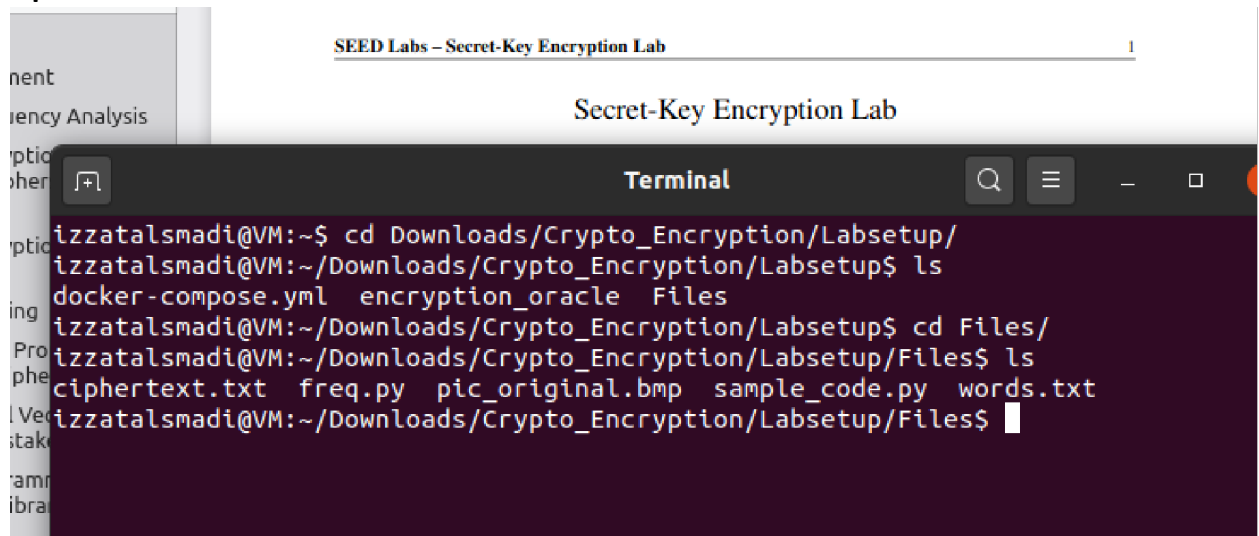


Quiz1

Secret Key Encryption Lab a sample walkthrough (without using the Docker container)

