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

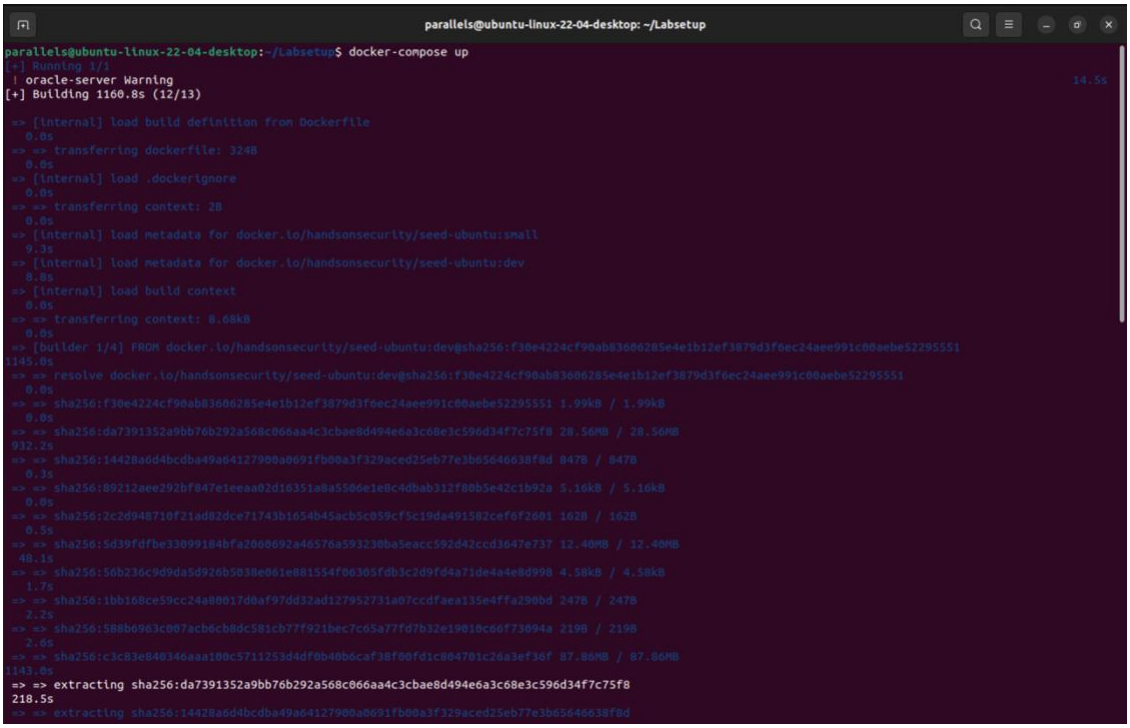
Mitre. (June 29, 2023). *CWE-1204: Generation of weak initialization vector (IV)*. Common Weakness Enumeration. https://cwe.mitre.org/data/definitions/1204.html

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

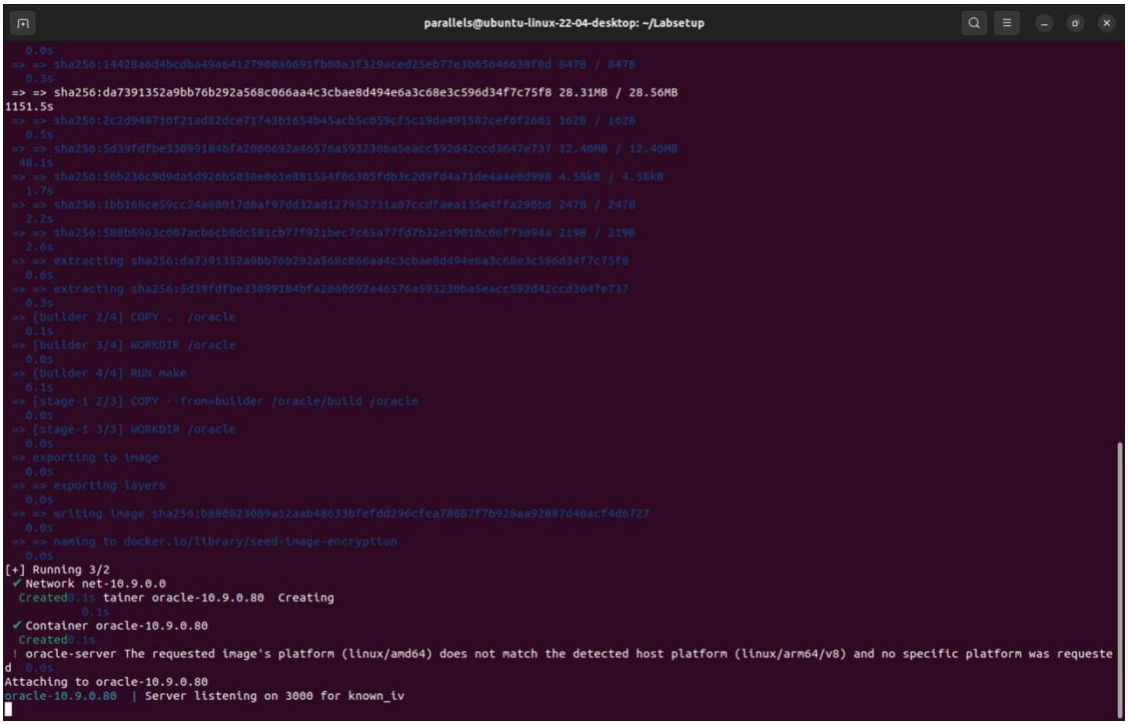
University of Notre Dame. (n.d.). *Frequency of letters of the alphabet in English*. University of Notre Dame. https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html

