CMPT 471 Assignment 4 – Isura Kumarapeli 301266292

Q1 – Design Document

This encryption scheme will work with any asci char from ASCII Dec 32 (which is the space key) to ASCII Dec 125 (which is }). This means all letters and all punctuation will work.

There are 6 key functions:

Function 1 - mapChar (char,offset) – this function runs the substitution encryption scheme, it takes a char and an offset value, and returns a char offset by that value. In my code it loops between ASCII 34 and ASCII 125.

Function 2 - transListChar (charList,transList) – This function takes a list of chars (a string converted to a list) and a transList (list of 16 (4*4) transpositions) and applies it to every 16 char chunk of the charList. It also adds ~ as extra padding to the transposed list to make sure it is a multiple of 16. In the end it returns an encrypted list of chars.

Function 3 - completeEncryption(inputString, scheme) – This Function takes an input string and an encryption scheme (a pair of an offset value and a TranspositionList) runs function 1 and function 2 and returns an encrypted string.

Function 4 - revTransListChar (charList,transList) – This takes an encrypted charlist and returns a charlist with the reverse transList function applied.

Function 5 - revMapChar (char,offset) – This takes a char and an offset and returns a char with the reverse offset applied.

Function 6 - completeDecryption(inputString, scheme) – This takes an encrypted input string and security scheme and returns the decrypted string. By running functions 4 and 5.

SAMEPLE SECURITY SCHEME

```
TransList1=[15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
TransList2=[1,3,5,7,9,11,13,15,14,12,10,8,6,4,2,0]
TransList3=[2,7,5,3,1,8,12,13,10,15,4,14,6,9,11,0]
TransList4=[0,2,4,6,8,10,12,14,1,3,5,7,9,11,13,15]
TransList5=[0,15,1,14,2,13,3,12,4,11,5,10,6,9,7,8]
OSList=[(32,TransList1),(0,TransList2),(100,TransList3),(42,TransList4),(1,TransList5)]
```

The program First shares a secret Key via Diffie-Hellman algorithm and uses that to decide which Security scheme to use.

