

Secret Key Encryption Lab Report

Hayden Eubanks

School of Business, Liberty University

CSIS 463-B01

Dr. De Queiroz

September 24, 2023

Secret Key Encryption Lab Report

Introduction:

When implementing encryption algorithms, it is often that varying encryption modes can be chosen to modify the implementation of the encryption process (IBM, 2023). These encryption modes can be seen to modify aspects of the encryption algorithm such as the order in which operations are performed, the block size, the length of the encryption key, and the introduction of pseudorandom modification values (Sathya, Premalatha, & Rajasekar, 2021). Each of these encryption modes represents a vastly different implementation and each of these implementations will have use cases where they are most relevant (Geeks for Geeks, 2023). However, the improper configuration of encryption algorithm modes can have severe implications on the effectiveness of data confidentiality resulting in weak data security (Mitre, 2023). The importance of addressing this vulnerability is then further emphasized through the fact that a developer may not realize they have improperly configured the encryption leading to a vulnerability where a strong security implementation is perceived to exist. Fortunately, this security concern can be mitigated through the proper implementation of encryption algorithm modes relevant to the desired use case, and as such, a security professional needs to understand the implications of algorithm security modes (Basta, 2018). Through studying the nuance between encryption algorithm implementations, such as the Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback Mode (CFB), and Output Feedback Mode (OFB) a security professional can be better equipped to implement encryption algorithms and apply them to adequately provide data security (NIST, 2023).

The encryption mode chosen for the implementation of an encryption algorithm can have drastic implications on data security (Ametepe et al., 2022) and as such, this lab seeks to

demonstrate vulnerabilities associated with improper implementations. The first area of concern addressed by the lab relates to the ability for statistical analysis to be performed on data encoded one character at a time. This trait is the primary characteristic of substitution ciphers and by examining the application of computational statistical analysis the need for stronger encryption practices can be emphasized (Basta, 2018). With the need for stronger encryption practices identified, the lab then turns to the examination of varying encryption modes and how they differ in the encoding process. To accomplish this examination, the AES encryption algorithm was used, and the differing encryption modes of AES were compared. AES contains encryption modes that encode various-sized blocks of data in addition to introducing various degrees of randomness which contribute to the security of data given different application use cases (Ametepe et al., 2022). Studying the differences between these encryption modes then allows for a greater understanding of their applications and weaknesses to be understood enabling their proper usage to accomplish encryption objectives.

In examining the proper implementation of security modes, several use cases can be examined that highlight each property of the encryption algorithm modified by a usage mode. The first property to be examined in this lab is the need for randomness in achieving certain security objectives such as the encryption of images. In images, each pixel is encoded with RGB color values by which encryption can be performed. However, an encryption mode that produces the same mapping for a given input value will not accomplish the objectives of encrypting an image as the like colors of an image will result in a change in color value but will not obscure the image as a whole (NIST, 2023). This fact then highlights the importance of randomness in accomplishing certain security objectives through encryption (Mitre, 2023). Following the examination of randomness, the effect of an encryption mode on data padding can then be

observed. When certain block encryption algorithms encrypt a message, they must first pad out the input message to a size divisible by the block size (Geeks for Geeks, 2023). This padding is essential to the operation of block encryption algorithms and must be understood by a security professional to track the encryption process effectively. With this, an additional property of block encryption ciphers is that, as their name suggests, they encrypt whole blocks at a time. This means that an error of one bit could corrupt the data of just a bit, an entire block, or more depending on the encryption mode used (NIST, 2023). This fact highlights the importance for a security professional to understand the implications of encryption modes on environments prone to data corruption and can allow them to choose an encryption mode best suited for their encryption task.

The final portion of the lab then examines initialization vectors (IVs) which are implemented to introduce randomness to successive encryptions of the same value to better conceal encrypted messages (Sathya, Premalatha, & Rajasekar, 2021). The introduction of IVs can greatly increase the security of encrypted data as the randomness introduced makes it difficult for an observer to recognize patterns in encrypted values (NIST, 2023). Further, IVs solve the previously mentioned problem of encrypting images as pixels with the same color value can be encrypted to different output values resulting in an encrypted image that appears to represent completely random data. However, the introduction of IVs alone does not resolve these vulnerability issues as the improper application of IVs continues to result in weakened security (Sathya, Premalatha, & Rajasekar, 2021). To address these issues, IV properties are explored through the lab and best practices are introduced such as not reusing IV values and ensuring future IV values are not predictable. The proper implementation of IVs can then be seen to

protect data through an added layer of randomness and prevent eavesdroppers from detecting patterns in encryption.

With these features in mind, it is then possible to see the importance of appropriately selected encryption modes and parameters. Encryption modes can have massive implications on data security given a particular use case and for this reason, it is essential that a security professional understand encryption use modes and applications (Mitre, 2023). Through studying encryption algorithm implementations such as varying modes of the AES encryption algorithm, a better understanding of encryption parameters can be achieved and the security professional can be better equipped to apply strong applications of encryption modes (Ametepe et al., 2022). Encryption modes are an important aspect of the encryption process and through their proper implementation, the security objectives of cryptography can be furthered in the work of a security professional (Basta, 2018).

Lab Procedure:

Before the steps of this lab can be completed, there is first some pre-setup that must be configured within the Linux environment. To begin the lab the user first needed to use docker to install the containers that will be used to construct the lab environment ([Screenshot1](#)). After this, the containers could be constructed into the lab environment and set up to run in the background when needed for a later task ([Screenshot2](#)). With the environment constructed, a new shell could then be opened allowing the user to approach the lab tasks.

The first objective of this lab challenged the user with the task of utilizing Python code to encrypt a text using a monoalphabetic cipher before then breaking another monoalphabetic encrypted text using statistical analysis. To perform monoalphabetic encryption, a Python program was provided and could be installed on the Linux system to produce a random ordering of the English characters ([Screenshot3](#)). Executing this python script then produces a random ordering of characters ([Screenshot4](#)) and this ordering is the ordering that will be used as the key for the monoalphabetic encryption. To perform this encryption a file was made containing plaintext ([Screenshot5](#)) and then stripped of uppercase letters, spaces, and punctuation to make the encrypted text more difficult to decipher ([Screenshot6](#), [Screenshot7](#)). After this, the previously generated key mapping can be applied to the text of this file ([Screenshot8](#)) to create an encrypted message ([Screenshot9](#)).

After gaining exposure to the creation of monoalphabetic ciphers, the user is then presented with a file of text encrypted through a monoalphabetic mapping ([Screenshot10](#)). This then presents a great example of a use case through which statistical analysis can be applied to predict characters based on the commonality of their appearance throughout the English language (University of Notre Dame, n.d.). A Python script can then be used to present the

frequency of characters appearing as well as the frequency of bigrams and trigrams of letters ([Screenshot11](#), [Screenshot12](#)). By comparing these characters to their respective counterparts in the list of most commonly used English characters, best-guess substitutions can be made (University of Notre Dame, n.d.) such as replacing the letters ‘ytn’ with ‘THE’ as the most common trigram ([Screenshot13](#)). Checking this substitution with the context of the message appears to make sense ([Screenshot14](#)), so further substitutions could be made and verified by this pattern ([Screenshot15](#)) until the message has been decrypted ([Screenshot16](#)).

With an understanding of the security implications of monoalphabetic substitution established, the lab then transitioned toward an examination of encryption security modes to identify the advantages and disadvantages of encryption modes. The OpenSSL command was used to test the syntax and output of encryption files and with this, the descriptions of the encryption modes could be examined through the use of the manual pages ([Screenshot17](#), [Screenshot18](#)). After examining the different options available, a new text file was created ([Screenshot19](#)) and then encrypted through the use of the AES 128-bit Cipher Block Chaining (CBC) mode ([Screenshot20](#)), the AES 128-bit Cipher Feedback Mode (CFB) ([Screenshot22](#)), and Triple DES (DES3) ([Screenshot24](#)) algorithm modes. Each of their respective outputs could then be observed and their outputs compared ([Screenshot21](#), [Screenshot23](#), [Screenshot25](#)).

With a basic understanding of encryption modes established, a further examination of the implications of security modes on data security could then be approached. The next task approached this topic by giving the user a BMP image file ([Screenshot26](#)) and instructing the user to encrypt the file through both the Electronic Code Book (ECB) and CBC encryption modes of AES. The first encryption mode used was ECB ([Screenshot27](#)) with the distinguishing characteristic of ECB being the one-to-one encoding of input to output (Geeks for Geeks, 2023).

To view the encrypted image, the header from the original must be replaced into the encrypted image ([Screenshot28](#)), and then by viewing the image a blurred but still distinguishable version of the image can be seen ([Screenshot29](#)). The recognizable nature of this encrypted message is due to the one-to-one encoding of like colors (Mitre, 2023) and highlights a security vulnerability in using this encryption mode for image encryption. By repeating this experiment using the CBC encryption mode ([Screenshot30](#), [Screenshot31](#)) and viewing the image ([Screenshot32](#)) it can be seen that the image appears completely unrecognizable. The CBC encryption mode introduces an element of randomness to the encryption process (NIST, 2023) and this randomness then allows pixels of the same value to encode to different outputs. To emphasize the relevancy of this application vulnerability, a new photo was chosen being an image of myself ([Screenshot33](#)). Encrypting this message using the same processes ([Screenshot34](#) – [Screenshot37](#)) then reveals similar results highlighting the importance of choosing an encryption mode to protect personally identifiable data.

One additional difference between encryption modes that must be understood by a security professional is when padding is added to an encrypted value. In order to explore this concept, the lab instructed the user to test four different encryption modes and compare the output values to see if padded characters had been added. The encryption modes used for this test were ECB, CBC, CFB, and the Output Feedback Mode (OFB). To test for padding, a file was created ([Screenshot38](#)) and then tested with each of the respective outputs ([Screenshot39](#) – [Screenshot42](#)). Examining the output, it can then be seen that all of the modes except for ECB pad the hex string with extra characters. This is because the remaining modes process data in blocks and as such, require an input size divisible by the block size (Basta, 2018). The block size can then be further tested by creating files of 5, 10, and 16 bytes ([Screenshot43](#)) and then

encrypting the files using the different encryption modes used before ([Screenshot44](#)). Examining the output then reveals that the file size increased when a block size was not readily divisible by the existing file size ([Screenshot45](#)). Viewing the encrypted files then reveals the padded characters and their hex values ([Screenshot46](#) – [Screenshot50](#)).

The fact that some of these encryption modes process data in blocks then has further implications on data corruption that must be examined. In the lab, this was tested by creating a large data file ([Screenshot51](#) – [Screenshot53](#)) and then corrupting a single bit of that file ([Screenshot54](#)). By then encrypting the file using the varying AES encryption modes ([Screenshot55](#)) and viewing the output ([Screenshot56](#) – [Screenshot59](#)) it can be seen that 16 bytes, or one block of data, was corrupted in each of the output files except for that of the ECB encryption mode which only had a single character corrupted. This then highlights another security flaw of the ECB encryption mode which is that a single bit flipped in the input data will be present in the output allowing an observer to glean information about the encryption process through bit modification (Mitre, 2023).

With a firm understanding of the basics of encryption modes held, an examination of initialization vectors (IVs) can then be performed to understand the need for randomness with encryption algorithms. When encryption is performed without an element of randomness, the same input value will continue to map to the same output value (Mitre, 2023). This could have severe security implications as an outside observer could recognize patterns in encrypted values and then use those values to attempt to break the encryption or perform ciphertext-only attacks (Mitre, 2023). Additionally, this issue results in the lack of security in applications such as image encryption previously demonstrated. For these reasons, it is advantageous to add an element of randomness into the encryption process which is what is referred to as an initialization vector

(NIST, 2023). The exclusive or (XOR) operation is then performed on IVs and the encrypted value to produce an output value that is a function of the IV, key, and encryption mode.

However, initialization vectors must also follow best practices in their implementation to ensure the randomness is effective and meets security objectives. This factor of IVs is extremely important as the IVs must be transported with the encrypted message to allow the receiver to decrypt it (Mitre, 2023). Some encryption best practices include only using each IV with a given key once and ensuring IVs are not predictable. The lab tasks then introduce IV applications as well as guide the user through a process of testing the security concerns regarding IV use.

The first of the best practices explored in the lab is the property of uniqueness that must be held by each IV regarding its associated key (Mitre, 2023). By creating a message ([Screenshot60](#)) and then encrypting that message using the same IV ([Screenshot61](#)) it can be seen that both encryptions produce the same ciphertext value ([Screenshot62](#), [Screenshot63](#)). In contrast, producing an encryption with a different IV then produces a completely different output value ([Screenshot64](#)). Following this, the lab provided the user with a known plaintext and its associated ciphertext value. Having another ciphertext value under the assumption that the same IV was used provides the user with the ability to decipher the encrypted message using the commutative property of the XOR operation (NIST, 2023). To perform these XOR operations, the lab provided the user with a Python program ([Screenshot65](#)), and by modifying this program to hold the values given by the lab ([Screenshot66](#)) the plaintext value can be observed ([Screenshot67](#)).

The lab then further challenges the user with a scenario where the IV value changes with every message, but a predictable pattern exists between the IVs. This then highlights the next important feature of IV security which is that IVs must not be predictable (Mitre, 2023). As it is

given that the sent message will be either 'Yes' or 'No', these values were first converted to hexadecimal so they would be more readily recognizable ([Screenshot68](#)). Following this, a series of message exchanges were performed between the user and the system to identify the pattern between the IV values ([Screenshot69](#)). With this, in mind, it can be seen that several aspects of the problem are known to the user. First, the cyphertext and plaintext values of the initial message are known as the initial message was either 'Yes' or 'No', and the ciphertext value was given directly. Additionally, the IV value is given so the element of randomness is also known. Through testing values of either 'Yes' or 'No' with the newly generated IVs the user should then be able to XOR the values together and compare the values to get a matching value confirming which plaintext value was used. Additionally, the predictable nature of the IVs should allow for the deciphering of future encrypted messages using the same IV pattern highlighting the severity of this security concern. In my application of this lab step I created a program to attempt to XOR the values of each generated value pair ([Screenshot70](#)) but I was unable to definitively determine the plaintext input used ([Screenshot71](#)). However, despite this the value of generating IVs in such a way that the pattern cannot be predicted is clear, and as such it is essential that a security professional apply this best practice.

Encryption modes are an essential aspect of the encryption process, and by examining encryption modes and their functionality a security professional can be better equipped to apply encryption while avoiding the risks associated with misconfiguration (NIST, 2023). Additionally, the study of encryption modes can assist a security professional in their knowledge of encryption algorithm functionality in areas such as block cipher padding and the addition of randomness into the encryption process. This understanding can then enable a security professional to apply encryption in a way that effectively protects data and does not fall prey to

misconfiguration errors. Encryption is a crucial aspect of modern data security and through understanding encryption modes, a security professional can ensure that the security objectives of confidentiality and integrity are upheld in their work (Basta, 2018).