Screenshots: Task 1

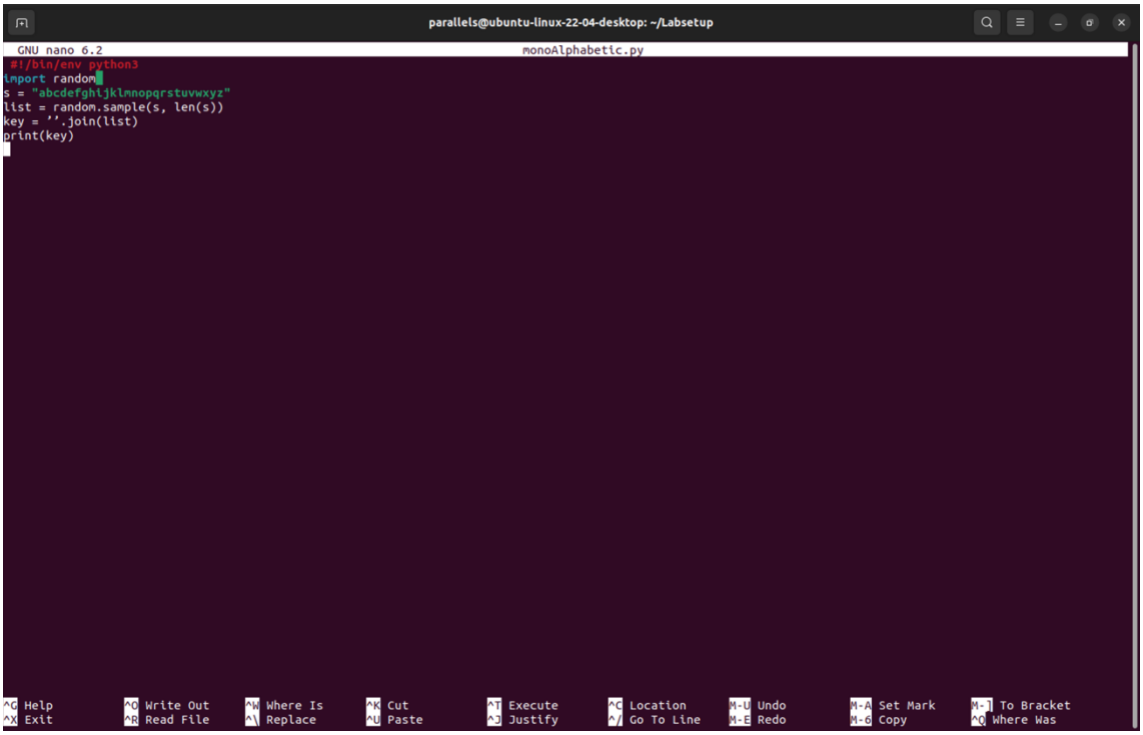
Screenshot1: (Return to text)



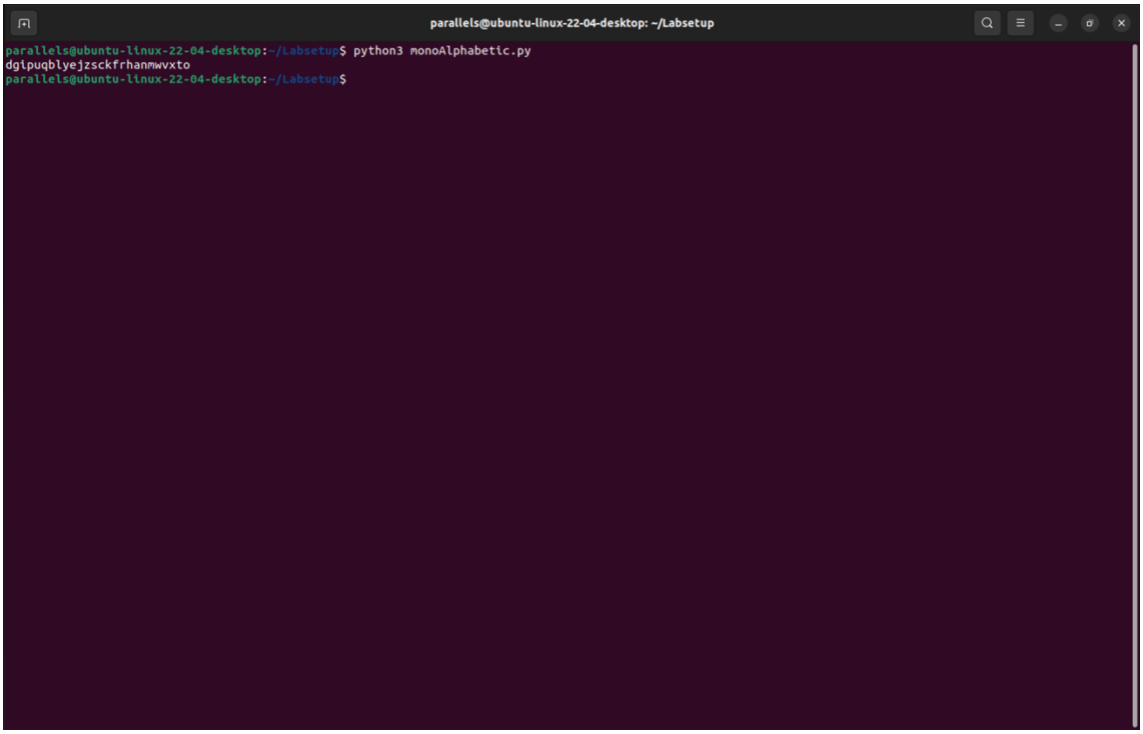
Screenshot2: (Return to text)