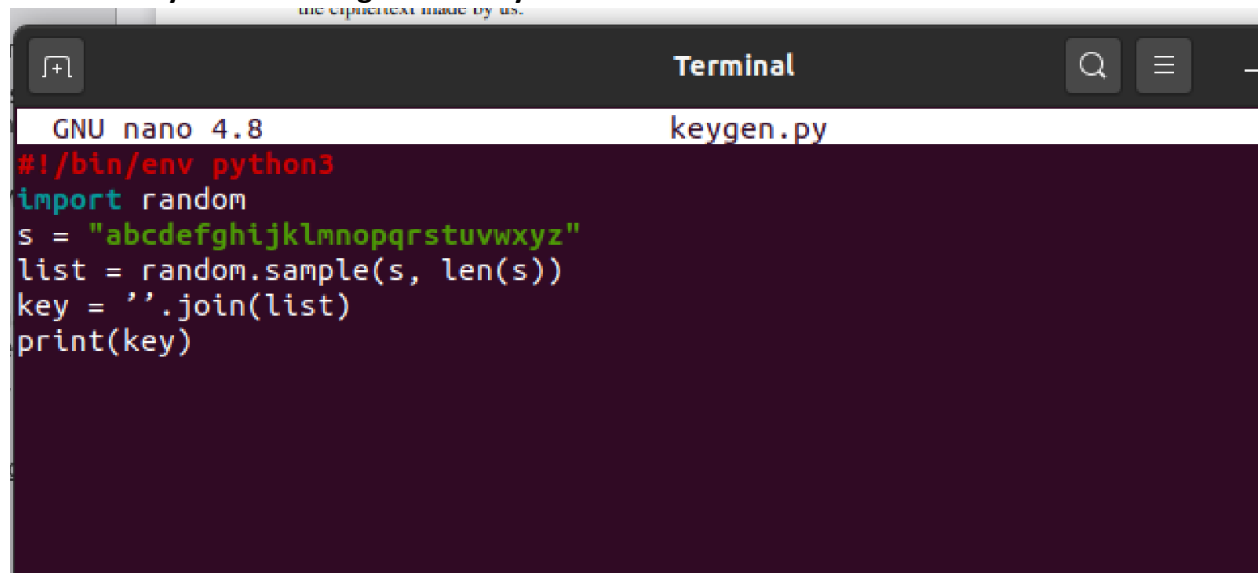
- Open content file from SEEDLAB folder



The screenshot shows a terminal window titled "Terminal" with a search icon and window controls. The user is navigating through a directory structure. The background shows a document titled "SEED Labs - Secret-Key Encryption Lab".

```
izzatalsmadi@VM:~$ cd Downloads/Crypto_Encryption/Labsetup/
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup$ ls
docker-compose.yml  encryption_oracle  Files
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup$ cd Files/
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ ls
ciphertext.txt  freq.py  pic_original.bmp  sample_code.py  words.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$
```

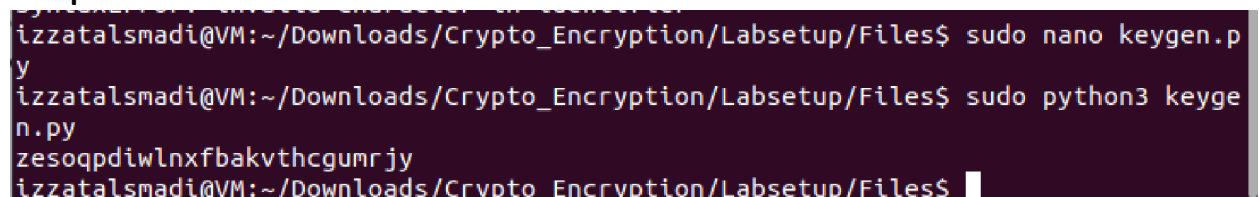
- Start with Python code to generate key



The screenshot shows a terminal window titled "Terminal" with a search icon and window controls. The user is editing a file named "keygen.py" using GNU nano 4.8. The background shows a document titled "the ciphertext made by us.".

```
GNU nano 4.8 keygen.py
#!/bin/env python3
import random
s = "abcdefghijklmnopqrstuvwxyz"
list = random.sample(s, len(s))
key = ''.join(list)
print(key)
```

- Compile and run



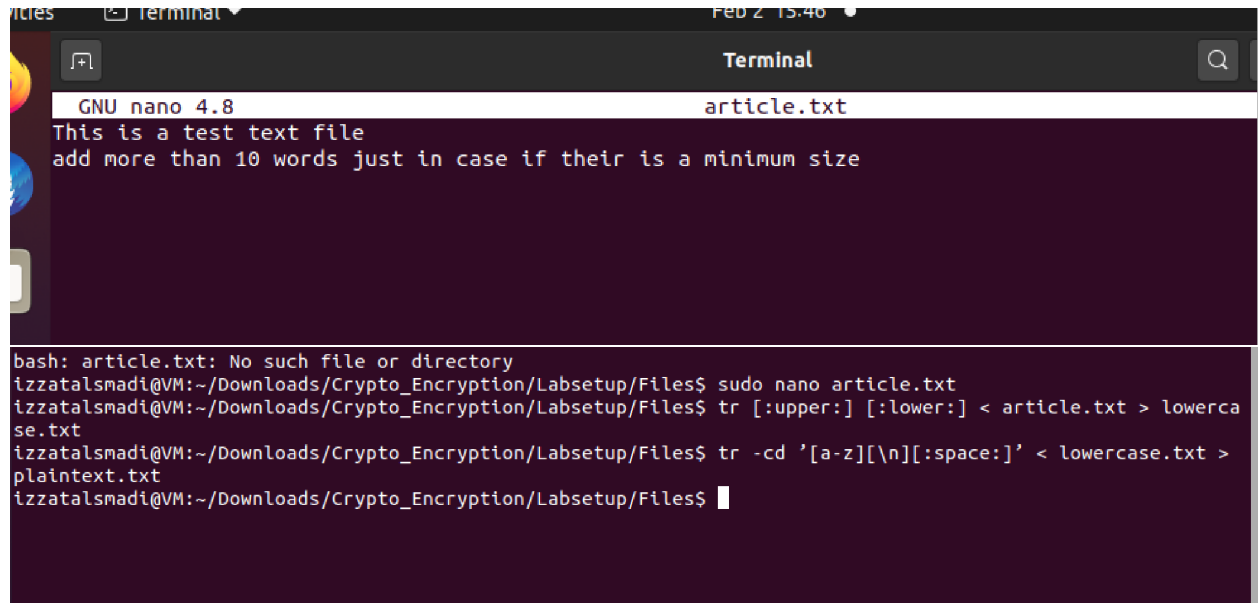
The screenshot shows a terminal window with the same user as the previous screenshots. The user is running the "keygen.py" script.

```
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ sudo nano keygen.p
y
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ sudo python3 keyge
n.py
zesopqdiwlxgbakvthcgumrjy
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$
```