References

- Ametepe, A. F., Ahouandjinou, Arnaud S. R. M., & Ezin, E. C. (2022). Robust encryption method based on AES-CBC using elliptic curves Diffie–Hellman to secure data in wireless sensor networks. *Wireless Networks*, 28(3), 991-1001. <https://doi.org/10.1007/s11276-022-02903-3>
- Basta, A. (2018). *Oriyano, cryptography: Infosec pro guide*. McGraw-Hill Education. <https://bookshelf.vitalsource.com/reader/books/9781307297003/pageid/14>
- Geeks for Geeks. (May 9, 2023). *Block cipher modes of operation*. Geeks for Geeks. <https://www.geeksforgeeks.org/block-cipher-modes-of-operation/>
- IBM. (July 31, 2023). *Encryption ciphers and modes*. IBM. <https://www.ibm.com/docs/en/informix-servers/12.10?topic=encryption-ciphers-modes>
- NIST. (April 28, 2023). *NIST SP 800-38A*. NIST. <https://doi.org/10.6028/NIST.SP.800-38A>
- Mitre. (June 29, 2023). *CWE-1204: Generation of weak initialization vector (IV)*. Common Weakness Enumeration. <https://cwe.mitre.org/data/definitions/1204.html>
- Sathya, K., Premalatha, J., & Rajasekar, V. (2021). Investigation of strength and security of pseudo random number generators. IOP Conference Series. *Materials Science and Engineering*, 1055(1), 12076. <https://doi.org/10.1088/1757-899X/1055/1/012076>
- University of Notre Dame. (n.d.). *Frequency of letters of the alphabet in English*. University of Notre Dame. <https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html>

Screenshots: Task 1

Screenshot1: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup
parallels@ubuntu-linux-22-04-desktop:~/Labsetup$ docker-compose up
[+] Running 3/3
! oracle-server Warning
[+] Building 1160.8s (12/13)

=> [internal] load build definition from Dockerfile
0.0s
=> transferring dockerfile: 324B
0.0s
=> [internal] load .dockerignore
0.0s
=> transferring context: 2B
0.0s
=> [internal] load metadata for docker.io/handsonsecurity/seed-ubuntu:small
9.3s
=> [internal] load metadata for docker.io/handsonsecurity/seed-ubuntu:dev
8.8s
=> [internal] load build context
0.0s
=> transferring context: 0.68kB
0.0s
=> [builder 1/4] FROM docker.io/handsonsecurity/seed-ubuntu:dev@sha256:f30e4224cf90ab83686285e4e1b12ef3879d3f6ec24aee991c08aeb52295551
1145.0s
=> resolve docker.io/handsonsecurity/seed-ubuntu:dev@sha256:f30e4224cf90ab83686285e4e1b12ef3879d3f6ec24aee991c08aeb52295551
0.0s
=> sha256:f30e4224cf90ab83686285e4e1b12ef3879d3f6ec24aee991c08aeb52295551 1.99kB / 1.99kB
0.0s
=> sha256:da7391352a9bb76b292a568c066aa4c3bae8d494e6a3c68e3c596d34f7c75f8 28.56MB / 28.56MB
932.2s
=> sha256:14428a6d4bcd8a49a64127900a0691fb00a3f329aced25eb77e3b65646638f8d 847B / 847B
0.3s
=> sha256:89212aee292bf847e1eaaa02d16351a8a506e1e8c4dbab312f80b5e42c1b92a 5.16kB / 5.16kB
0.0s
=> sha256:2c2d948710f21ad82dce71743b1654b45ac85c059cf5c19da491582cef6f2601 162B / 162B
0.5s
=> sha256:5d39dfbe33099184bf2060692a46576a593230ba5eacc592d42ccd3647e737 12.40MB / 12.40MB
48.1s
=> sha256:56b236c9d9da5d926b5038e061e081554f06305fdb3c2d9fda71de4a4e8d996 4.58kB / 4.58kB
1.7s
=> sha256:1bb168ce59cc24a80017d0af97dd32ad127952731a07cddfaea135e4ffa290bd 247B / 247B
2.2s
=> sha256:588b6963c007acbc8bdc581cb77f921bec7c65a77fd7b32e19010c66f73094a 219B / 219B
2.6s
=> sha256:c3c83e840346aaa100c5711253d4df0b40bdc4f38f00fd1c804701c26a3ef36f 87.86MB / 87.86MB
1143.0s
=> extracting sha256:da7391352a9bb76b292a568c066aa4c3bae8d494e6a3c68e3c596d34f7c75f8
218.5s
=> extracting sha256:14428a6d4bcd8a49a64127900a0691fb00a3f329aced25eb77e3b65646638f8d

```

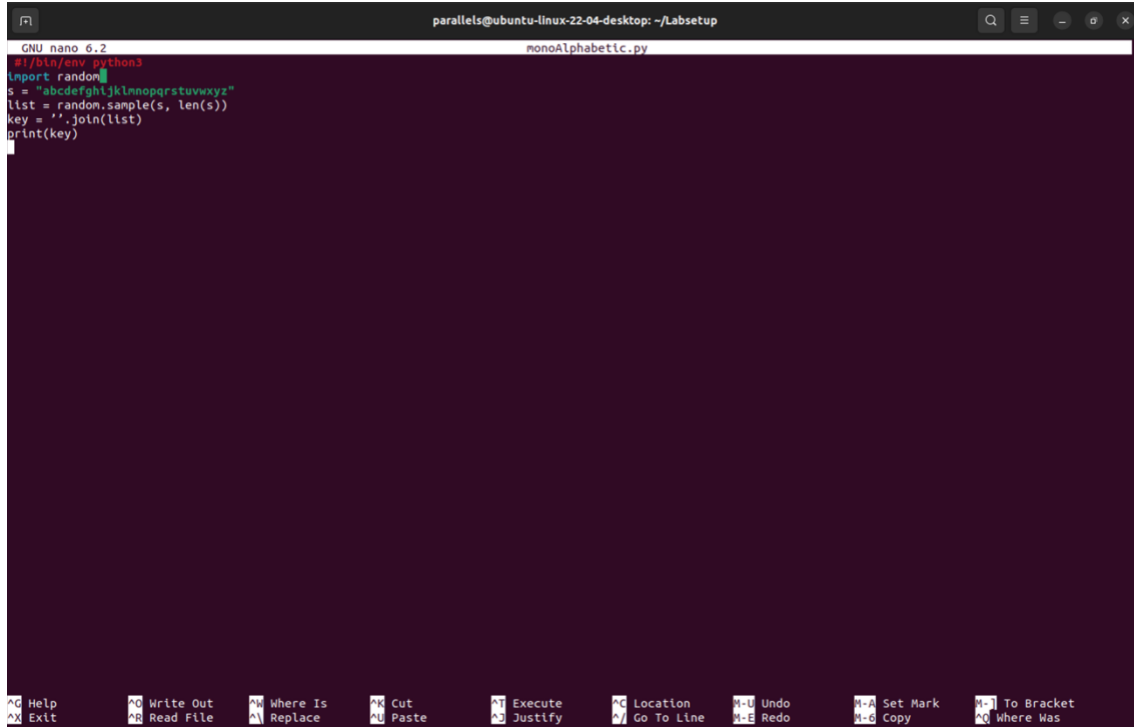
Screenshot2: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup
0.0s
=> sha256:14428a6d4bcd8a49a64127900a0691fb00a3f329aced25eb77e3b65646638f8d 847B / 847B
0.3s
=> sha256:da7391352a9bb76b292a568c066aa4c3bae8d494e6a3c68e3c596d34f7c75f8 28.31MB / 28.56MB
1151.5s
=> sha256:2c2d948710f21ad82dce71743b1654b45ac85c059cf5c19da491582cef6f2601 162B / 162B
0.5s
=> sha256:5d39dfbe33099184bf2060692a46576a593230ba5eacc592d42ccd3647e737 12.40MB / 12.40MB
48.1s
=> sha256:56b236c9d9da5d926b5038e061e081554f06305fdb3c2d9fda71de4a4e8d996 4.58kB / 4.58kB
1.7s
=> sha256:1bb168ce59cc24a80017d0af97dd32ad127952731a07cddfaea135e4ffa290bd 247B / 247B
2.2s
=> sha256:588b6963c007acbc8bdc581cb77f921bec7c65a77fd7b32e19010c66f73094a 219B / 219B
2.6s
=> extracting sha256:da7391352a9bb76b292a568c066aa4c3bae8d494e6a3c68e3c596d34f7c75f8
0.6s
=> extracting sha256:5d39dfbe33099184bf2060692a46576a593230ba5eacc592d42ccd3647e737
0.3s
=> [builder 2/4] COPY . /oracle
0.1s
=> [builder 3/4] WORKDIR /oracle
0.0s
=> [builder 4/4] RUN make
0.1s
=> [stage-1 2/3] COPY --from=builder /oracle/build /oracle
0.0s
=> [stage-1 3/3] WORKDIR /oracle
0.0s
=> exporting to image
0.0s
=> exporting layers
0.0s
=> writing image sha256:b880623009a32aab48633bfedd296cfea788b77b928aa92887d40acf4d6727
0.0s
=> naming to docker.io/library/seed-image-encryption
0.0s
[+] Running 3/2
✔ Network net-10.9.0.0
Created 0.1s tainer oracle-10.9.0.80 Creating
0.1s
✔ Container oracle-10.9.0.80
Created 0.1s
! oracle-server The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requeste
d
0.0s
Attaching to oracle-10.9.0.80
oracle-10.9.0.80 | Server listening on 3000 for known_lv

```

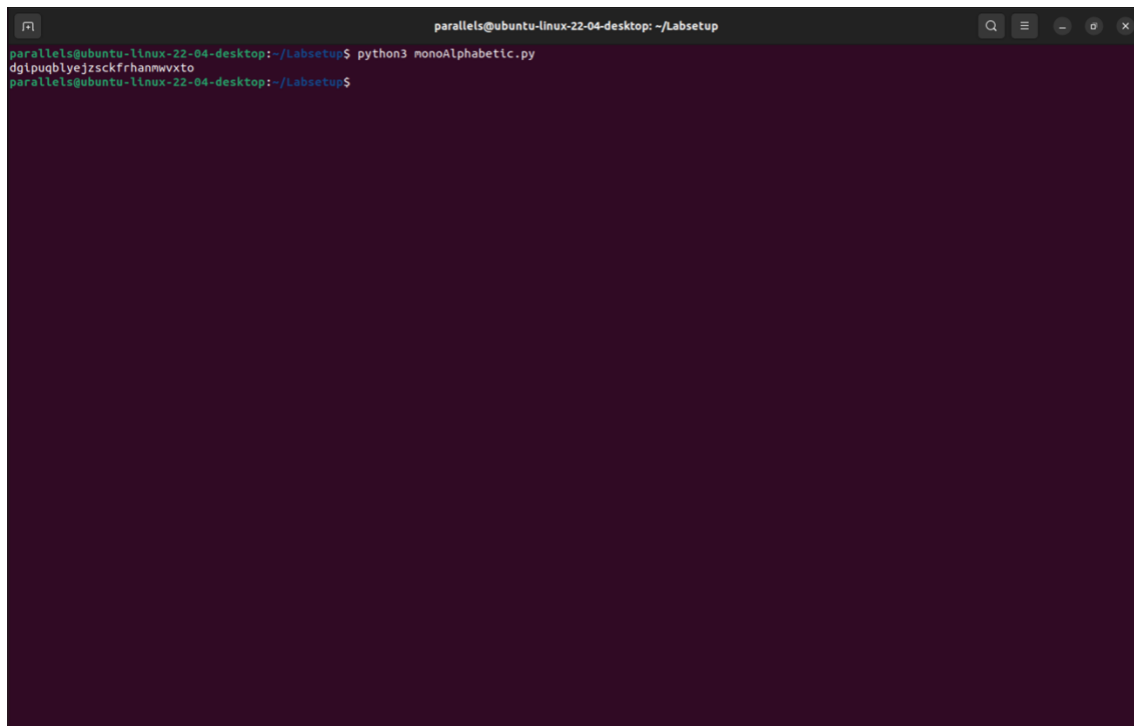
Screenshot3: ([Return to text](#))



```
GNU nano 6.2                                monoAlphabetic.py
#!/bin/env python3
import random
s = "abcdefghijklmnopqrstuvwxyz"
list = random.sample(s, len(s))
key = ''.join(list)
print(key)
```

^G Help ^O Write Out ^M Where Is ^K Cut ^T Execute ^C Location ^U Undo ^-A Set Mark ^-] To Bracket
^X Exit ^R Read File ^_| Replace ^_ Paste ^_ Justify ^_/ Go To Line ^_E Redo ^-G Copy ^_ Where Was

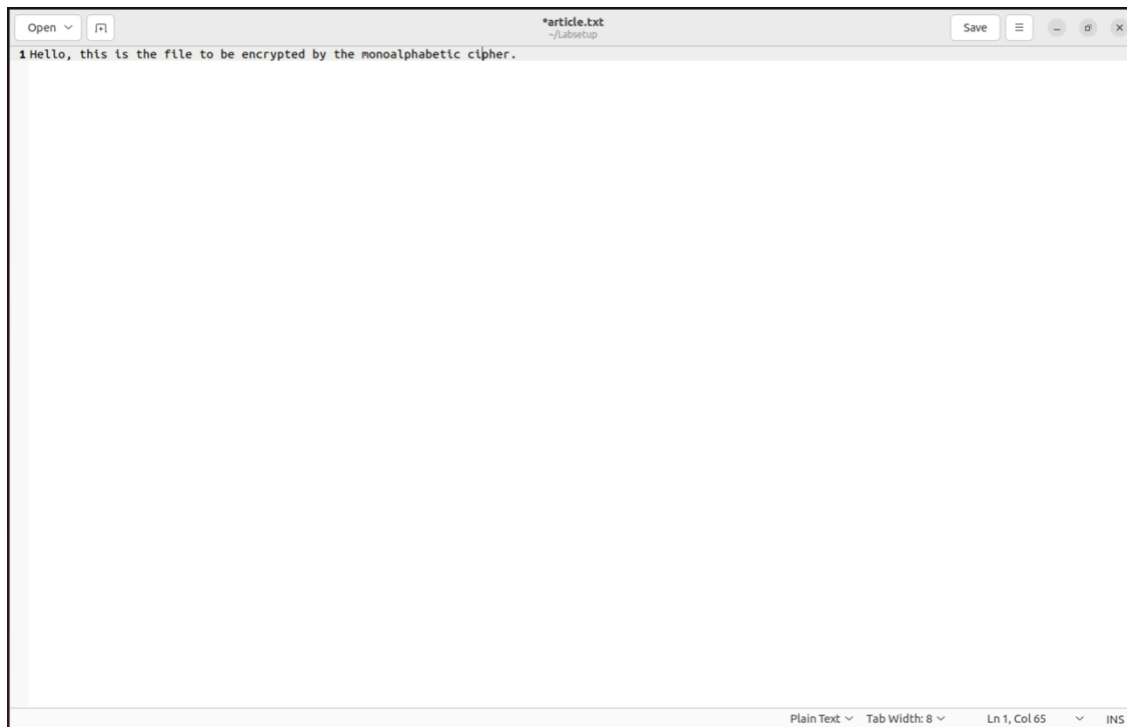
Screenshot4: ([Return to text](#))



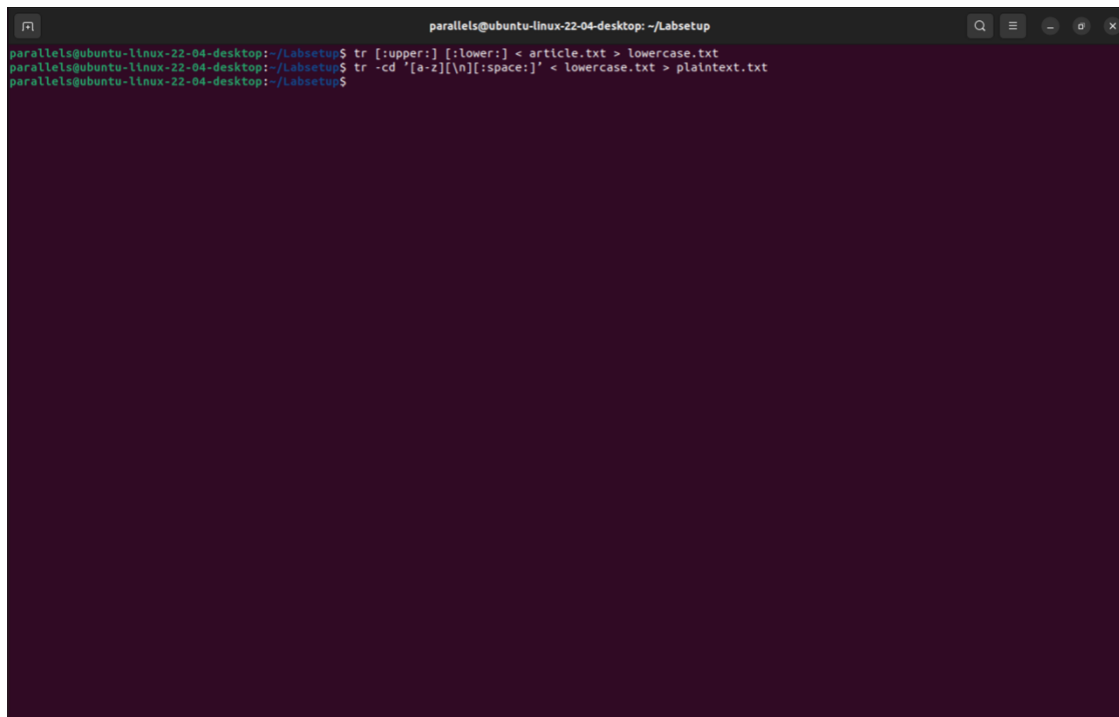
```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup$ python3 monoAlphabetic.py
dgipugblyejzscckfrhammwvxt0
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup$
```