Screenshot3: (Return to text)

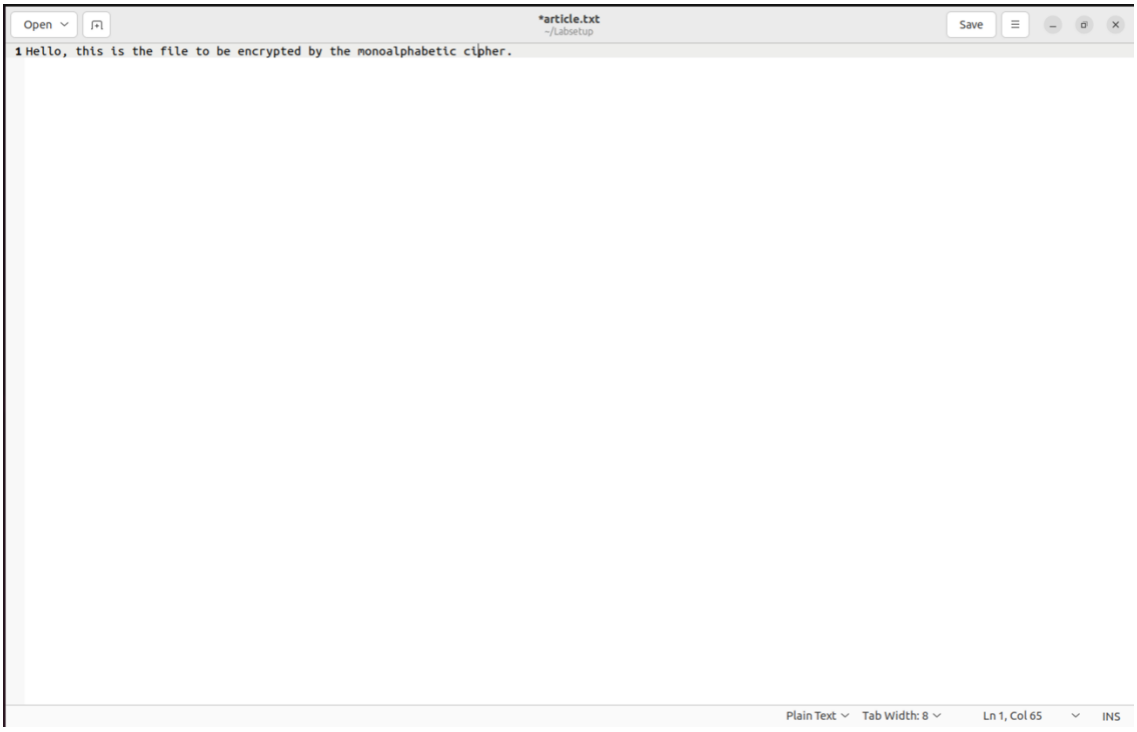


Screenshot4: (Return to text)