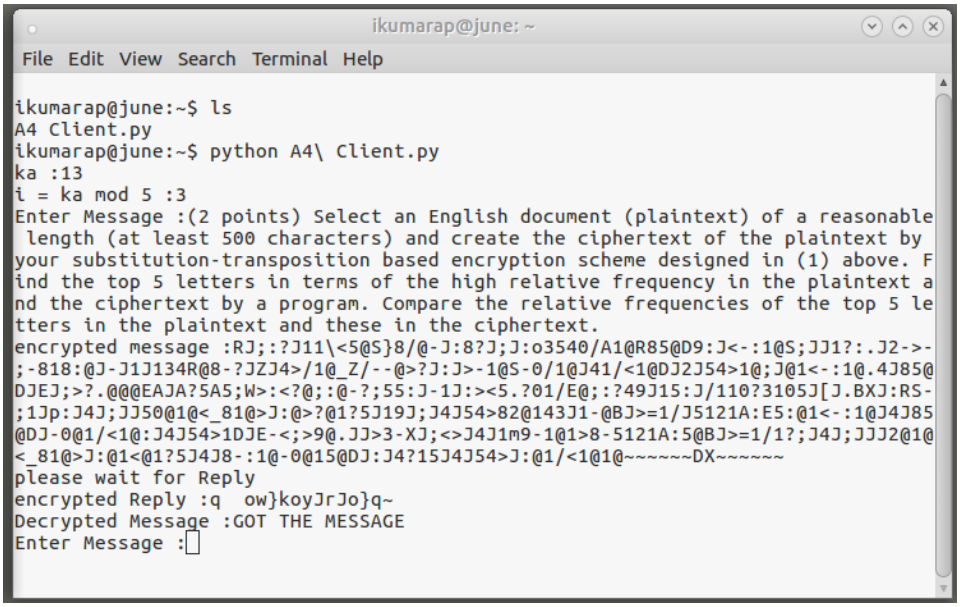
Q1 -TESTING

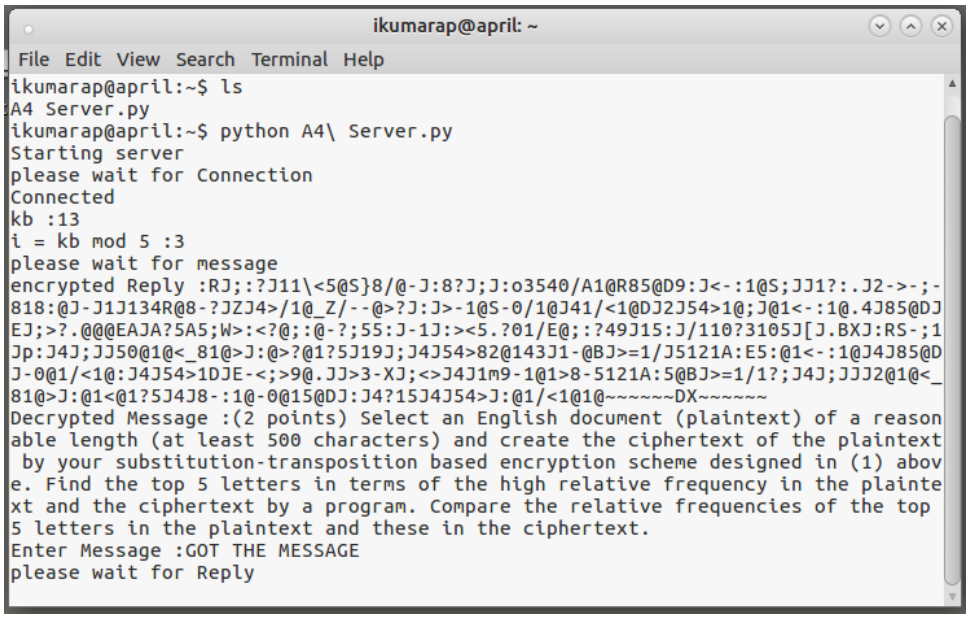
For testing I will send the following text from the assignment PDF

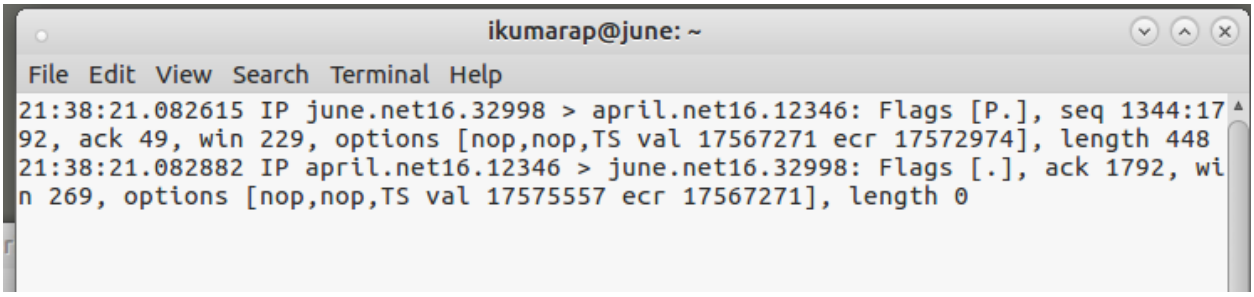
"(2 points) Select an English document (plaintext) of a reasonable length (at least 500 characters) and create the ciphertext of the plaintext by your substitution-transposition based encryption scheme designed in (1) above. Find the top 5 letters in terms of the high relative frequency in the plaintext and the ciphertext by a program. Compare the relative frequencies of the top 5 letters in the plaintext and these in the ciphertext."

****Note the text in the message must be or be between ascii value 34 and ASCII value 125.

Below shows the message from client June and server April (client initiates message but server can reply and keep messaging back and forth)







Captured packet of encrypted data from June to April by running "sudo tcpdump -i eth1 host 172.16.1.4" on host June when sending Previous message to April.

Q2 – Design Document

******All design documentation from Q1 also apply to this along with some extra steps.**

Like before the program first shares a secret key using the via Diffie-Hellman algorithm**.** The difference is this shared key is not used as a key for choosing the security scheme to use, but as a seed for a random number generator.