Screenshots: Task 2

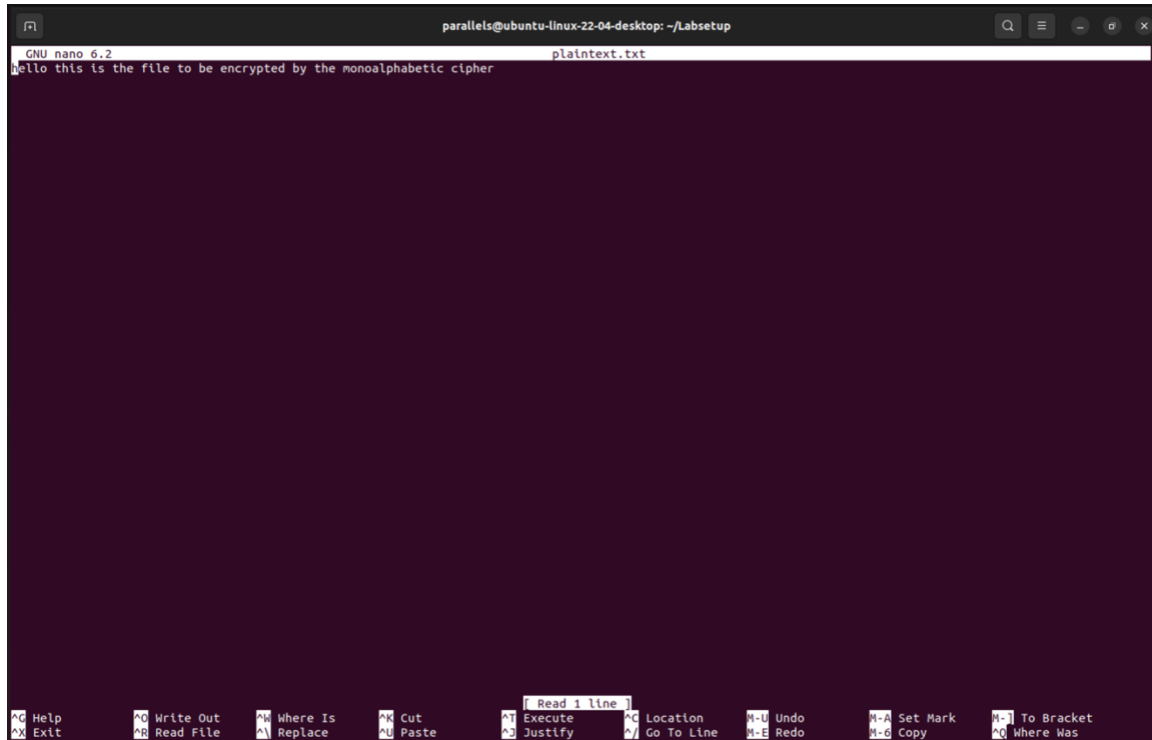
Screenshot5: ([Return to text](#))



Screenshot6: ([Return to text](#))



Screenshot7: ([Return to text](#))

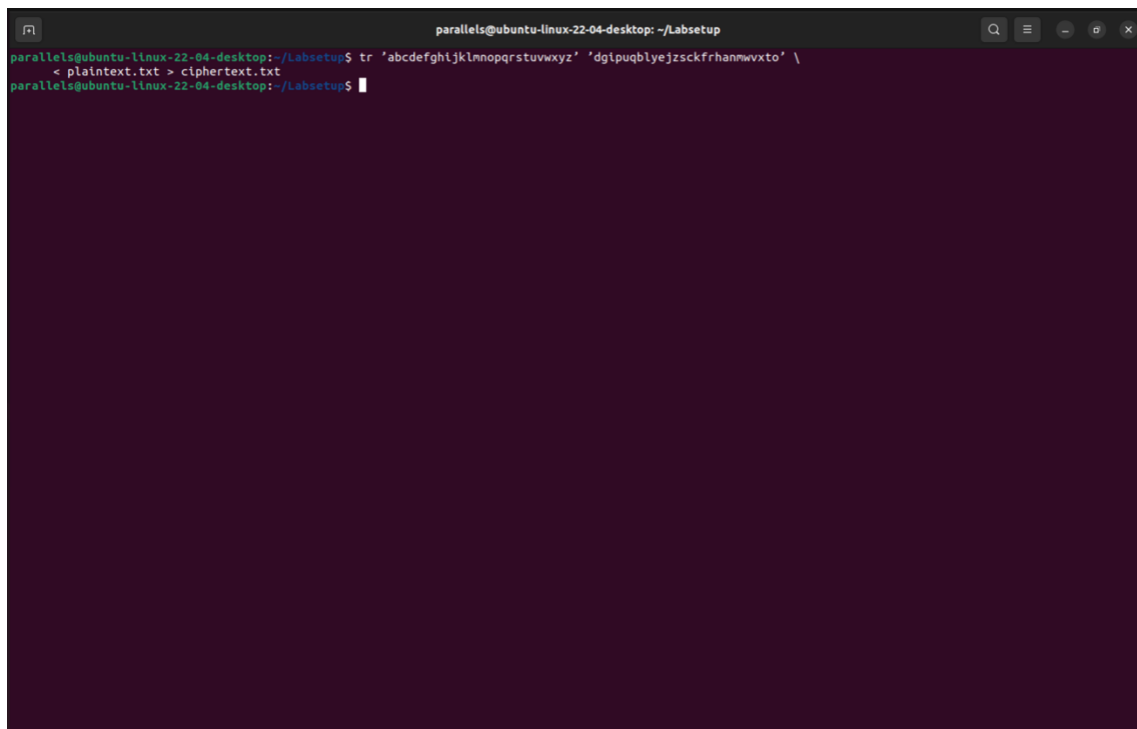


```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup
GNU nano 6.2 plaintext.txt
hello this is the file to be encrypted by the monoalphabetic cipher
```

Read 1 line

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

Screenshot8: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup
parallels@ubuntu-linux-22-04-desktop:~/Labsetup$ tr 'abcdefghijklmnopqrstuvwxyz' 'dglpuqblyejzscckfrhanmwvxt' \
< plaintext.txt > ciphertext.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup$
```

Screenshot9: ([Return to text](#))

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup

```
GNU nano 6.2 cipher.txt
luzzk nlyz ya nlu qyzu nk gu uchtfnup gt nlu skckdzfldgunyl tyfluh
```

Help Write Out Where Is Cut Read 1 line Location Undo Set Mark To Bracket
Exit Read File Replace Paste Justify Go To Line Redo Copy Where Was

Screenshot10: ([Return to text](#))

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files

```
GNU nano 6.2 cipher.txt
ytn xqavhq yzhu xu qzupvd ltnat qnncq vxgzy hmrty vbynh ytnq lxur qyhvurn
vlvhpq yhne ytn gvrnln bnnlq lnsn v uxuvrnvhnvu yxx

ytn vlvhpq hvan lvq gxssnupnp gd ytn pncnq xb tvhfnf lnnuqynnu vy myq xzyqny
vup ytn veevhnuy nceixqmxu xb tnq bnic axcevud vy ytn nup vup my lvq qtnvnp gd
ytn ncnhrnuan xb cnyxx ymcnq ze givasrxlu extnynmaq vhcavupd vaynfnqc vup
v uvyxnuxvl axufnhqvyvnu vq ghmbb vup cvp vq v bnfnn phnvc vxgzy ltnytnh ytnhn
xzrtz yx gn v ehngqpnuy lnuhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv lxur my lvq
nkyhv lxur gnavaqn ytn xqavhq lnnh cxfnp yx ytn bnhqy lnnsnup nu cvhat yx
vfxnp axublnaynur lmyt ytn alxqmur anhnxcud xb ytn lnuynh xidcenaq ytvusq
ednxuratvur

kun gnr jznqynxu qzhhxzupmur ytnq dnvhq vavpncd vlvhpq mq txl xh mb ytn
anhnxcud lml vpphnq cnyxx nqenanvliid vbynh ytn rxlpnu rixgnq ltnat gnacn
v ozgmivuy axcmurxy evhyd bxh ymcnq ze ytn cxfncnuq qenvhtnvpnp gd
exlnhbzl txiidxp lxcnu ltx tnienn hvnqn cmlinxuq xb pxlihvq yx bmrty qnkzvl
tvhvqqcnuy vhxzup ytn axzuyhd

qnruvlmur ytnnh qzeexhy rxlpnu rixgnq vyynupnnq qlvytnp ytnqnlfnq mu givas
qexhynp lvent emuq vup qxzupnp xbb vxgzy qnkny exlnh ncvivuanq bhxc ytn hnp
avhney vup ytn qyrrn xu ytn vnh n lvq avlihp xzy vxgzy evd munjznd vbynh
myq bxhcnh vuatxh avyq qvplnh jzny xuan qtn lnhunp ytyv qtn lvq cvsmur bvh
lnqq ytvu v cvln axtxqy vup pzhmur ytn anhnxcud uvyvinn exhyvnu yxxs v glzuy
vup qvymqbdmur pnr vy ytn vlilvin hxqynh xb uxcmuvynp pnlnayxhq txl axzip
ytyv gn yxeenp

vq my yzhuq xzy vy lnvqy mu ynncq xb ytn xqavhq my ehxvgld lxuy gn

lxcnu mufxlnp mu yncnq ze qmp ytyv vlytxzt ytn rixgnq qnrubnnp ytn
munymvynfnq lvzat ytnf unfnh munynupn my yx gn ozqy vu vlvhpq qnvqxu
avcevmru xh xun ytyv gnacn vqcxamvynp xuld lmyt hnpavhney vaymxuq muqynvp
v qexsnqlxcvu qmp ytn rhxez mq lxhsnur gntmup alxqnp pxxhq vup tvq qmuan
vcvqqnp cmlinxu bxh myq lnrvl pnbnuqn bzup ltnat vbynh ytn rixgnq lvq
blxxpnp lmyt ytxzqvupq xb pxuvymxuq xb xh lnqq bhxc enxeln mu qxcn
axzuyhmq

ux avil yx lnhv givas rxluq lnuu xzy nu vpfvuan xb ytn xqavhq ytxzt ytn
cxfncnuq lnti vlcxy anhyvuld gn hnbhnuanp gnbnhu vup pzhmur ytn anhnxcud
nqenanvliid qmuan fxavi cnyxx qzeexhynq lnsn vqtlnd ozpp lvzhv pnhu vup
unaxln snpcvu vhn qatnpzlnp ehngquynhq

vuxytnh bnvyznh xb ytnq qnvqxu ux xun hnviid suxlg ltx mq rxmur yx lnu gnyq
enayznh vhrzvgld ytnq tveenuq v lxy xb ytn ymcn muvhrzvgld ytn umnignynh
```

Help Write Out Where Is Cut Execute Location Undo Set Mark To Bracket
Exit Read File Replace Paste Justify Go To Line Redo Copy Where Was

Screenshot11: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 freq.py
-----
1-gram (top 20):
n: 488
y: 373
v: 348
x: 291
u: 280
q: 276
m: 264
h: 235
t: 183
l: 166
p: 156
a: 116
c: 104
z: 95
i: 90
g: 83
b: 83
r: 82
e: 76
d: 59
-----
2-gram (top 20):
yt: 115
tn: 89
mu: 74
nh: 58
vh: 57
hn: 57
vu: 56
nq: 53
xu: 52
up: 46
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
vl: 32
gn: 32
av: 31
-----
3-gram (top 20):
ytn: 78

```

Screenshot12: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
r: 82
e: 76
d: 59
-----
2-gram (top 20):
yt: 115
tn: 89
mu: 74
nh: 58
vh: 57
hn: 57
vu: 56
nq: 53
xu: 52
up: 46
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
vl: 32
gn: 32
av: 31
-----
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mxu: 14
gnq: 14
ytn: 13
nqy: 13
vll: 13
bxh: 13
lvq: 12
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tnq: 10
vhp: 10
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot13: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ tr 'ytn' 'THE' < ciphertext.txt > plaintext.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot14: ([Return to text](#))

```

GNU nano 6.2                parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
plaintext.txt
THE xqavhq Tzhu xu qzupud lHmA qEEq vgxzt hmrHT vbTEH THmq Lxur qThvurE
vLVhpq Thme THE gvrreH bEEq tnsE v uxuvrEuvhmV Txx

THE vLVhpq hvaE lvq gxxsEupEp gd THE pECmqE xb HvHfEd lEnuqTEmu vT nTq xztqET
vup THE veevHEuT mcelxqmxu xb Hmq bntc axcevud vT THE Eup vup mT lvq qHveEp gd
THE EcEhrEuaE xb cETxx TncEq ze glvasrxlu exlntma q vhcavupd vaTnfnqc vup
v uvTnxuvl axufEHqvTmxu vq ghmEb vup cvp vq v BEFEH pHEvc vgxzt LHETHEH THEHE
xzrHT Tx gE v eHEqmpEUT lnuhEd THE qEVqxu pmpuT ozqT qEEc EkThv Lxur mT lvq
EkThv Lxur gEavzqE THE xqavhq lEHE cxfEP Tx THE bnhqT lEESep mu cvhAH Tx
vfxmp axublnaTmur lNTH THE alxqmur aHEcXud xb THE lnuTEH xldcenaq THvusq
edExuraHvur

xuE gnr jzEqTnxu qzhxhzupmur THmq dEVhq vavpEd vLVhpq mq HxI xh mb THE
aHEcXud lntL vppHEq cETxx EgeEamvld vbTEH THE rxlpEu rlxgEq lHmA gEavcE
v ozgmivuT axcmurxzT evhTd bxh TncEq ze THE cxfEcEuT qEVHHEvpEp gd
exlEhbzt Hxldlxp lxcEu lHx HEleEp hvnqE cmlnxuq xb pxlvhq Tx bnrHT qEkzvl
HvhvqcEuT vhxzup THE axzuThd

qnruvimur THEnh qzeexHT rxlpEu rlxgEq vTEupEEq qlVTHEP THEcqlfEq mu givas
qexhTEP lveEl emuq vup qxupEp xbb vgxzt qEknoT exlEH mcgvlvuaEq bhxc THE hEP
avheET vup THE qTvrE xu THE vmh E lvq avlEP xzt vgxzt evd muEjzntd vbTEH
mTq bxhcEH vuaHxh avTT qvpLEh jznT xuaE qHE lEVhuEP THVT qHE lvq cvsmur bvh
lEqq THvu v cvLE axHxqT vup pzhmur THE aHEcXud uvTvInE exhTcvu Txxs v glzUT
vup qvTmqbdmur pnr vT THE vllcvLE hxqTEH xb uxcmuvTEP pnhEaTxhq HxI axzlp
THVT gE TxeeEp

vq mT Tzhuq xzt vT lEvqT mu TEhcq xb THE xqavhq mT ehxgvld LxuT gE

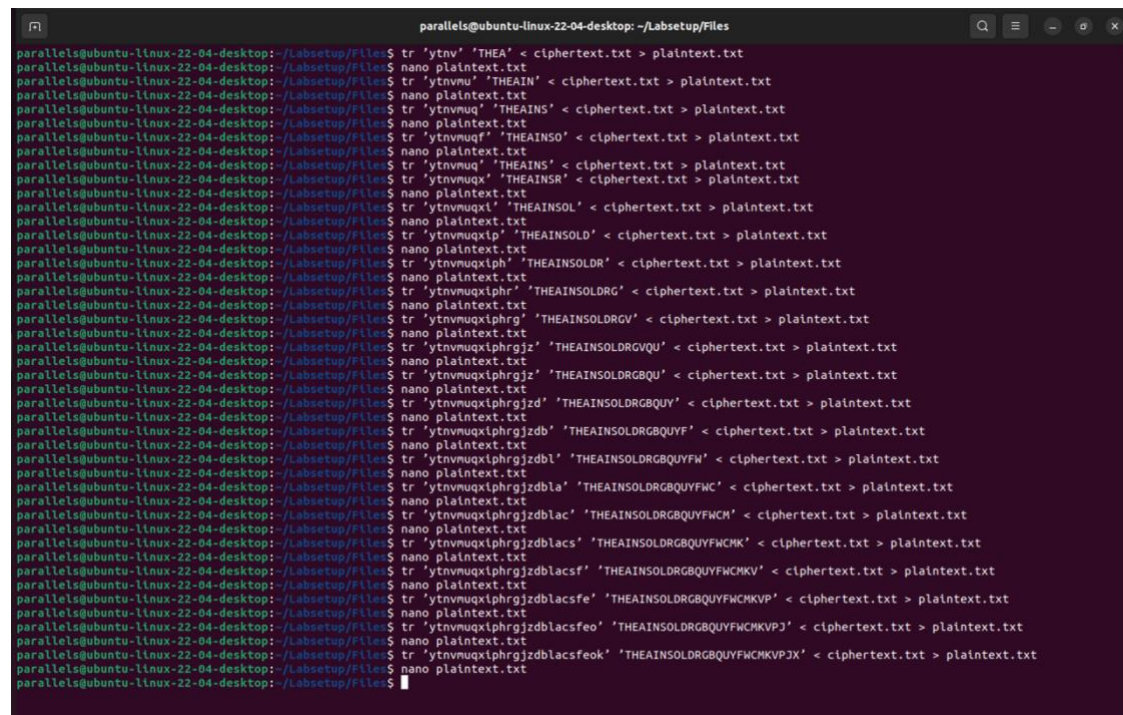
lxcEu mufxlfEP mu TncEq ze qvmp THVT vLTHxzh THE rlxgEq qnrumbEP THE
munTnvTnfEq lVzuAH THEd uEFEH muTEupEp mT Tx gE ozqT vu vLVhpq qEVqxu
avcevnru xh xuE THVT gEavcE vqgxanvTEP xuld lNTH hEpavheET vaTnxuq muqTEvp
v qexsEqlxcvu qvmp THE rhxze mq lxhsmur gEHmup alxqEp pxxhq vup Hvq gnuAE
vcvqEp cmlnxu bxh mTq lErvt pEBEuqE bzup lHmA vbTEH THE rlxgEq lvq
blxxpEp lNTH THxqvupq xb pxuvTmxuq xb xh lEqq bhxc eXelE mu qxcE
axzuThmEq

ux avll Tx lEvh givas rxluq lEUT xzt mu vpfvuaE xb THE xqavhq THxzh THE
cxfEcEuT lntL vlcxqT aEHTvnuId gE hEBHEuaEP gEBxHE vup pzhmur THE aHEcXud
EgeEamvld gnuAE fxavl cETxx qzeexHTeq tnsE vqHlEd ozpp lVzhv pEHu vup
umaxlE snpcvu vHE qAHepzEP eHEqUTEHq

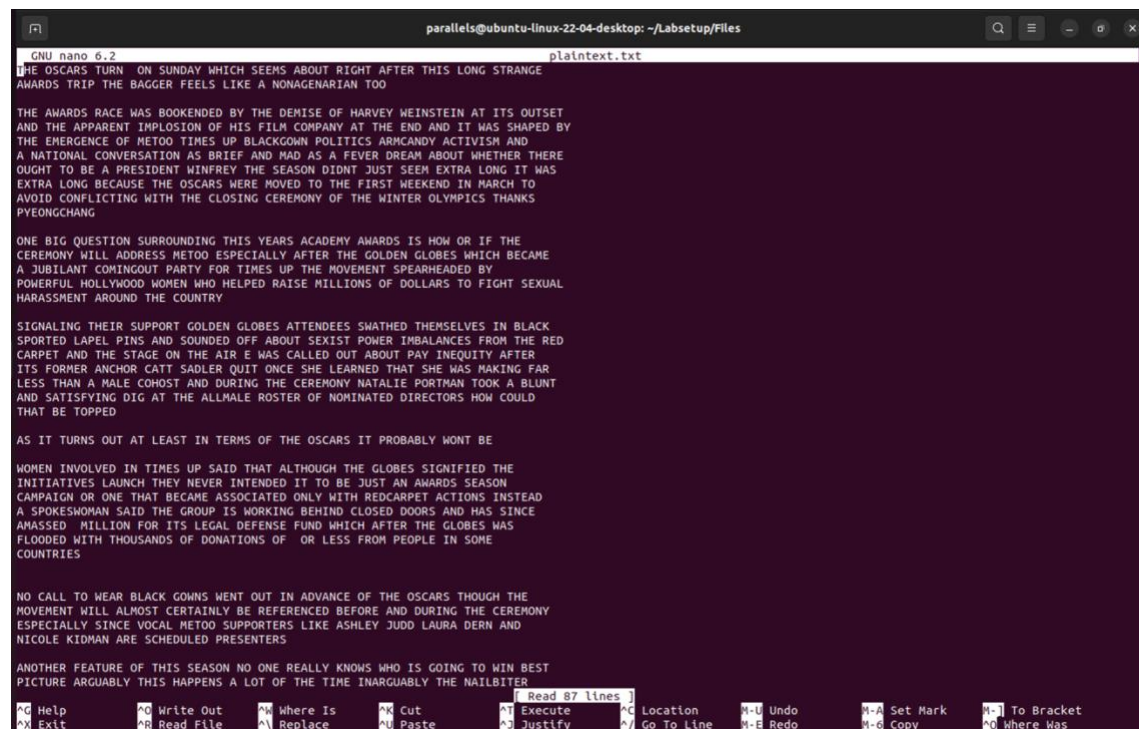
vuXTHEh bEVtzhE xb THmq qEVqxu ux xuE hEvld suxliq lHx mq rxmur Tx lnu gEqT
enaTzHE vhrzvgld THmq HveeEq v lXt xb THE TncE muvhrzvgld THE umnignTEH