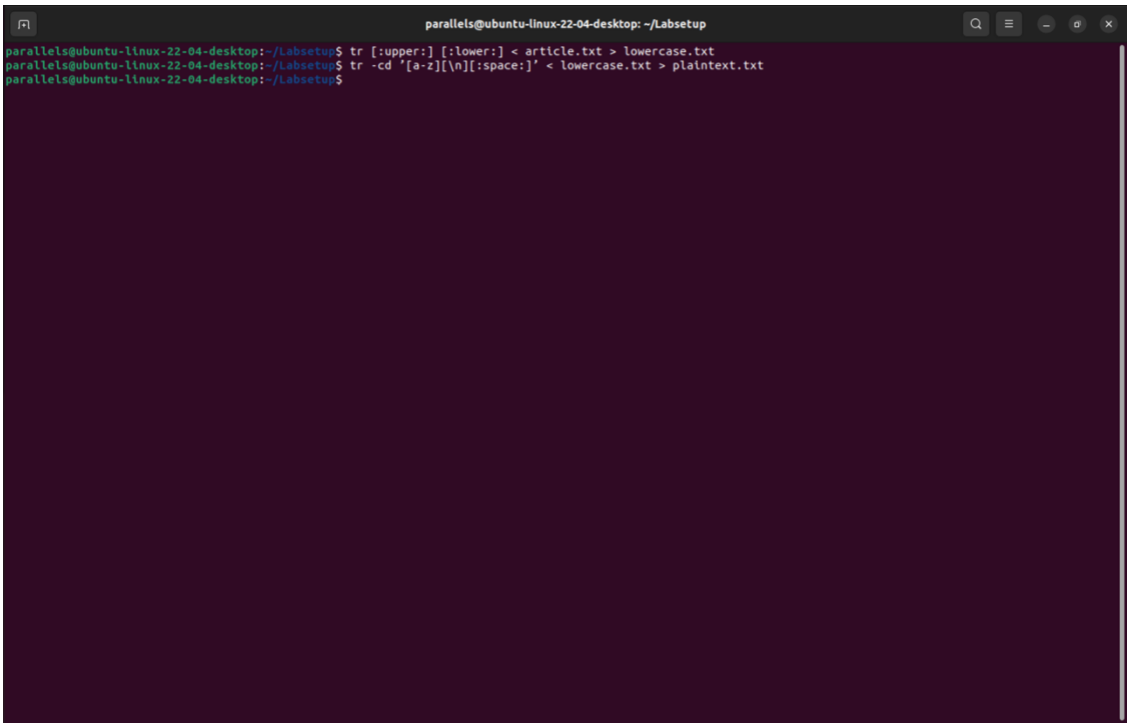


Screenshots: Task 2

Screenshot5: (Return to text)

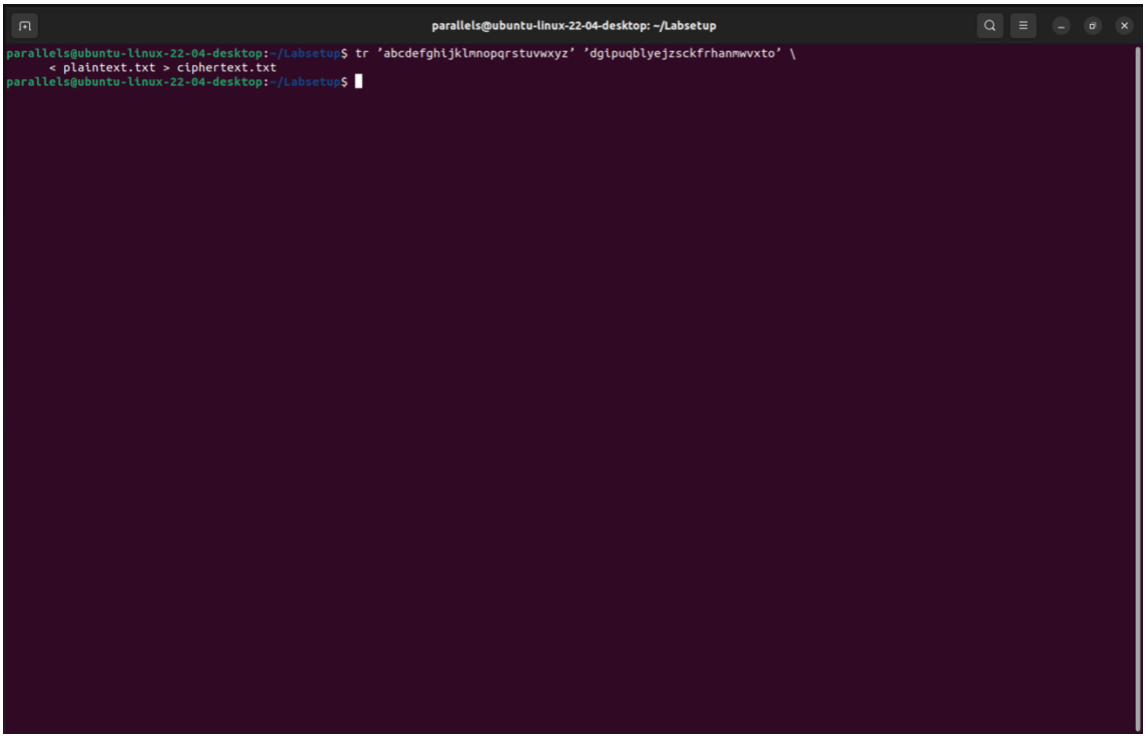


Screenshot6: (Return to text)



