9','sr','""','{{','ºy','§l','l2','ol','zx','ii',

          'LKA','FO0','56S','6DA','78CA','9Sa','89S','78&','&$0','0&d']


text=input("Enter a word:")

l=list(text)

list1=[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]

p=random.choice(list1)
```

```
q=random.choice(list1)

print(p,q)

n=(p*q)

pin=(p-1)*(q-1)



def gcd(a, b):

    while b != 0:

        a, b = b, a % b

    return a


def coprime(a, b):

    return gcd(a, b) == 1

for i in range(0,1000):

    e=random.randint(2,pin)

    if coprime(e,pin):

        print("yes",e,pin)

        break

print("success")


for j in range(0,100000):

        if ((e*j)%pin)==1:

            d=j

            break
```

```
print("d=",d)

l2=[]

l3=[]

l4=[]

#for g in range(len(l)):

#    print(ord(l[g]),end=" ")

#print()


##encryption

for k in range(len(l)):

    temp1=(ord(l[k]))

    temp2=((temp1**e)%n)%300

    temp3=((temp1**e)%n)//300

    #print("values:",temp2)

    l2.append(mynewchar[temp2])

    l4.append(temp3)


print("ENCRYPTION:")

for g in range(len(l2)):

    print(l2[g],end="")

print()


#print("q:")

#print(l4)
```

```
##decryption
for c in range(len(l2)):
    temp1=(mynewchar.index(l2[c]))
    #print("index:",temp1)
    temp3=((300*(l4[c]))+temp1)
    temp2=(temp3**d)%n
    l3.append(chr(temp2))
print("DECRYPTION:")
for g in range(len(l)):
    print(l3[g],end=" ")
print()
```

7. SAMPLE INPUT AND OUTPUT:

ENCRYPTION:

INPUT:

```
hello everyone?
```

OUTPUT:

```
.,q95578&99Tq91(q9hikC78&&^q956S
```

DECRYPTION:

INPUT:

```
.,q95578&99Tq91(q9hikC78&&^q956S
```

OUTPUT:

```
h e l l o   e v e r y o n e ?
```

8. PERFORMANCE ANALYSIS OF PROPOSED CRYPTOSYSTEM BY COMPARING WITH THE EXISTING SYSTEMS:

EXISTING SYSTEM	**PROPOSED SYSYTEM**
Language dependent: It only allows you to encrypt and decrypt on particular languages which has been assigned.. Eg: Enter a text:ஆ ENCRYPTION: [] DECRYPTION: [] 14.9721929292	**Language independent:** It can encrypt and decrypt most of the languages instead of blocking it. Eg: Enter a word: ஆ ENCRYPTION: *ES^yy DECRYPTION: ஆ 5.98858563187
Less Run time efficient: It is less effiecient Eg: Enter a text: fazeel ENCRYPTION: [10, 9, 2, 14, 16, 26] DECRYPTION: ['f', 'a', 'z', 'e', 'e', 'l'] Time: 3.30388870196	**More Run time efficient:** It is more efficient compared to existing system. Eg: Enter a word: fazeel ENCRYPTION: ee3rddg§l§ll2 DECRYPTION: f a z e e l Time : 3.02644477362

Easy to Decode: It is Easy to decode as it has the key set by comparing them we can able to perdict the desired output.	**Difficult to Decode:** It is quite difficult to decrypt as it has several steps such as RSA and the use of 300 key set ,so it will be much difficult to decode compared to existing method.

9. CONCLUSION:

By applying this algorithm you could able to generate different cipher text each time when we run the program. It is difficult to predict the expected output. It produces the public and private key randomly. So it makes the cipher text difficult to decrypt and also supports multiple language support ,we can encrypt the other language text too.

FUTURE WORKS:

To create a Graphical User Interface Application which implements the following proposed algorithm, With each keys randomly generated every time, through that the cipher text is generated.