```

Screenshot15: ([Return to text](#))



Screenshot16: ([Return to text](#))



Screenshots: Task 2

Screenshot17: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
OPENSsl-CMD5(1SSL)                                OpenSSL                                OPENSsl-CMD5(1SSL)

NAME
    asn1parse, ca, ciphers, cms, crl, crl2pkcs7, dgst, dhparam, dsa, dsaparam, ec, ecpkcs1, enc, engine, errstr, gendsa, genpkey, genrsa, info, kdf,
    mac, nseq, ocsf, passwd, pkcs12, pkcs7, pkcs8, pkey, pkeyparam, pkeyutl, prime, rand, rehash, req, rsa, rsautl, s_client, s_server, s_time,
    sess_id, snlme, speed, spkac, srp, storeutl, ts, verify, version, x509 - OpenSSL application commands

SYNOPSIS
    openssl cmd -help | [-option] [-option arg] ... [arg] ...

DESCRIPTION
    Every cmd listed above is a (sub-)command of the openssl(1) application. It has its own detailed manual page at openssl-cmd(1). For example, to
    view the manual page for the openssl dgst command, type "man openssl-dgst".

OPTIONS
    Among others, every subcommand has a help option.

    -help
        Print out a usage message for the subcommand.

SEE ALSO
    openssl(1), openssl-asn1parse(1), openssl-ca(1), openssl-ciphers(1), openssl-cms(1), openssl-crl(1), openssl-crl2pkcs7(1), openssl-dgst(1),
    openssl-dhparam(1), openssl-dsa(1), openssl-dsaparam(1), openssl-ec(1), openssl-ecparam(1), openssl-enc(1), openssl-engine(1), openssl-errstr(1),
    openssl-gendsa(1), openssl-genpkey(1), openssl-genrsa(1), openssl-info(1), openssl-kdf(1), openssl-mac(1), openssl-nseq(1), openssl-ocsf(1),
    openssl-passwd(1), openssl-pkcs12(1), openssl-pkcs7(1), openssl-pkcs8(1), openssl-pkey(1), openssl-pkeyparam(1), openssl-pkeyutl(1),
    openssl-prime(1), openssl-rand(1), openssl-rehash(1), openssl-req(1), openssl-rsa(1), openssl-rsautl(1), openssl-s_client(1), openssl-s_server(1),
    openssl-s_time(1), openssl-sess_id(1), openssl-snlme(1), openssl-speed(1), openssl-spkac(1), openssl-srp(1), openssl-storeutl(1), openssl-ts(1),
    openssl-verify(1), openssl-version(1), openssl-x509(1),

HISTORY
    Initially, the manual page entry for the "openssl cmd" command used to be available at cmd(1). Later, the alias openssl-cmd(1) was introduced,
    which made it easier to group the openssl commands using the apropos(1) command or the shell's tab completion.

    In order to reduce cluttering of the global manual page namespace, the manual page entries without the 'openssl-' prefix have been deprecated in
    OpenSSL 3.0 and will be removed in OpenSSL 4.0.

COPYRIGHT
    Copyright 2019-2020 The OpenSSL Project Authors. All Rights Reserved.

    Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in
    the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.

3.0.2                                2023-05-24                                OPENSsl-CMD5(1SSL)
~
~
~
Manual page enc(1ssl) line 1/43 (END) (press h for help or q to quit)

```

Screenshot18: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
OPENSsl(1SSL)                                    OpenSSL                                    OPENSsl(1SSL)

NAME
    openssl - OpenSSL command line program

SYNOPSIS
    openssl command [ options ... ] [ parameters ... ]

    openssl list standard-commands | digest-commands | cipher-commands | cipher-algorithms | digest-algorithms | mac-algorithms | public-key-algorithms

    openssl no-XXX [ options ]

DESCRIPTION
    OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and
    related cryptography standards required by them.

    The openssl program is a command line program for using the various cryptography functions of OpenSSL's crypto library from the shell. It can be
    used for

    o Creation and management of private keys, public keys and parameters
    o Public key cryptographic operations
    o Creation of X.509 certificates, CSRs and CRLs
    o Calculation of Message Digests and Message Authentication Codes
    o Encryption and Decryption with Ciphers
    o SSL/TLS Client and Server Tests
    o Handling of S/MIME signed or encrypted mail
    o Timestamp requests, generation and verification

COMMAND SUMMARY
    The openssl program provides a rich variety of commands (command in the "SYNOPSIS" above). Each command can have many options and argument
    parameters, shown above as options and parameters.

    Detailed documentation and use cases for most standard subcommands are available (e.g., openssl-x509(1)).

    The list options -standard-commands, -digest-commands, and -cipher-commands output a list (one entry per line) of the names of all standard
    commands, message digest commands, or cipher commands, respectively, that are available.

    The list parameters -cipher-algorithms, -digest-algorithms, and -mac-algorithms list all cipher, message digest, and message authentication code
    names, one entry per line. Aliases are listed as:

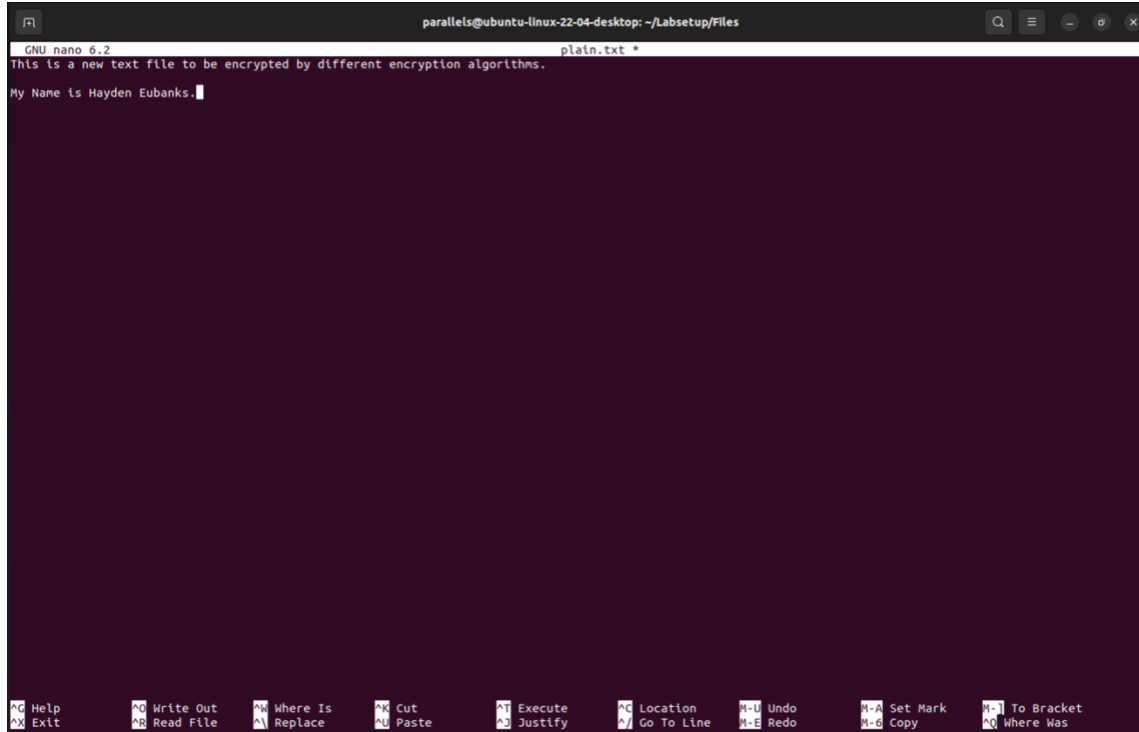
    from => to

    The list parameter -public-key-algorithms lists all supported public key algorithms.

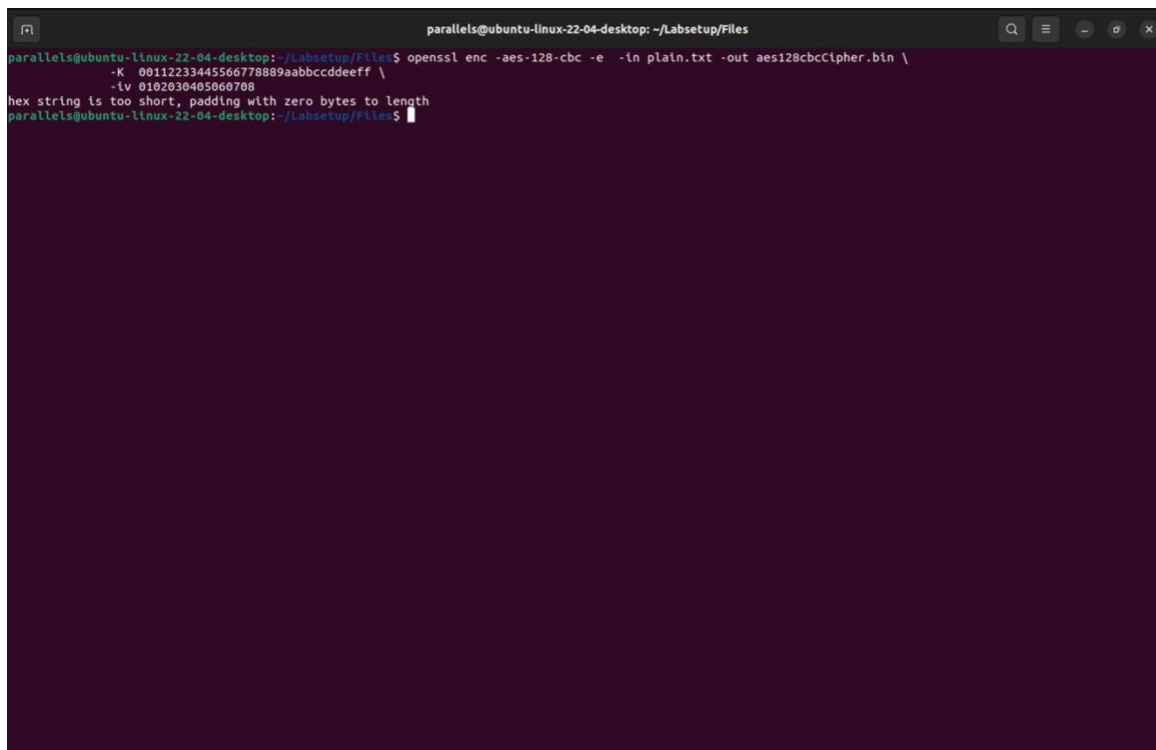
    The command no-XXX tests whether a command of the specified name is available. If no command named XXX exists, it returns 0 (success) and prints
    no-XXX; otherwise it returns 1 and prints XXX. In both cases, the output goes to stdout and nothing is printed to stderr. Additional command line
    arguments are always ignored. Since for each cipher there is a command of the same name, this provides an easy way for shell scripts to test for

Manual page openssl(1ssl) line 1 (press h for help or q to quit)

```

Screenshot19: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
GNU nano 6.2 plain.txt *
This is a new text file to be encrypted by different encryption algorithms.
My Name is Hayden Eubanks.
^O Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   ^U Undo       ^-A Set Mark  ^-] To Bracket
^X Exit      ^R Read File  ^A Replace    ^U Paste      ^J Justify    ^_ Go To Line ^-E Redo      ^-G Copy     ^-O Where Was
```

Screenshot20: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in plain.txt -out aes128cbcCipher.bin \
-K 08112233445566778899aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot21: ([Return to text](#))