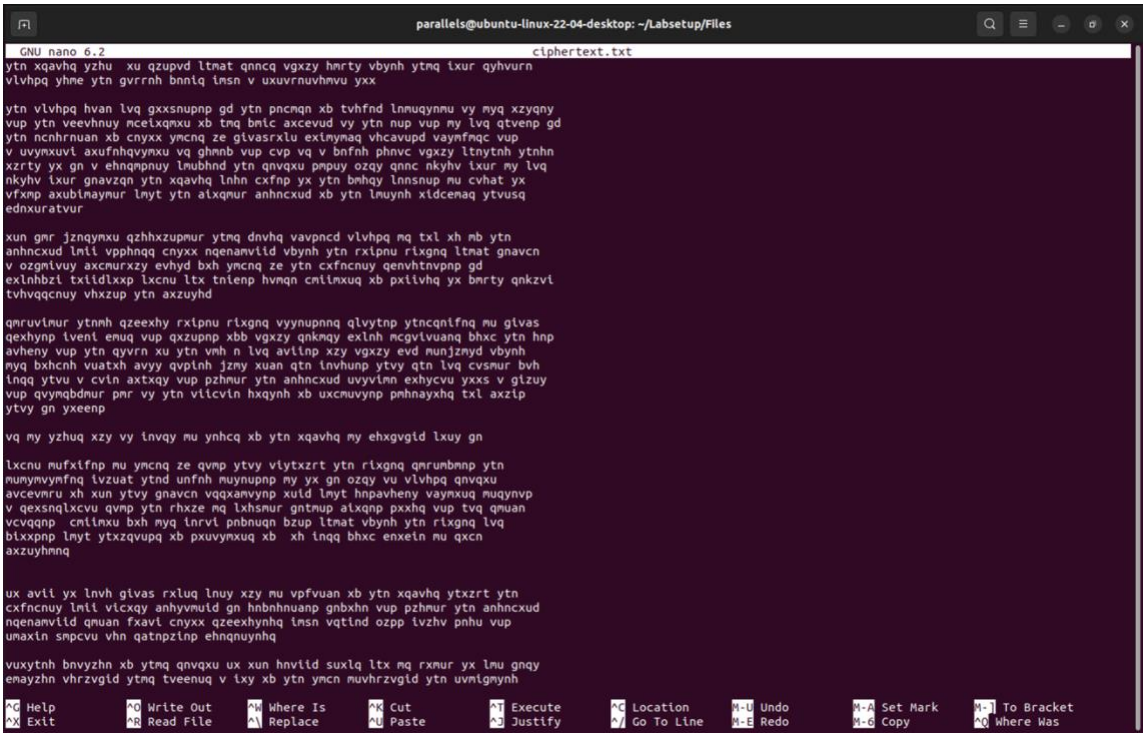
Screenshot7: (Return to text)

Screenshot8: (Return to text)

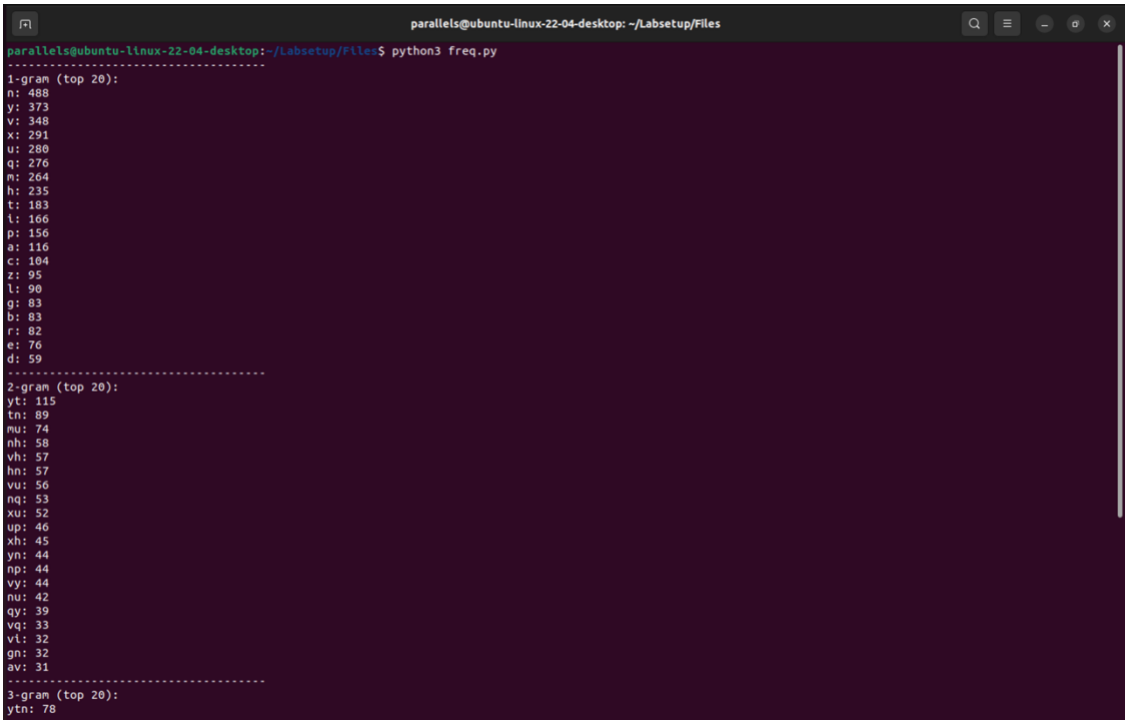


Screenshot9: (Return to text)