We then use that seed to generate a random number and use that number to choose a security scheme and encode the first chunk for 4 chars with that scheme. Then generate another random number and use that to encode the next for chunks and so on. On the other end, we decode the message using the same random number generator seed therefore the same scheme for each chunk of 4 chars.

Q1 -TESTING

For testing I will send the following text from the assignment PDF

"(2 points) Select an English document (plaintext) of a reasonable length (at least 500 characters) and create the ciphertext of the plaintext by your substitution-transposition based encryption scheme designed in (1) above. Find the top 5 letters in terms of the high relative frequency in the plaintext and the ciphertext by a program. Compare the relative frequencies of the top 5 letters in the plaintext and these in the ciphertext "

****Note the text in the message must be or be between ascii value 34 and ASCII value 125.

Below shows the message from client June and server April (client initiates message but server can reply and keep messaging back and forth)

```
[1]+  Stopped                 sudo tcpdump -i eth1 host 172.16.1.4
ikumarap@june:~$ sudo tcpdump -i eth1 host 172.16.1.4
[sudo] password for ikumarap:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
23:11:22.437282 IP june.net16.39596 > april.net16.12345: Flags [P.], seq 1131506
946:1131508690, ack 516443200, win 229, options [nop,nop,TS val 18962610 ecr 189
51434], length 1744
23:11:22.437725 IP april.net16.12345 > june.net16.39596: Flags [.], ack 1744, wi
n 281, options [nop,nop,TS val 18970895 ecr 18962610], length 0
23:11:27.446261 ARP, Request who-has june.net16 tell april.net16, length 46
23:11:27.446470 ARP, Reply june.net16 is-at 00:50:56:85:09:78 (oui Unknown), len
gth 28
```

```
ikumarap@june: ~
File  Edit  View  Search  Terminal  Help
Last login: Wed Mar 20 23:05:25 2019 from 192.168.0.101
ikumarap@june:~$ ls
A4 Client.py  A4 Client - Q2.py
ikumarap@june:~$ python A4\ Client\ -\ Q2.py
ka :9
i = ka mod 5 :4
Enter Message :(2 points) Select an English document (plaintext) of a reasonable
 length (at least 500 characters) and create the ciphertext of the plaintext by
your substitution-transposition based encryption scheme designed in (1) above. F