```

GNU nano 6.2 aes128cbcCipher.bin
0x08\x0e\x1e^C^e^T^e^Y^e^N];^E^48`^?^@^Y^l^@^X^t^7\^e^@^S^退^e^@^D^e^f^e^p^e^D^s!^p^K^e^@^E^e^e^H^e^e^Q^e^@^X^X^e^e^f^@^L^L^Q^v^t^@^B^Q^@^X^e^e^e^e^0`^@^R^e^@^H^S^e^e^o^7^e^e^@^#^I^e^@^}^@^l

```

Screenshot22: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in plain.txt -out aes128cbcCipher.bin \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ nano aes128cbcCipher.bin
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cfb -e -in plain.txt -out aes128cfbCipher.bin -K 00112233445566778
889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ nano aes128cfbCipher.bin
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

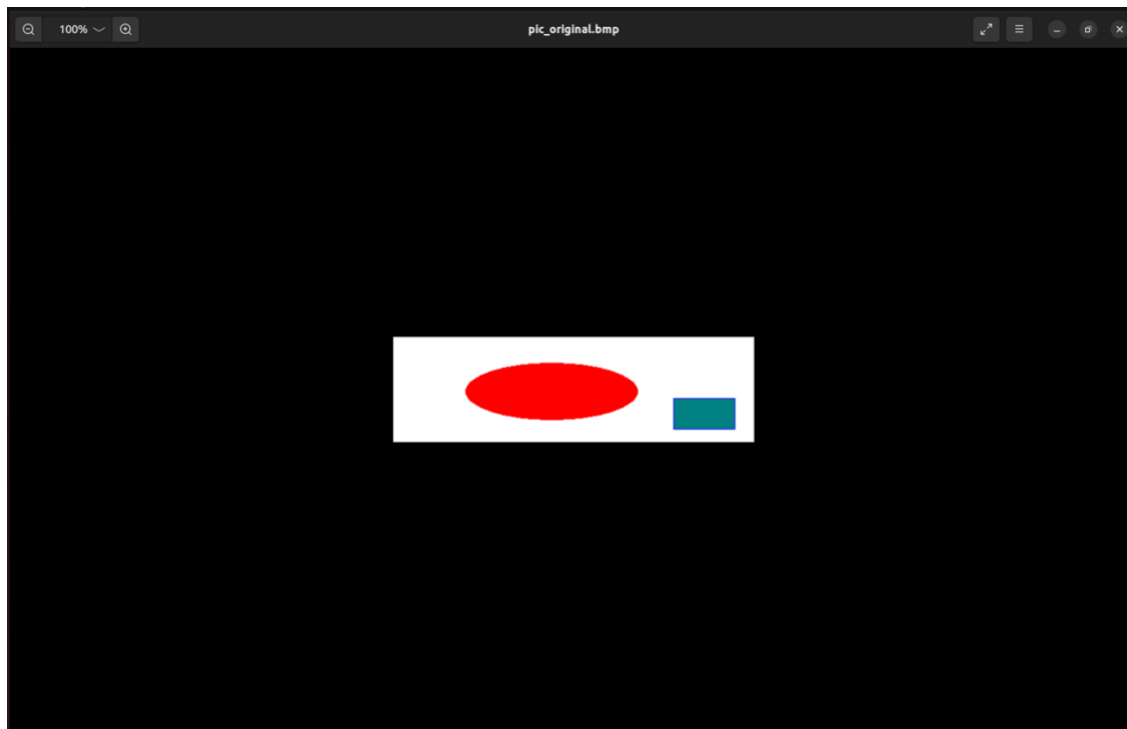
```


Screenshot25: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
GNU nano 6.2 des3Cipher.bin
Bf000Q0^UZ0+00^A0>^D00kD0000^V0^K0<'z^^X^Z^C0000n0L0^F000UY0^V0L00000^X00{\0YR0^b^D0^F0^}0jL.00d000(000
000v0c^B00kP^H0d000=
^G Help      ^O Write Out  ^N Where Is   ^K Cut        ^T Execute    ^C Location   ^U Undo       ^-A Set Mark   ^-] To Bracket
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^/_ Go To Line  ^E Redo       ^-G Copy      ^-0 Where Was
```

Screenshots: Task 3

Screenshot26: ([Return to text](#))



Screenshot27: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out ECBPic.bmp \
-K 00112233445566778899aabbccddeeff
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot28: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out ECBPic.bmp \
-K 00112233445566778889aabbccddeeff
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ dd if=pic_original.bmp of=ECBPic.bmp bs=54 count=1 conv=notrunc
1+0 records in
1+0 records out
54 bytes copied, 0.000470642 s, 115 kB/s
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot29: ([Return to text](#))



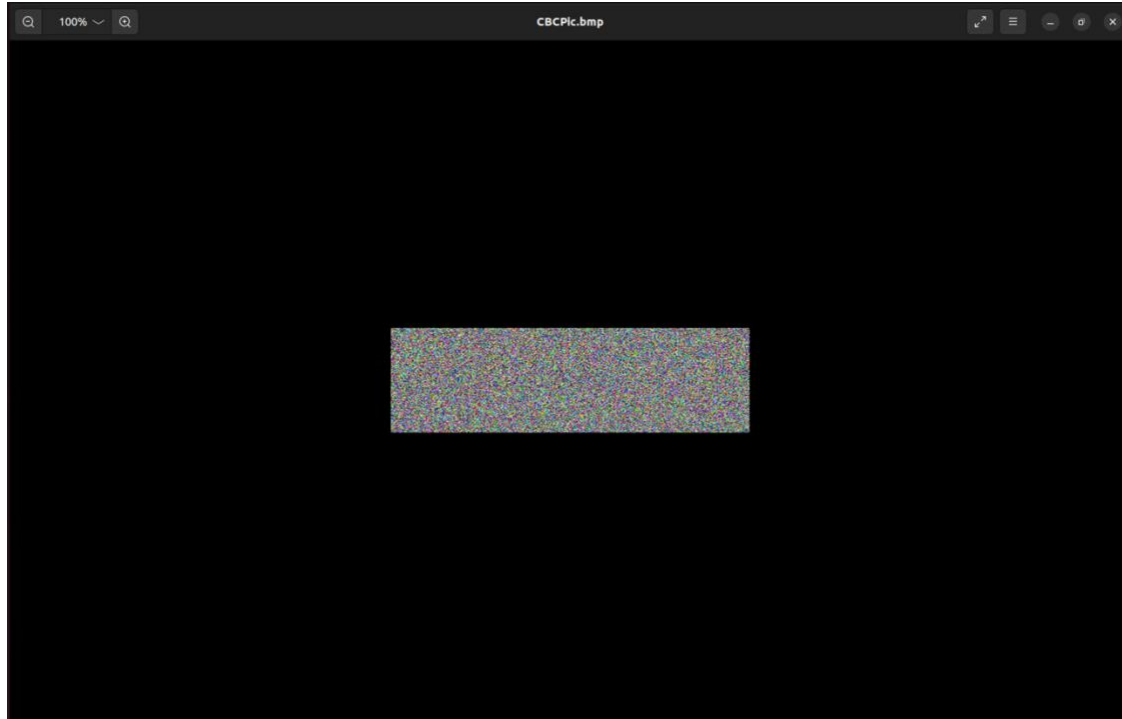
Screenshot30: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out CBCPic.bmp \
-K 00112233445566778889aabbccddeeff
iv undefined
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out CBCPic.bmp \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

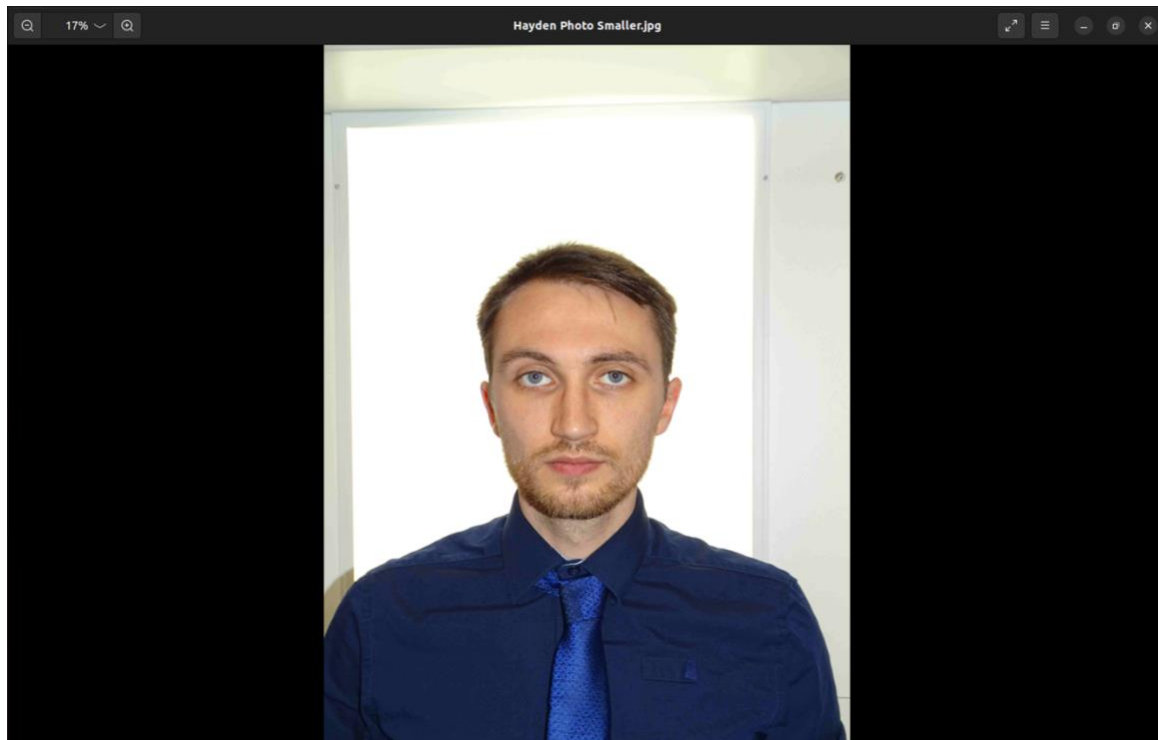
Screenshot31: ([Return to text](#))

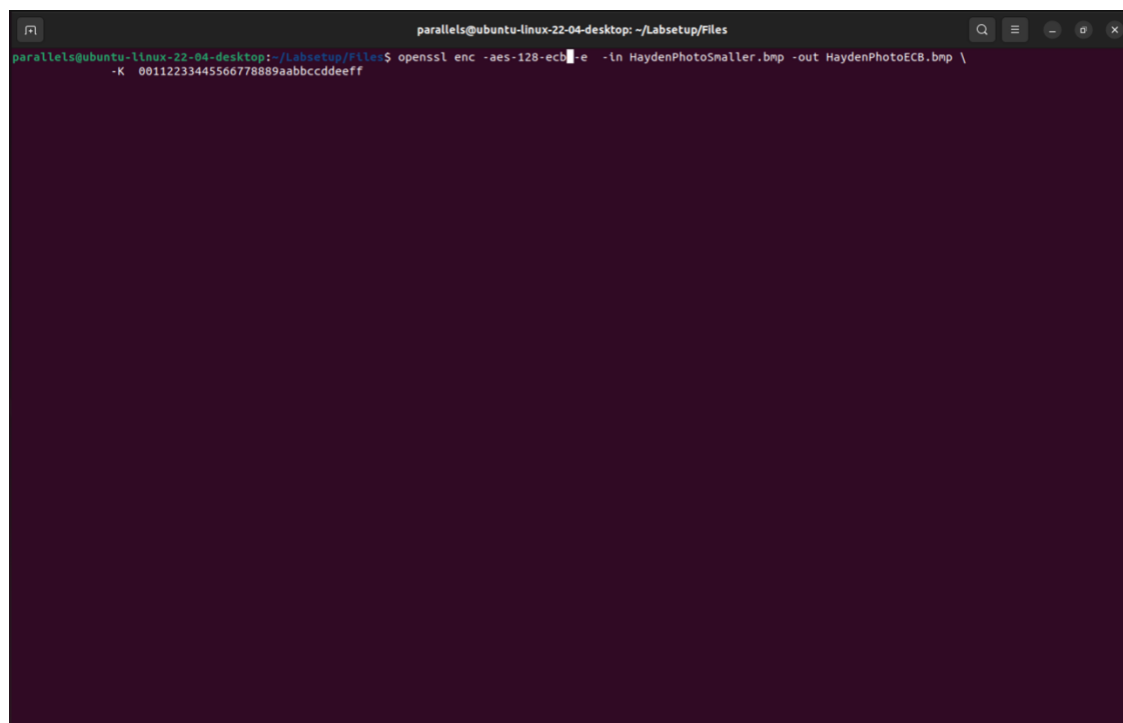
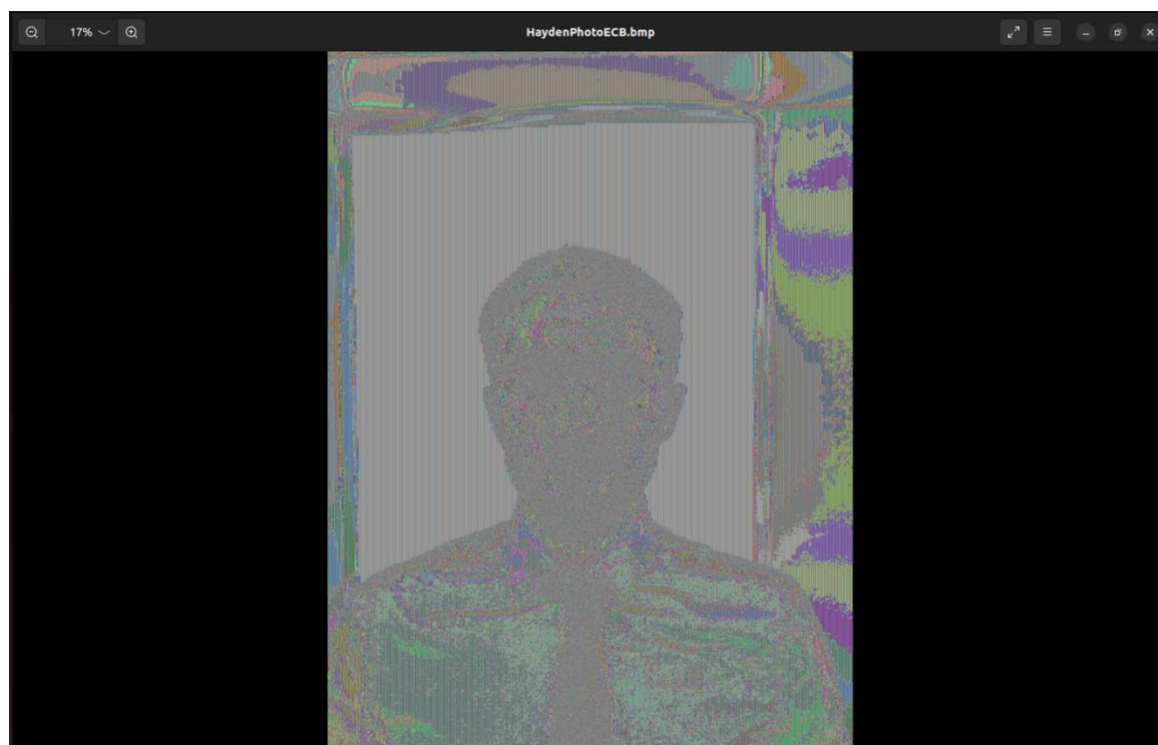
```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out CBCPic.bmp \
-K 00112233445566778889aabbccddeeff
iv undefined
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out CBCPic.bmp \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ dd if=pic_original.bmp of=CBCPic.bmp bs=54 count=1 conv=notrunc
1+0 records in
1+0 records out
54 bytes copied, 0.000160084 s, 337 kB/s
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

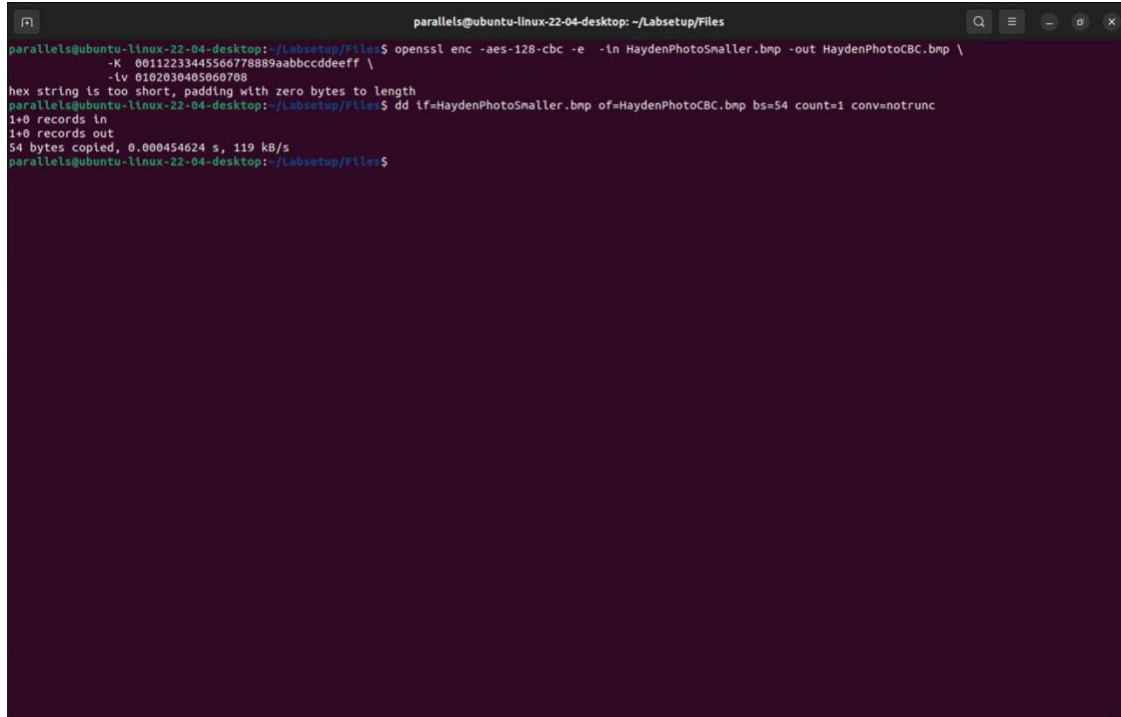
Screenshot32: ([Return to text](#))