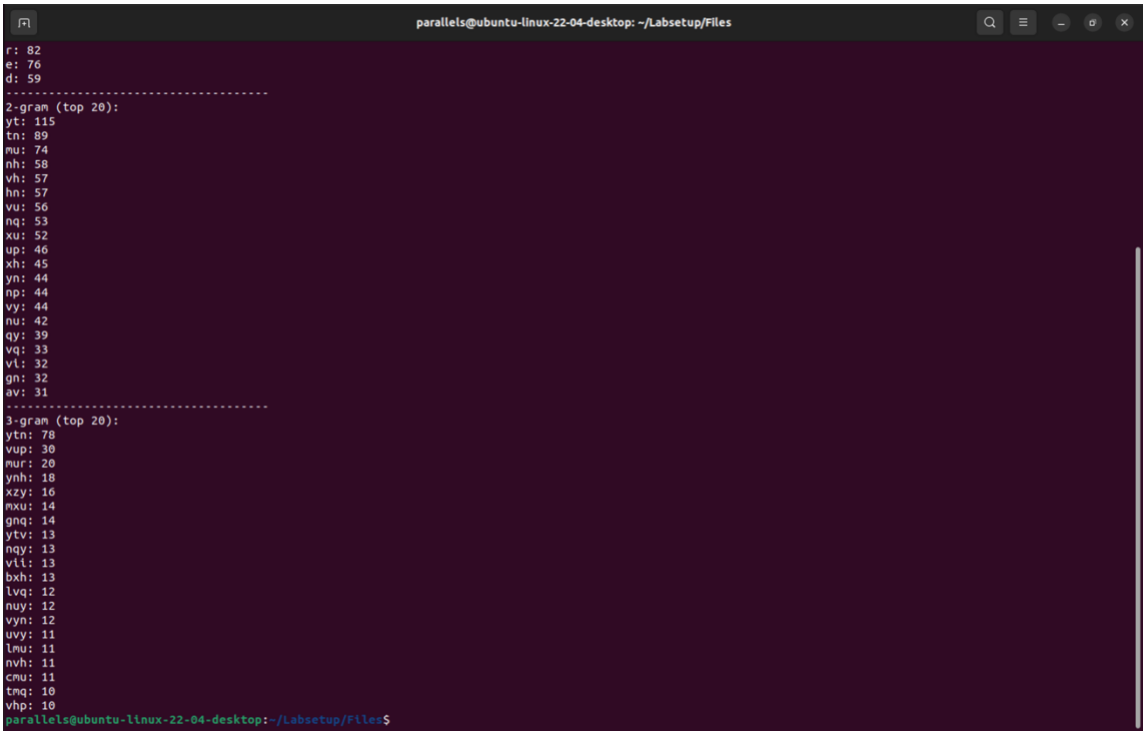
Screenshot10: (Return to text)



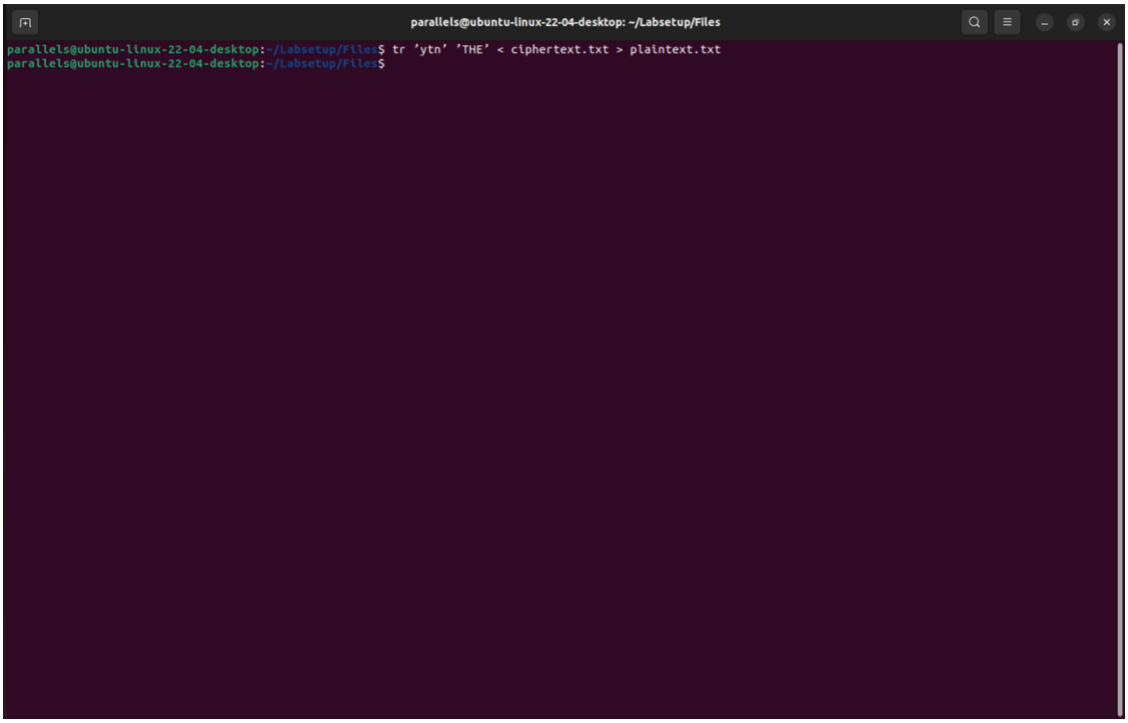
Screenshot11: (Return to text)