- Step 2: let us do some simplification to the original article. We convert all upper cases to lower cases, and then removed all the punctuations and numbers. We do keep the

spaces between words, so you can still see the boundaries of the words in the ciphertext. In real encryption using monoalphabetic cipher, spaces will be removed. We keep the spaces to simplify the task. We did this using the following command

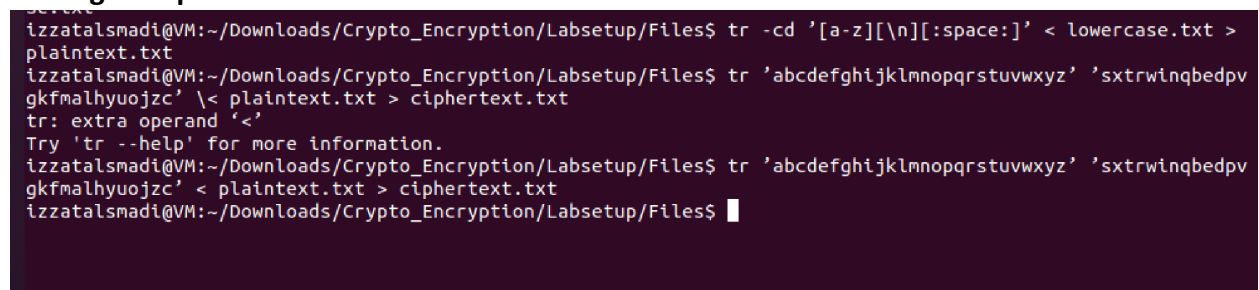
- Create a demo test article.txt file



```
GNU nano 4.8 article.txt
This is a test text file
add more than 10 words just in case if their is a minimum size

bash: article.txt: No such file or directory
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ sudo nano article.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr [:upper:] [:lower:] < article.txt > lowercase.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$
```

- Step 3: we use the tr command to do the encryption. We only encrypt letters, while leaving the space and return characters alone.



```
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'sxtrwinqbedpv gkfmalhyuojzc' < plaintext.txt > ciphertext.txt
tr: extra operand '<'
Try 'tr --help' for more information.
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'sxtrwinqbedpv gkfmalhyuojzc' < plaintext.txt > ciphertext.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$
```

- We have created a ciphertext using a different encryption key (not the one described above). It is included in Labsetup.zip file, which can be downloaded from the lab's website. Your job is to use the frequency analysis to figure out the encryption key and the original plaintext. We have also provided a Python program (freq.py) inside the Labsetup/Files folder. It reads the ciphertext.txt file, and produces the statistics for n-grams, including the single-letter frequencies, bigram frequencies (2-letter sequence), and trigram frequencies (3-letter sequence), etc.

```

izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ ./freq.py
./freq.py: command not found
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ ./freq.py
-----
1-gram (top 20):
b: 10
h: 8
l: 8
w: 7
s: 5
v: 4
q: 3
r: 3
a: 3
g: 3
i: 2
k: 2
y: 2
j: 1
p: 1
o: 1
e: 1
t: 1
c: 1
-----
2-gram (top 20):
hq: 3
bl: 3

```

- For example, in the following, we replace letters a,e, and t in in.txt with letters X,G,E, respectively; the results are saved inout.txt.

```

ylh: 1
tsl: 1
slw: 1
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr 'aet' 'XGE' < in.txt > out.txt
bash: in.txt: No such file or directory
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ sudo nano in.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ tr 'aet' 'XGE' < in.txt > out.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$

```

Task 2: Encryption using Different Ciphers and Modes

- In this task, we will play with various encryption algorithms and modes. You can use the following openssl enc command to encrypt/decrypt a file. To see the manuals, you can type man openssl and man enc.

```

izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ openssl enc -aes-128-cbc -e -in article.txt -o
ut cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$

```

- Check the new file

```

izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$ ls
article.txt  ciphertext.txt  in.txt        lowercase.txt  pic_original.bmp  sample_code.py
cipher.bin   freq.py        keygen.py     out.txt       plaintext.txt     words.txt
izzatalsmadi@VM:~/Downloads/Crypto_Encryption/Labsetup/Files$

```

Task 3: Encryption Mode – ECB vs. CBC

- Continue to the rest of the tasks