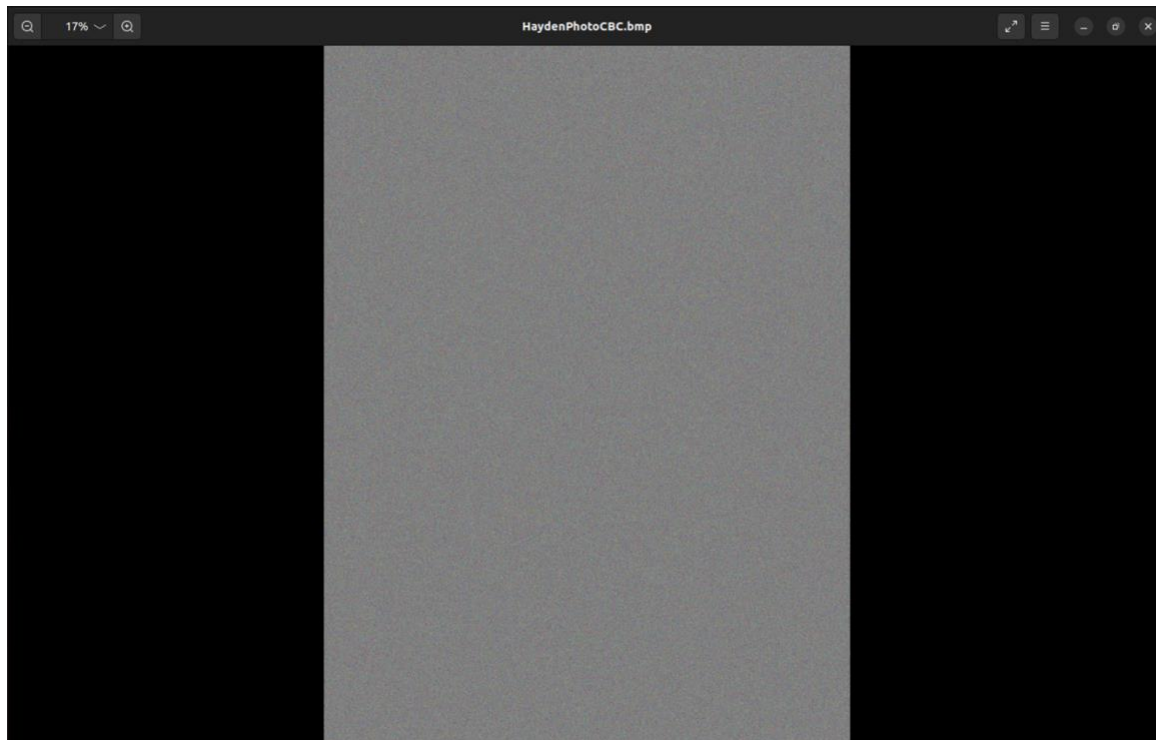
Screenshot33: ([Return to text](#))



Screenshot34: ([Return to text](#))Screenshot35: ([Return to text](#))

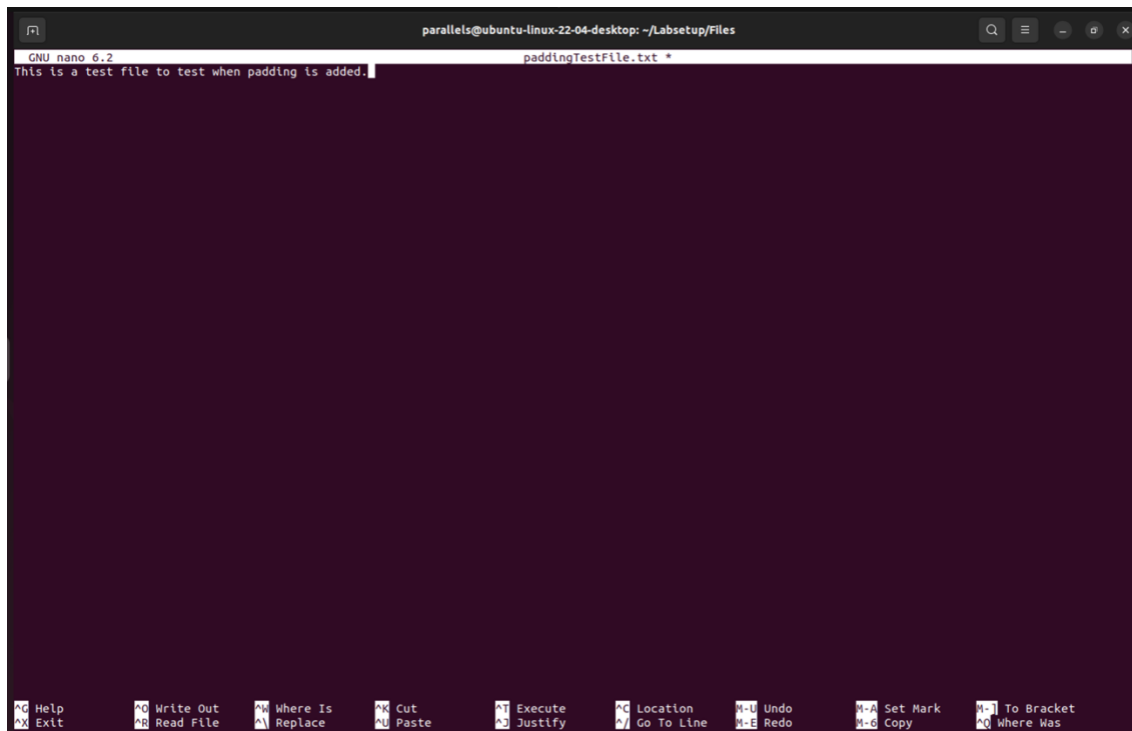
Screenshot36: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in HaydenPhotoSmaller.bmp -out HaydenPhotoCBC.bmp \
-K 00112233445566778899aabbccddeeff \
-lv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ dd if=HaydenPhotoSmaller.bmp of=HaydenPhotoCBC.bmp bs=54 count=1 conv=notrunc
1+0 records in
1+0 records out
54 bytes copied, 0.000454624 s, 119 kB/s
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot37: ([Return to text](#))

Screenshots: Task 4

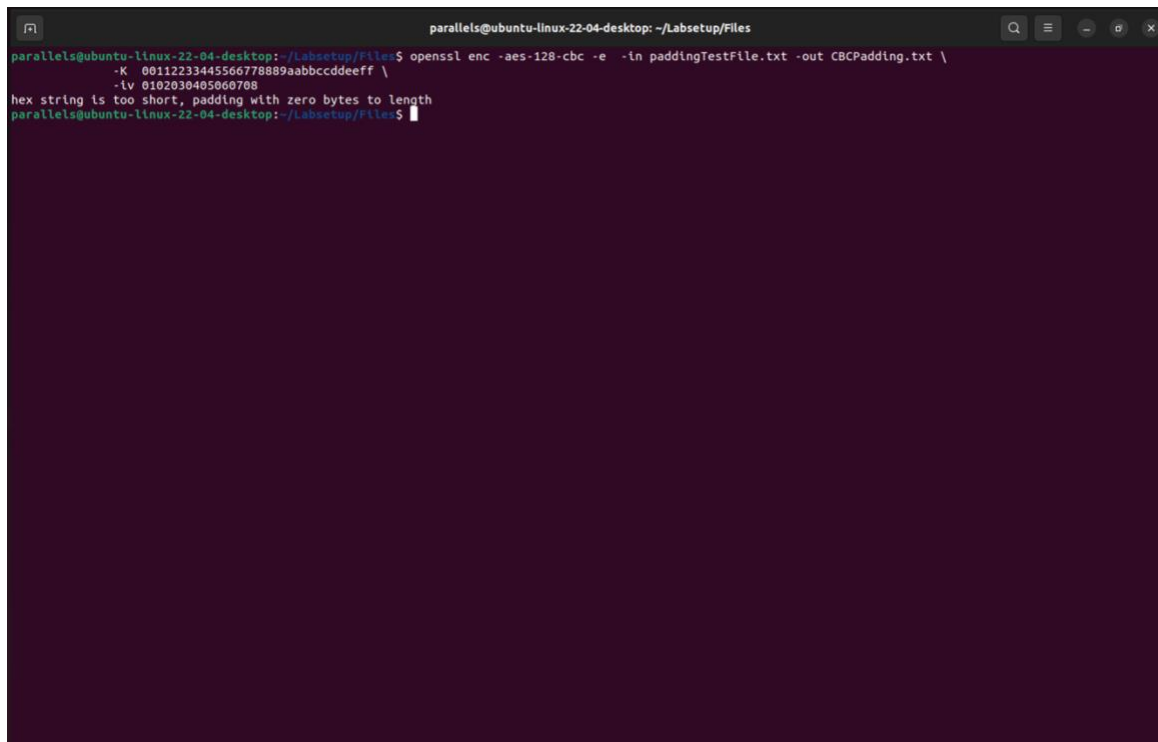
Screenshot38: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
GNU nano 6.2 paddingTestFile.txt *
This is a test file to test when padding is added.
```

The screenshot shows a terminal window with the nano text editor. The editor is open to a file named 'paddingTestFile.txt' in the directory '~/Labsetup/Files'. The file contains the text 'This is a test file to test when padding is added.' The nano editor's status bar at the bottom shows various keyboard shortcuts for editing and navigation.

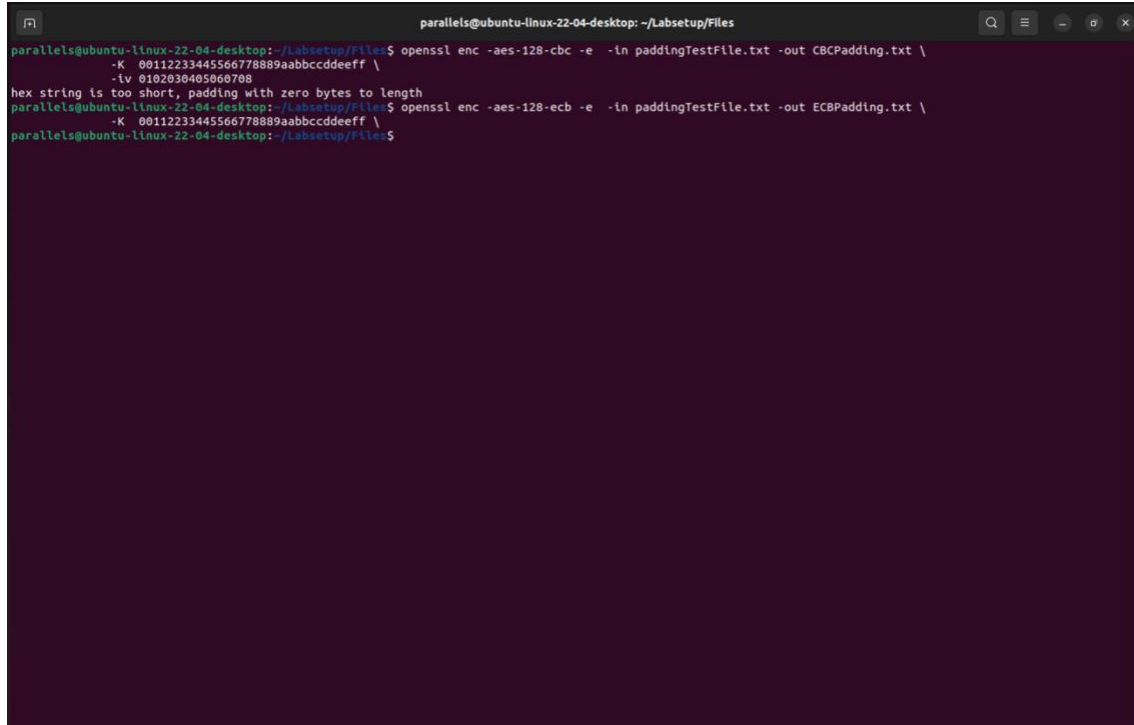
Screenshot39: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in paddingTestFile.txt -out CBCPadding.txt \
-K 00112233445566778899aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

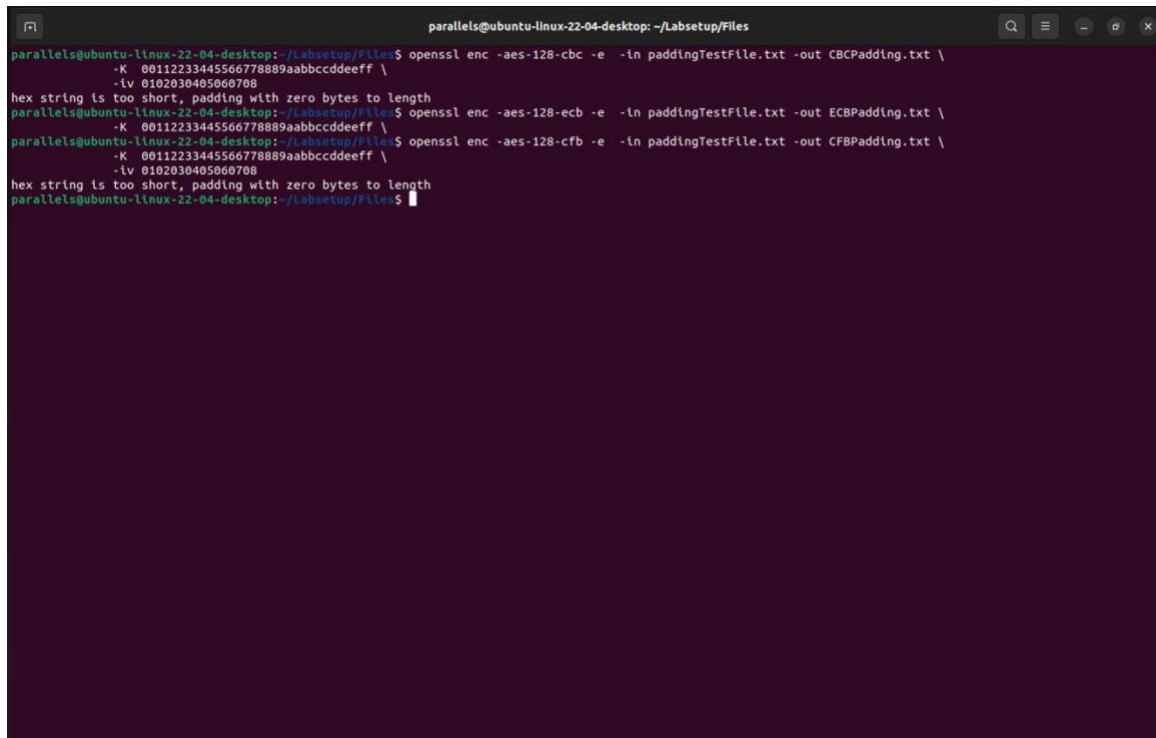
The screenshot shows a terminal window with the command 'openssl enc -aes-128-cbc -e -in paddingTestFile.txt -out CBCPadding.txt \ -K 00112233445566778899aabbccddeeff \ -iv 0102030405060708' being executed. The output shows the command running successfully, and the message 'hex string is too short, padding with zero bytes to length' is displayed. The prompt returns to the user.

Screenshot40: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in paddingTestFile.txt -out CBCPadding.txt \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in paddingTestFile.txt -out ECBPadding.txt \
-K 00112233445566778889aabbccddeeff \
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot41: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in paddingTestFile.txt -out CBCPadding.txt \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in paddingTestFile.txt -out ECBPadding.txt \
-K 00112233445566778889aabbccddeeff \
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cfb -e -in paddingTestFile.txt -out CFBPadding.txt \
-K 00112233445566778889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot42: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in paddingTestFile.txt -out CBCPadding.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in paddingTestFile.txt -out ECBPadding.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cfb -e -in paddingTestFile.txt -out CFBPadding.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ofb -e -in paddingTestFile.txt -out OFBPadding.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot43: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ echo -n "12345" > f1.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ echo -n "123456789a" > f1.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ echo -n "0123456789abcdef" > f3.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ echo -n "123456789a" > f2.txt
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot44: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f1.txt -out f1Encrypted.txt \
-K 00112233445566778889aabbccddeeff \
-Iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f2.txt -out f2Encrypted.txt \
-K 00112233445566778889aabbccddeeff \
-Iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in f3.txt -out f3Encrypted.txt \
-K 00112233445566778889aabbccddeeff \
-Iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot45: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ ll f*
-rw-rw-r-- 1 parallels parallels  5 Sep 22 11:37 f1.txt
-rw-rw-r-- 1 parallels parallels 16 Sep 22 11:37 f1Encrypted.txt
-rw-rw-r-- 1 parallels parallels 10 Sep 22 11:34 f2.txt
-rw-rw-r-- 1 parallels parallels 16 Sep 22 11:35 f2Encrypted.txt
-rw-rw-r-- 1 parallels parallels 16 Sep 22 11:33 f3.txt
-rw-rw-r-- 1 parallels parallels 32 Sep 22 11:36 f3Encrypted.txt
-rwxrwxr-x 1 parallels parallels 786 Nov  7 2021 freq.py*
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```


Screenshot50: ([Return to text](#))

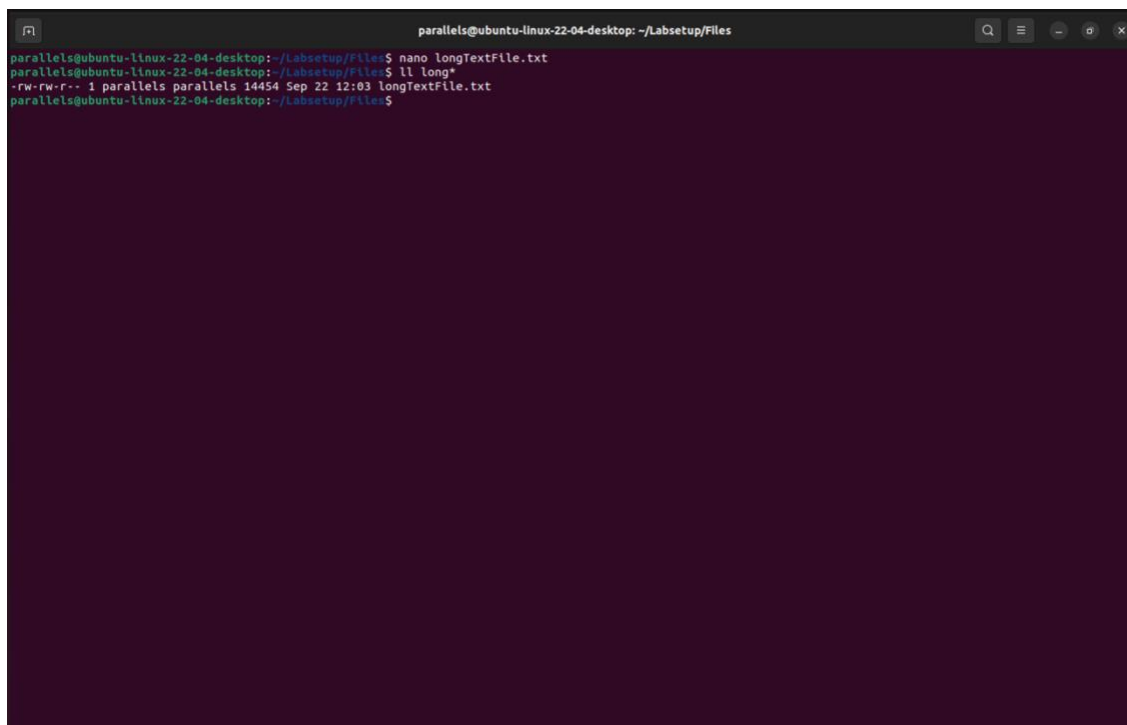
```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ hexdump -C f1Decrypted.txt
00000000  31 32 33 34 35 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b |12345.....|
00000010
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ hexdump -C f2Decrypted.txt
00000000  31 32 33 34 35 36 37 38 39 61 06 06 06 06 06 06 |123456789a.....|
00000010
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ hexdump -C f3Decrypted.txt
00000000  30 31 32 33 34 35 36 37 38 39 61 62 63 64 65 66 |0123456789abcdef|
00000010  10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 |.....|
00000020
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ xxd f1Decrypted.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b 12345.....
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ xxd f2Decrypted.txt
00000000: 3132 3334 3536 3738 3961 0606 0606 0606 123456789a.....
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ xxd f3Decrypted.txt
00000000: 3031 3233 3435 3637 3839 6162 6364 6566 0123456789abcdef
00000010: 1010 1010 1010 1010 1010 1010 1010 1010 .....
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshots: Task 5