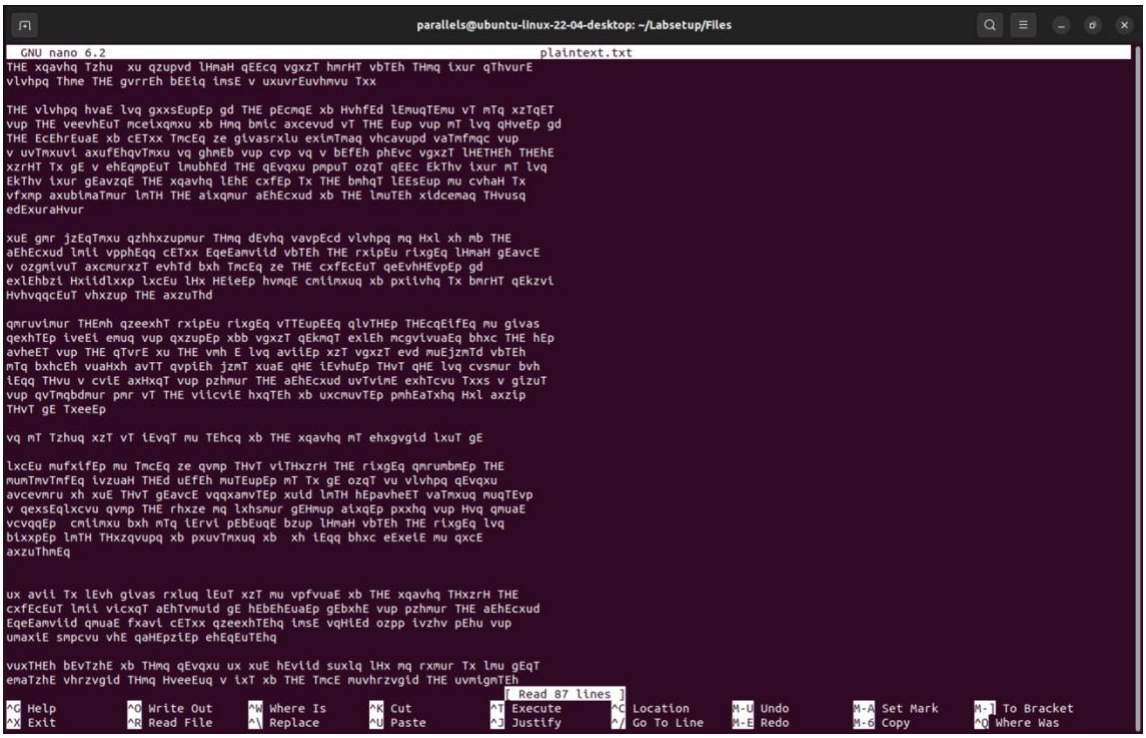
Screenshot12: (Return to text)



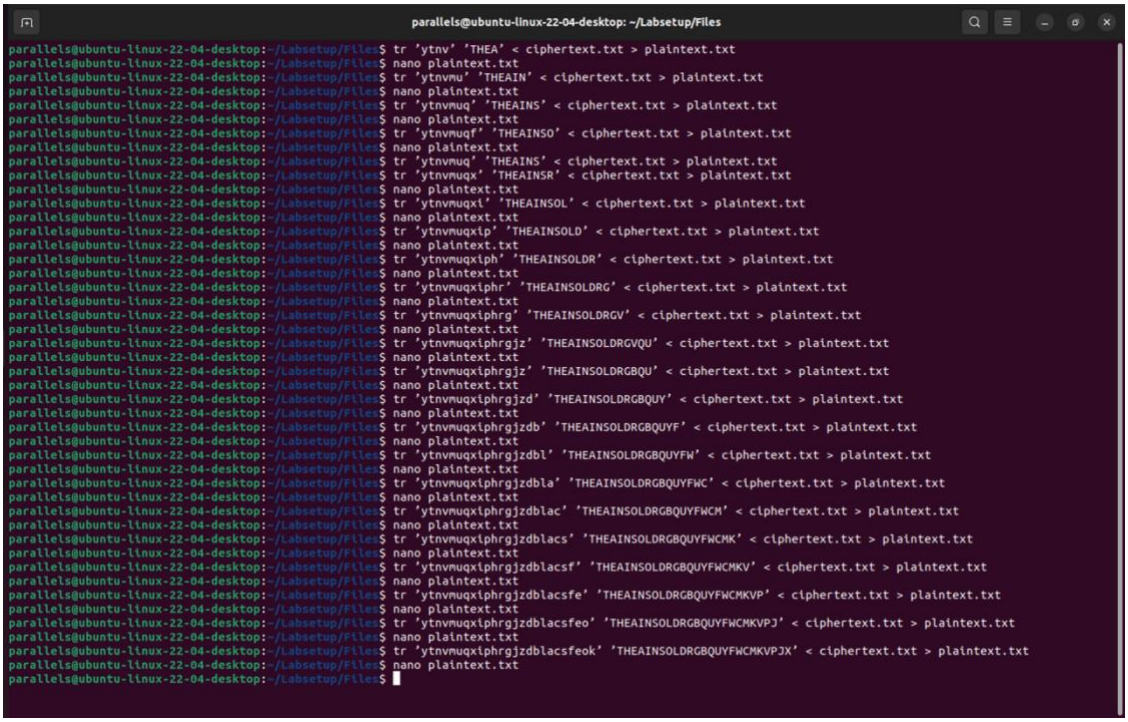
Screenshot13: (Return to text)