ind the top 5 letters in terms of the high relative frequency in the plaintext a
nd the ciphertext by a program. Compare the relative frequencies of the top 5 le
tters in the plaintext and these in the ciphertext.
encrypted message is :2p~~~~~~~~~~~~ (t~~zo~~~~~~~~~~u&~~Y/~~~~~~~~~~y~~~~~~~~~~
~~%'.'~~~~~~~~~~~~0#@6t~~mK~~~~~~~~~~&8?~~~~~~54~~~~~~u~~ij~~~~~~~~~~&v~n~f~o~~~
~~~~~~u~!~)~q~~~~~~~~~~~~~~~~~~~~~~0+#.~~~zk~~~~~~~~~~z~~~~~~~~~~~~(1@I!~b~!~s~~~
~~~~~~ao~~~~~~~~~~~~seo~b~c~m~~~~~~~~~18~~~~~~J1~~~~~~gh~~~~~~~~~~~~tn!~)~b~u~~~
~~~~~~~~~~~~~~~~~~#'.@~~~~~~~~~~~~U@650c~~~~~~~~~~~~ 0x~~gg~~~~~~~~~~ntr~~~~~~~~
~~~~ec~~~~~~~~~~~~~#@I5~~~~~~~~~~~~%@&0~~~~~~~~~~~~6#'4 h~~~~~~~~~~~~tef~!~d~j~~~
~~~~~~~~~~~~~~~~~~4'*2@D~~~~~~1@~~~~~~J2~~~~~~;J~~~~~~~~~~~~~~~~~~@'*6<-~~~~~~~85
~~~~~~~~~~~~~~~~~~:'60@.~~~~~~JE~~~~~~J;~~~~~~EA~~~~~~y~~{&~~~~~~~~~~xc~t~u~j~~~
~~~~~~u~v~u~j~~~~~~~~~nt~~~~~~~~~~~~o>:~~~~~~~-?~~~~~~~~~~~~~~~~~~+512in~~~~~~~~
~~~~otbs~~~~~~~~~~~~a 1J~~~~~~01~~~~~~cy~~~~~~~~~~~~rnto~~~~~~~~~~~~ip~~~~~~~~~~
~~%5@0~~~~~~~~~~~~~'/'*~~~~~~~~~~~~~5'&@~~~~~~~~~~~~~'0)+05~~~~~~J:~~~~~~7~~/.~~~~~
~~~~~&h~~ug~~~~~~~~~~&~~~~~~~~~~~~~@N'8p:~~~~~~50~~~~~~te~~~~~~~~~~~~h J;~~~~~~@<
~~~~~~~~~~~~~~~~~~.@U@1@~~~~~~@1~~~~~~~~~~~~~~~~~~+@54~~~~~~~~~~~~~'6@0y~~&s~~~~~
~~~~~xft~~~~~~~~~~~~ oeh~~~~~~~~~~~~ hn~~&m~~~~~~~~~~os~f~m~b~~~~~~~~~u~j~w~f~~~
~~~~~~~~~~~~~~~~~~'4(@=1~~~~~~A:~~~~~~yi~~~~~~~~~~~~ c~~~~~~~~~~~~*6@0v~~r&~~~~~
~~~~~k-:~~~~~~5@~~~~~~~~~~~~~~~~~~@6:'j~~&t~~~~~~~~~~gh ~~~~~~~~~~~~etv~~no~~~~~
~~~~~if~s~u~f~~~~~~~~~tb~~~~~~~~~~~~ xE-~~~~~~~JJ~~~~~~q~s~p~h~~~~~~~~~~s~~4g~~~~~
~~~~~xCm~~~~~~~~~~~~o x~~kg~~~~~~~~~~vJ4~~~~~~@1~~~~~~rl~~~~~~~~~~~~e ~~~~~~~~~~
~~8+6#~~~~~~~~~~~~~4(@'{~~kw~~~~~~~~~~k~~~~~~~~~~~~'+%0u~~l&~~~~~~~~~~y!~u~i~f~~~
~~~~~~u~~vz~~~~~~~~~~&&~~r;~~~~~~~~~~&te~~~~~~~~~~~~te>J~~~~~~?5~~~~~~:@~~~~~~J4
~~~~~~ l~~~~~~~~~~~~peit~~~~~~~~~~~~na~~~~~~~~~~~~@6:'j~~&t~~~~~~~~~~g~~~~~~~~~~
~~5'*6f~!~j~o~~~~~~~~~J4~~~~~~@1~~~~~~~~~~~~~~~~~~2+%@et~~~~~~~~~~~~rhx.~~~~~~~~
~~~~te
please wait for message
```

```
ikumarap@april: ~
File  Edit  View  Search  Terminal  Help
A4 Server.py  A4 Server - Q2.py
ikumarap@april:~$ python A4\ Server\ -\ Q2.py
Starting server
please wait for Connection
Connected
kb :9
i = kb mod 5 :4
seednum :66238
please wait for message
encrypted Reply is :2p~~~~~~~~~~~~ (t~~zo~~~~~~~~~~u&~~Y/~~~~~~~~~~y~~~~~~~~~~~~