Screenshot51: ([Return to text](#))



Screenshot52: ([Return to text](#))



Screenshot53: ([Return to text](#))

```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in longTextFile.txt -out longEncryption.txt \
-K 00112233445566778889aabbccddeeff \
-lv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-cfb -e -in longTextFile.txt -out longEncryptionCFB.txt \
-K 00112233445566778889aabbccddeeff \
-lv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ofb -e -in longTextFile.txt -out longEncryptionOFB.txt \
-K 00112233445566778889aabbccddeeff \
-lv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ openssl enc -aes-128-ecb -e -in longTextFile.txt -out longEncryptionECB.txt \
-K 00112233445566778889aabbccddeeff
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$

```

Screenshot54: ([Return to text](#))

/home/parallels/Labsetup/Files/longEncryptionCBC.txt* - Bless

File Edit View Search Tools Help

longEncryptionCBC.txt*

```

00000000 1A 00 91 8B A5 1A 5D EC B0 A0 9E B2 C4 09 95 70 18 83 FF F3 D2 88 57 85 41 8C 20 44 9C ED A4 0C DB .....].....p.....W.A. D.....
00000001 6D ED 3D E7 60 D7 BA C0 64 E8 ED 89 DC 06 75 DE A6 4D 6D 58 E5 C0 3E 8D 15 12 56 AC 3C E5 F3 31 E9 m.=.'...d.....u.MmX...>...V.<...l.
00000002 4C 59 C6 1C BC C5 57 4F 03 38 7B 89 1F 6F B6 8B 0C 8F 8D 80 02 34 83 E0 C5 21 A0 B7 90 B9 76 DC EB LY...WO.8{...o.....4.....V...
00000003 8F 7C 5C 70 36 88 08 9C 93 BB D4 DA 68 2B 9C 2E 90 0F 2B 1E 86 94 A9 BB E3 7D 69 05 5E 2B 0C E9 AB .|\p6.....h+.....4.....]i.A+...
00000004 07 1D 31 37 09 F6 E0 3B 44 DD D7 BE E6 F5 B9 D1 9E 53 19 45 BE 68 A8 98 89 1B 6A 47 05 9C 1D 1C 3B ..17...;D.....S.E.h.....jG...;
00000005 49 30 9D 0D B5 C6 78 07 05 E2 16 FB 0D 36 EA C9 E7 6B 16 4E 6F 1F E3 64 BA 6D 4B 9E C3 39 41 11 E5 IO...x.....6...k.No...d.mK...9A...
00000006 38 3A 4C 23 0A C1 3E 86 63 3E 2D 11 8D 30 1C C1 C9 BE 26 F3 71 E9 CA B0 12 4F 02 F1 BA 1A B6 3C 02 8:L#>...c>...o...&...O...<...
00000007 0B D3 14 46 4A 97 9A 74 E6 0D D5 BE 72 10 2A 21 39 77 DE FA D3 AD 02 36 52 FE 00 4B 92 5F D7 5B FE ...FJ...t...t...*!9w...6R...K...[.
00000008 89 6E 39 28 31 60 D0 46 CE 7E 41 1F 3D 47 BE 62 BF 16 00 53 C1 7E 84 9B 28 F3 8B 25 C3 7D 69 0F BE .n9(1'.F.-A.=G.b...S...~(...%)i...
00000009 E8 F2 82 03 FF 78 8E 0C 2D B3 60 4D 1C D3 88 FA C7 D6 A4 DC 1C F8 44 9E E4 03 FE 63 50 B6 49 F8 AB .....x...M.....D.D...cP.I...
0000000A F1 83 2B CA D6 C2 75 A2 A1 E7 55 FA AF 1E B0 31 8A 9A 0F 85 7F 92 D2 89 D9 07 01 1F B0 D3 C9 0D AC .+...u...U.....l.....
0000000B 59 AC 03 15 71 72 C0 DB 9B 17 3D D8 AA EE 8F 7B 6F 6C BF C3 D5 F6 E4 34 ED 4B 80 89 51 A9 FF 68 71 Y...qr.....[ol.....4.H...Q..hq
0000000C 27 38 3B EE 8E F8 4F 0C DB 79 0E 19 DE 86 65 FF 83 43 24 E3 DC B0 3E 9F 05 80 4B 53 42 00 BC 52 *8)...O...y.....cB...>...KBB...R
0000000D 50 09 71 20 95 55 9B 0E 0B 07 0C 6D FA 77 C4 2C 55 21 13 87 19 D6 64 33 A1 E4 D8 63 55 FA EB 4F AE p.q...U.....m.W,U.....43.....cU...O...
0000000E 03 D0 BE F8 87 4B 32 AA B9 6D FA 1C 98 4E FC D3 FF BB 51 35 21 7A 7B 9D 8A 2D 06 FE 61 DD 65 C8 0F .....K2...m...N...Q5iz{...a...e...
0000000F 2D F3 AE 51 57 85 CF EF B2 CE 85 69 4E 36 39 73 02 2C C8 A2 3C D4 5F 61 35 DC 71 01 87 AE A9 BF 4B ...QW.....iN69m...<...a5.q...e...X
00000010 41 52 55 A4 6A 4D C6 D5 7A BA 9B 7F AD F2 03 E3 40 B7 93 C1 DD 12 0E 3B F0 BE 13 4B 82 1C 5E 2E 1F ARU.jM.z...z...e...<...K...A...
00000011 BA 40 E4 B5 B7 9D F6 F2 CF B1 40 18 65 43 F1 5E 11 EE 15 3D C3 97 68 49 50 52 56 A6 FC 07 25 2F 2E .@.....@...eC.A...=.hIPRV...%/.
00000012 D4 90 C1 01 DD D2 B1 45 88 B5 89 97 05 6A 57 EA 34 6E 99 DC A9 88 BA 7F 6F 93 D3 91 50 07 0C 47 DE .....E.....jW.4n.....O...P...G...
00000013 B9 8D A4 76 86 F1 D3 29 BA 41 BD 1E 53 F4 A2 DB E6 80 05 36 8D 49 00 9C FA 4E C4 DD C0 A4 FB C2 A9 ...v...).A...S.....6.I...N.....
00000014 8E 76 4A 51 82 25 01 C1 DB 13 27 8D A3 90 F3 2E 12 3B D6 02 80 A8 33 26 68 C3 C9 A2 A3 93 AF B4 C4 .vJQ.%.....;.....3khe...
00000015 A2 0B 97 02 64 A7 2C 74 9D 25 21 88 90 8A 28 C4 FB 65 FD D2 10 2E 5A 43 2F 30 C7 B3 BA 2E C9 80 9A ...d...t.%!...{...e...>C/0.....
00000016 BC C0 4B 69 3B F9 23 4E D8 B6 16 36 5D 32 63 83 D7 C7 1A 9C 6C 70 3F CF 7E 54 3C 05 6B CF 99 E9 AC .Ki;.#N...6]2c...lp?...T<.k....
00000017 F2 0B 2E EE B6 9C 6F 63 2D 43 51 87 96 55 47 EC B3 6C DD E6 26 DC F6 3F 8A 57 23 67 DD B9 2E F6 EA .....oc-CQ..UG..l...&?...?..WBg...
00000018 86 66 59 CE F3 90 1F 6A 8B 58 53 20 09 2D BE F8 5C B9 D7 9E 79 BC 31 0A 2B 4E CF AE 1D 98 63 41 19 .fY...j.XS.....>...y.l.+N.....CA.
00000019 D5 F9 D7 FB 93 DB 1D 07 11 A9 D9 CA 10 26 F6 8D 80 70 D7 F7 3D D2 54 ED FD 20 11 B7 2B 0C B4 46 3E .....>...&h.p...m.T...>...FP
0000001A DD 3F 42 9B DD B4 73 1E A0 60 F0 33 87 5A 97 BD 16 AE 07 8D 04 C9 5C 8B B4 B1 D8 05 D5 5F 14 46 8E .7B...>...3.Z.....>...>...F...
0000001B C3 75 91 15 4E B6 B3 EC 6B ED E1 97 0F 6C 73 79 5C F6 BD 08 B4 28 22 4B E4 72 58 C4 9F 8E 41 0D 0B .u...W...K...ley'....('K.vK...A...
0000001C 78 8E 9D 34 57 80 35 52 5D ED 5B 5F 3F F8 3A F9 11 AF 58 A6 6B 78 3A 3E 6C 5F D8 47 6F DA B0 0F D5 |x..4W.5R].[_?...x.kx;>1_Go....

```

Signed 8 bit: -64 Signed 32 bit: -1072360874 Hexadecimal: C0 15 12 56

Unsigned 8 bit: 192 Unsigned 32 bit: 3222606422 Decimal: 192 021 018 086

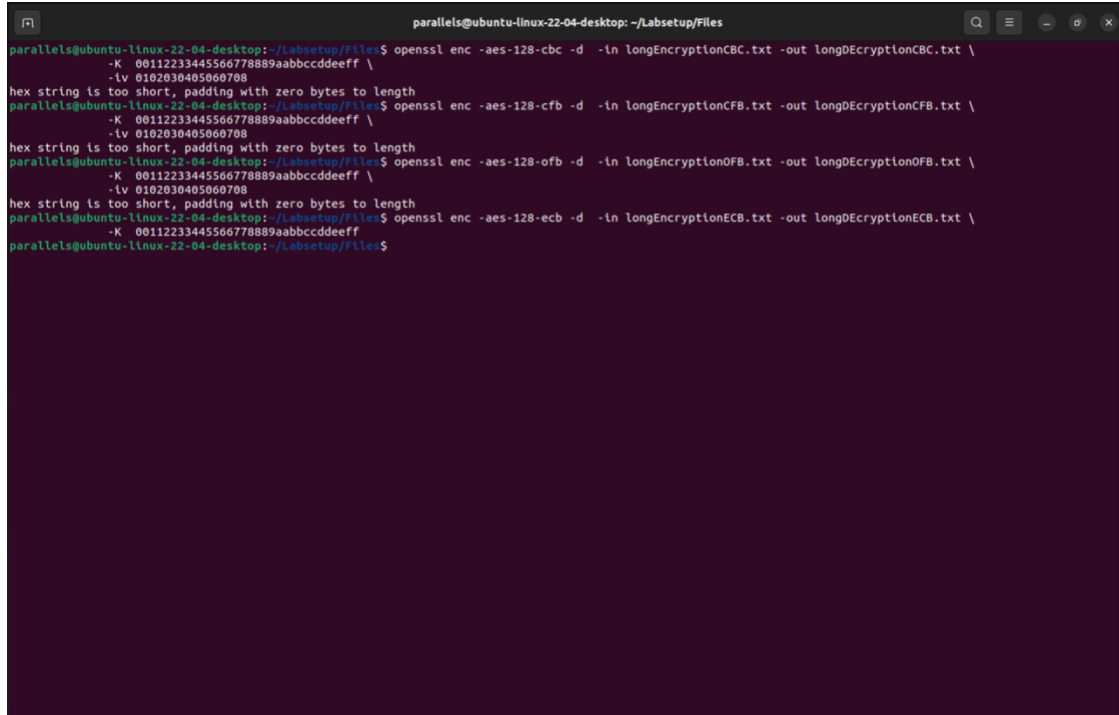
Signed 16 bit: -16363 Float 32 bit: -2.329244 Octal: 300 025 022 126

Unsigned 16 bit: 49173 Float 64 bit: -5.26790875550931 Binary: 11000000 00010101 00010010 01010110

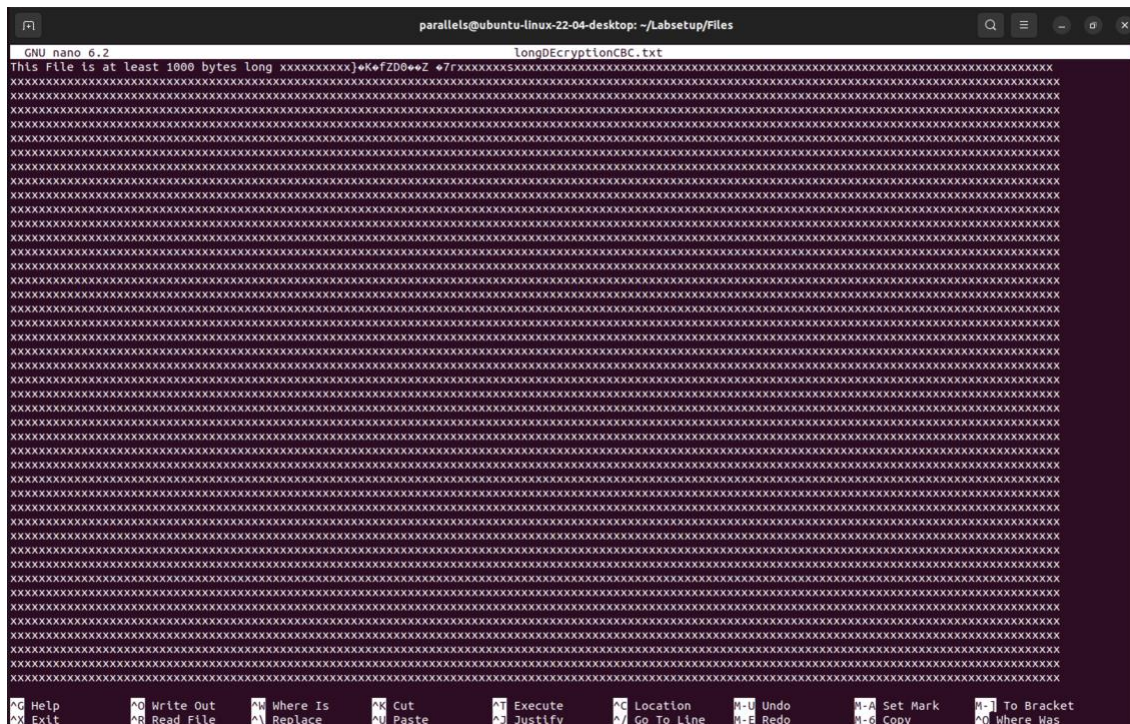
Show little endian decoding Show unsigned as hexadecimal ASCII Text: 788E9D34578035525DED5B5F3FF83AF911AF58A66B783A3E6C5FD8476FDA B00FD5

Offset: 56 / 14463 Selection: None INS

Screenshot55: ([Return to text](#))



Screenshot56: ([Return to text](#))



Screenshot57: ([Return to text](#))



Screenshot58: ([Return to text](#))



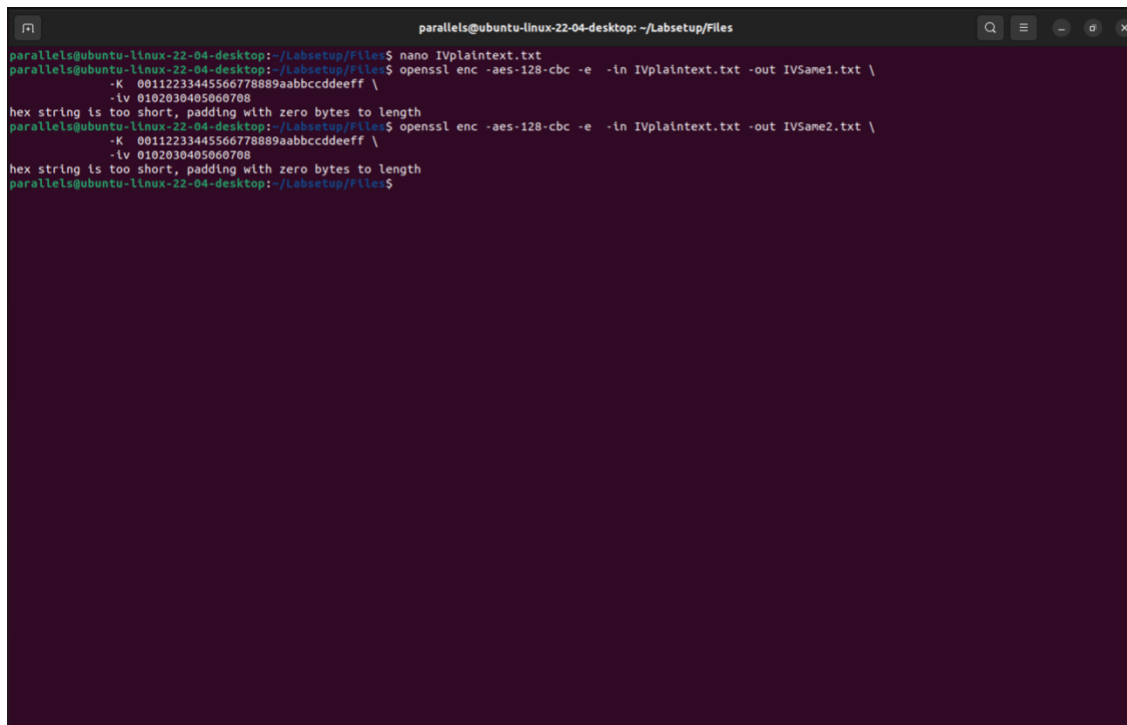
Screenshots: Task 6

Screenshot60: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
GNU nano 6.2 IVplaintext.txt
This text will be encrypted using IVs to distinguish the importance of IV uniqueness.
```

Screenshot61: ([Return to text](#))



```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$ nano IVplaintext.txt
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in IVplaintext.txt -out IVSame1.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$ openssl enc -aes-128-cbc -e -in IVplaintext.txt -out IVSame2.txt \
-K 0011223344556677889aabbccddeeff \
-iv 0102030405060708
hex string is too short, padding with zero bytes to length
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$
```


Screenshot64: ([Return to text](#))

```

GNU nano 6.2                                IVDiff.txt
S^_l*****\%$- B3e^7, *A*****HoeP=#e@44le+Nege.2.6%***Le-z)'oele**OGg[*^_nga^e+le^_SeFPee^X$!e
^G Help          ^G Write Out    ^M Where Is     ^K Cut          ^T Read 1 line  ^C Location     ^U Undo         ^-A Set Mark    ^-] To Bracket
^X Exit          ^R Read File    ^\ Replace      ^J Paste        ^_ Justify      ^_ Go To Line   ^E Redo         ^-G Copy       ^_ Where Was

```

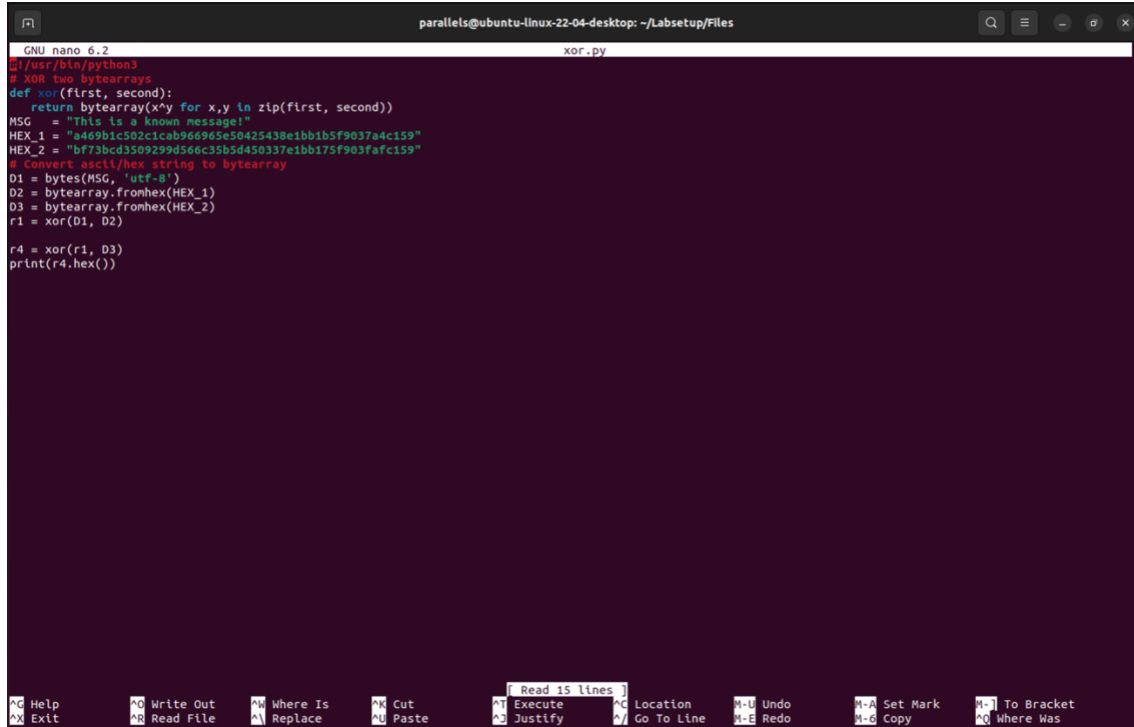
Screenshot65: ([Return to text](#))

```

GNU nano 6.2                                xor.py
#!/usr/bin/python3
# XOR two bytearrays
def xor(first, second):
    return bytearray(x^y for x,y in zip(first, second))
MSG = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"
# Convert ascii/hex string to bytearray
D1 = bytes(MSG, 'utf-8')
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)
r1 = xor(D1, D2)
r2 = xor(D2, D3)
r3 = xor(D2, D2)
print(r1.hex())
print(r2.hex())
print(r3.hex())

```

Screenshot66: ([Return to text](#))



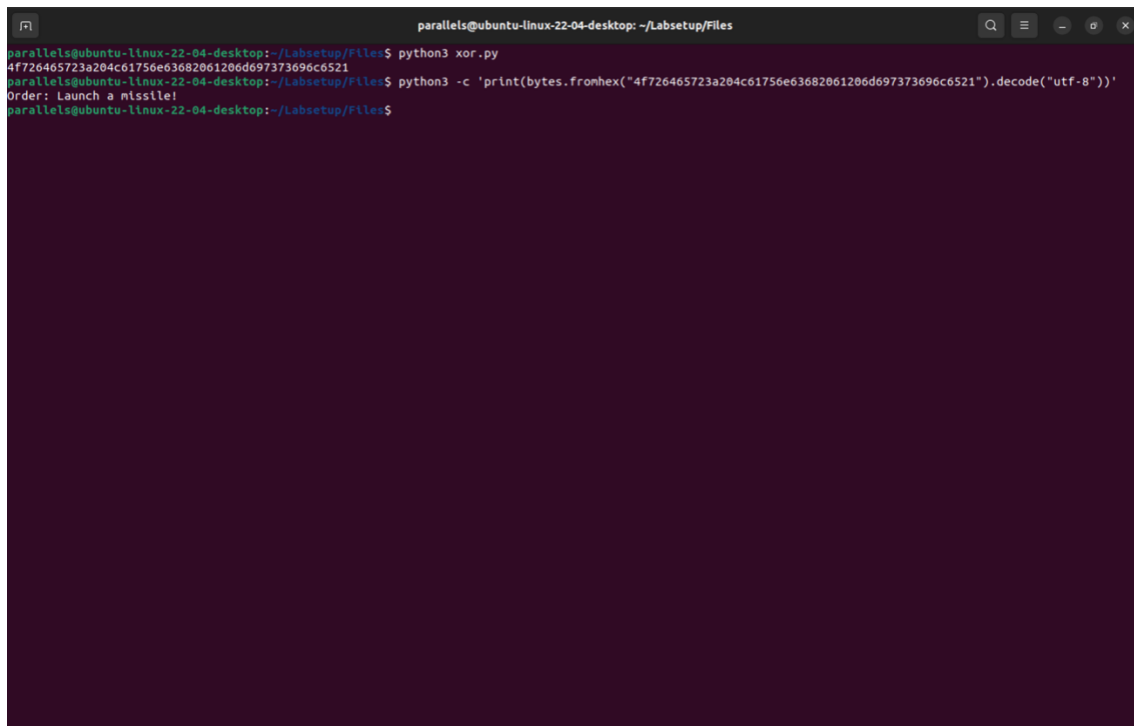
```

GNU nano 6.2                                xor.py
#!/usr/bin/python3
# XOR two bytearrays
def xor(first, second):
    return bytearray(x^y for x,y in zip(first, second))
MSG = "This is a known message!"
HEX_1 = "a4e9b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "b773bcd3509299d566c35b5d450337e1bb175f903fafc159"
# Convert ascii/hex string to bytearray
D1 = bytes(MSG, 'utf-8')
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)
r1 = xor(D1, D2)

r4 = xor(r1, D3)
print(r4.hex())
  
```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

Screenshot67: ([Return to text](#))



```

parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 xor.py
4f726465723a204c61756e63682061206d697373696c6521
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 -c 'print(bytes.fromhex("4f726465723a204c61756e63682061206d697373696c6521").decode("utf-8"))'
Order: Launch a missile!
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
  
```


Screenshot68: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 -c 'print("Yes".encode("utf-8").hex())'
596573
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 -c 'print("No".encode("utf-8").hex())'
4e6f
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$
```

Screenshot69: ([Return to text](#))

```
parallels@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 -c 'print("Yes".encode("utf-8").hex())'
596573
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ python3 -c 'print("No".encode("utf-8").hex())'
4e6f
parallels@ubuntu-linux-22-04-desktop:~/Labsetup/Files$ nc 10.9.0.80 3000
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertext: a882a422dbd82bc413b488d3980ea187
The IV used : 3a2b8af0f0ebdc7355d307d929b34856

Next IV : 447fd448f1ebdc7355d307d929b34856
Your plaintext : 596573
Your ciphertext: d30f4f38283e174ef7932fe2945a0962

Next IV : 7788fb50f1ebdc7355d307d929b34856
Your plaintext : 4e6f
Your ciphertext: c5627aa72d7a00d7eac68c602636a71a

Next IV : eb894e7df1ebdc7355d307d929b34856
Your plaintext :
```

Screenshot70: ([Return to text](#))

```

parallel@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
GNU nano 6.2
xor.py
#!/usr/bin/python3
# XOR two bytearrays
def xor(first, second):
    return bytearray(x^y for x,y in zip(first, second))
MSG1 = "Yes"
MSG2 = "No"
HEX_1 = "3a2b8af0f0ebdc7355d307d929b34856"
HEX_2 = "a882a422dbd82bc4136408d3980ea187"
HEX_3 = "447fd448f1ebdc7355d307d929b34856"
HEX_4 = "d30f4f38283e174ef7932fe2945a0962"
# Convert ascii/hex string to bytearray
D1 = bytes(MSG1, 'utf-8')
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)
D4 = bytes(MSG2, 'utf-8')
D5 = bytearray.fromhex(HEX_3)
D6 = bytearray.fromhex(HEX_4)

r1 = xor(D1, D2)
r2 = xor(r1, D3)

r3 = xor(D1, D5)
r4 = xor(r3, D6)

r5 = xor(D4, D2)
r6 = xor(r5, D3)

r7 = xor(D4, D5)
r8 = xor(r7, D6)

print("IV1 XORed with Yes and Ciphertext Value: ")
print(r2.hex())
print(bytes.fromhex(r2.hex()).decode("latin-1"))
print("\n")

print("IV2 XORed with Yes and Ciphertext Value: ")
print(r4.hex())
print(bytes.fromhex(r4.hex()).decode("latin-1"))
print("\n")

print("IV1 XORed with No and Ciphertext Value: ")
print(r6.hex())
print(bytes.fromhex(r6.hex()).decode("latin-1"))

[ Read 52 lines ]
[ Help  Write Out  Where Is  Cut  Execute  Location  Undo  Set Mark  To Bracket
  Exit  Read File  Replace  Paste  Justify  Go To Line  Redo  Copy  Where Was ]

```

Screenshot71: ([Return to text](#))

```

parallel@ubuntu-linux-22-04-desktop: ~/Labsetup/Files
parallel@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$ nano xor.py
parallel@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$ python3 xor.py
IV1 XORed with Yes and Ciphertext Value:
cbcc5d
Ëi]

IV2 XORed with Yes and Ciphertext Value:
ce15e8
îè

IV1 XORed with No and Ciphertext Value:
dcc6
Uæ

IV2 XORed with No and Ciphertext Value:
d91f
Û

parallel@ubuntu-linux-22-04-desktop: ~/Labsetup/Files$

```