%'.'~~~~~~~~~~~~0#@6t~~mK~~~~~~~~~~&8?~~~~~~54~~~~~~u~~ij~~~~~~~~~~&v~n~f~o~~~~~
~~~~u~!~)~q~~~~~~~~~~~~~~~~~~~~~~0+#.~~~zk~~~~~~~~~~z~~~~~~~~~~~~(1@I!~b~!~s~~~~~
~~~~ao~~~~~~~~~~~~seo~b~c~m~~~~~~~~~18~~~~~~J1~~~~~~gh~~~~~~~~~~~~tn!~)~b~u~~~~~
~~~~~~~~~~~~~~~~~~#'.@~~~~~~~~~~~~U@650c~~~~~~~~~~~~ 0x~~gg~~~~~~~~~~ntr~~~~~~~~~~
~~ec~~~~~~~~~~~~~#@I5~~~~~~~~~~~~%@&0~~~~~~~~~~~~6#'4 h~~~~~~~~~~~~tef~!~d~j~~~~~
~~~~~~~~~~~~~~~~~~4'*2@D~~~~~~1@~~~~~~J2~~~~~~;J~~~~~~~~~~~~~~~~~~@'*6<-~~~~~~~85~~
~~~~~~~~~~~~~~~~~~:'60@.~~~~~~JE~~~~~~J;~~~~~~EA~~~~~~y~~{&~~~~~~~~~~xc~t~u~j~~~~~
~~~~u~v~u~j~~~~~~~~~nt~~~~~~~~~~~~o>:~~~~~~~-?~~~~~~~~~~~~~~~~~~+512in~~~~~~~~~~
~~otbs~~~~~~~~~~~~a 1J~~~~~~01~~~~~~cy~~~~~~~~~~~~rnto~~~~~~~~~~~~ip~~~~~~~~~~~~
%5@0~~~~~~~~~~~~'/'*~~~~~~~~~~~~~5'&@~~~~~~~~~~~~'0)+05~~~~~~J:~~~~~~7~~/.~~~~~~~
~~~&h~~ug~~~~~~~~~~&~~~~~~~~~~~~@N'8p:~~~~~~50~~~~~~te~~~~~~~~~~~~h J;~~~~~~@<~~
~~~~~~~~~~~~~~~~~.@U@1@~~~~~~@1~~~~~~~~~~~~~~~~~~+@54~~~~~~~~~~~~'6@0y~~&s~~~~~~~
~~~xft~~~~~~~~~~~~ oeh~~~~~~~~~~~~ hn~~&m~~~~~~~~~~os~f~m~b~~~~~~~~~u~j~w~f~~~~~
~~~~~~~~~~~~~~~~~~'4(@=1~~~~~~A:~~~~~~yi~~~~~~~~~~~~ c~~~~~~~~~~~~*6@0v~~r&~~~~~~~
~~~k-:~~~~~~5@~~~~~~~~~~~~~~~~~~@6:'j~~&t~~~~~~~~~~gh ~~~~~~~~~~~~etv~~no~~~~~~~
~~~if~s~u~f~~~~~~~~~tb~~~~~~~~~~~~ xE-~~~~~~~JJ~~~~~~q~s~p~h~~~~~~~~~~s~~4g~~~~~~~
~~~xCm~~~~~~~~~~~~o x~~kg~~~~~~~~~~vJ4~~~~~~@1~~~~~~rl~~~~~~~~~~~~e ~~~~~~~~~~~~
8+6#~~~~~~~~~~~~~4(@'{~~kw~~~~~~~~~~k~~~~~~~~~~~~'+%0u~~l&~~~~~~~~~~y!~u~i~f~~~~~
~~~~u~~vz~~~~~~~~~~&&~~r;~~~~~~~~~~&te~~~~~~~~~~~~te>J~~~~~~?5~~~~~~:@~~~~~~J4~~
~~~~ l~~~~~~~~~~~~peit~~~~~~~~~~~~na~~~~~~~~~~~~@6:'j~~&t~~~~~~~~~~g~~~~~~~~~~~~
5'*6f~!~j~o~~~~~~~~~J4~~~~~~@1~~~~~~~~~~~~~~~~~~2+%@et~~~~~~~~~~~~rhx.~~~~~~~~~~
~~te
Decrypted Reply is :(2 points) Select an English document (plaintet) of a reason
able length (at least 500 characters) and create the ciphertext of the plaintext
 by your substitution-transposition based encryption scheme designed in (1) abov
e. Find the top 5 letters in terms of the high relative frequency in the plainte
xt and the ciphertext by a program. Compare the relative frequencies of the top
5 letters in the plaintext and these in the ciphertext.
Enter Message :
```

Q3

*****Message used for this portion is included below for reference.**

Letter Frequencies using Q1 Method ON LEFT using Q1 Method ON RIGHT

```
###letter frequency of message befor encryption
3560
max 1 Char :
max 1 Ratio:537/3560
max 2 Char :e
max 2 Count:414/3560
max 3 Char :t
max 3 Count:260/3560
max 4 Char :n
max 4 Count:239/3560
max 5 Char :s
max 5 Count:216/3560
###letter frequency of message befor encryption
###letter frequency of message After encryption
3568
max 1 Char :@
max 1 Ratio:537/3568
max 2 Char :'
max 2 Count:414/3568
max 3 Char :6
max 3 Count:260/3568
max 4 Char :0
max 4 Count:239/3568
max 5 Char :5
max 5 Count:216/3568
```

```
3560
max 1 Char :
max 1 Ratio:537/3560
max 2 Char :e
max 2 Count:414/3560
max 3 Char :t
max 3 Count:260/3560
max 4 Char :n
max 4 Count:239/3560
max 5 Char :s
max 5 Count:216/3560
14240
max 1 Char :~
max 1 Ratio:10682/14240
max 2 Char :&
max 2 Count:159/14240
max 3 Char :t
max 3 Count:157/14240
max 4 Char :@
max 4 Count:145/14240
max 5 Char :o
max 5 Count:120/14240
```

By comparing these two results, we can see that the first way of doing it is much less secure, as the frequencies are almost identical, (only difference is extra symbol added to ensure 16 chars per encryption) But looking at the second method the frequencies are not similar at all. There are lots of extra chars added due to extra chars to ensure each 4 chars has 16 chars to work with but ignoring the ~ symbol. We can still see there is no relationship between the top chars in the second scheme.

***text used for Q3 Calculations**

End-to-end encryption (E2EE) is a system of communication where only the communicating users can read the messages. In principle, it prevents potential eavesdroppers - including telecom providers, Internet providers, and even the provider of the communication service - from being able to access the cryptographic keys needed to decrypt the conversation.[1] In many messaging systems, including email and many chat networks, messages pass through intermediaries and are stored by a third party, from which they are retrieved by the recipient. Even if the messages are encrypted, they are typically only encrypted 'in transit', and are stored in decrypted form by the third party. This allows the third party to provide search and other features, or to scan for illegal and unacceptable content, but also means they can be read and misused by anyone who has access to the stored messages on the third party system, whether this is by design or via a backdoor. This can be seen as a concern in many cases where privacy is very important, such as persons living under repressive governments, whistleblowing, mass surveillance, businesses whose reputation depends on its ability to protect third party data, negotiations and communications that are important enough to have a risk of targeted 'hacking', and where sensitive subjects such as health, sexuality and information about minors are involved.End-to-end encryption is intended to prevent data being read or secretly modified, other than by the true sender and recipient(s). The messages are encrypted by the sender but the third party does not have a means to decrypt them, and stores them encrypted. The recipient retrieves the encrypted data and decrypts it themselves.Because no third parties can decipher the data being communicated or stored, for example, companies that use end-to-end encryption are unable to hand over texts of their customers' messages to the authorities.[2]Key exchange[edit]In an E2EE system, encryption keys must only be known to the communicating parties. To achieve this goal, E2EE systems can encrypt data using a pre-arranged string of symbols, called a pre-shared secret (PGP), or a one-time secret derived from such a pre-shared secret (DUKPT). They can also negotiate a secret key on the spot using Diffie-Hellman key exchange (OTR).[3]Modern usage[edit]As of 2016, typical server-based communications systems do not include end-to-end encryption. These systems can only guarantee the protection of communications between clients and servers, meaning that users have to trust the third parties who are running the servers with the original texts. End-to-end encryption is regarded as safer because it reduces the number of parties who might be able to interfere or break the encryption.[4] In the case of instant messaging, users may use a third-party client to implement an end-to-end encryption scheme over an otherwise non-E2EE protocol.[5]Some non-E2EE systems, such as Lavabit and Hushmail, have described themselves as offering "end-to-end" encryption when they did not.[6] Other systems, such as Telegram and Google Allo, have been criticized for not having end-to-end encryption, which they offer, enabled by default.[7][8]Some encrypted backup and file sharing services provide client-side encryption. The encryption they offer is here not referred to as end-to-end encryption, because the services are not meant for sharing messages between users. However, the term "end-to-end encryption" is often used as a synonym for client-side encryption.[citation needed]