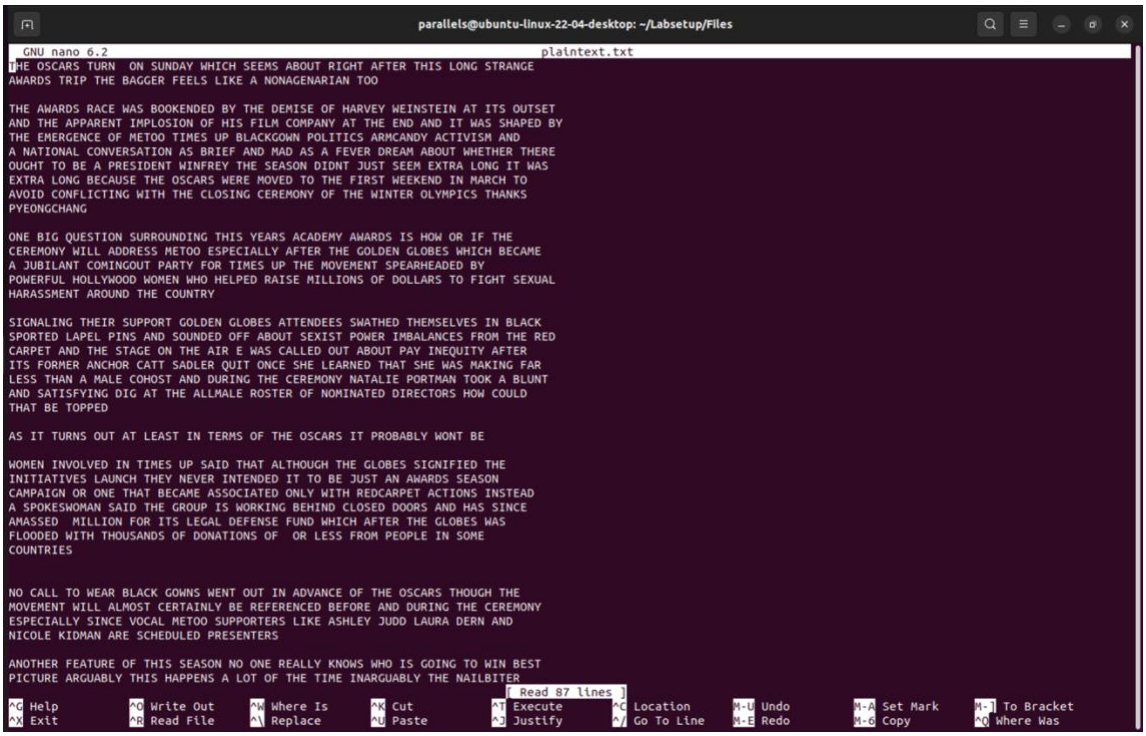
Screenshot14: (Return to text)



Screenshot15: (Return to text)

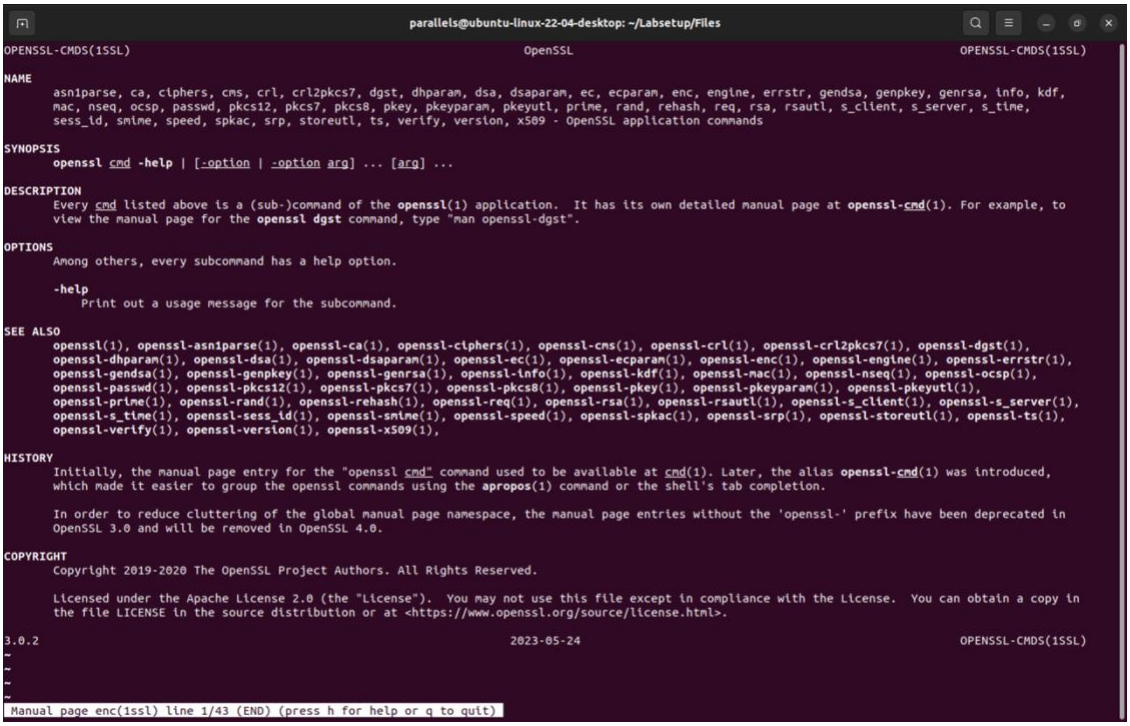


Screenshot16: (Return to text)

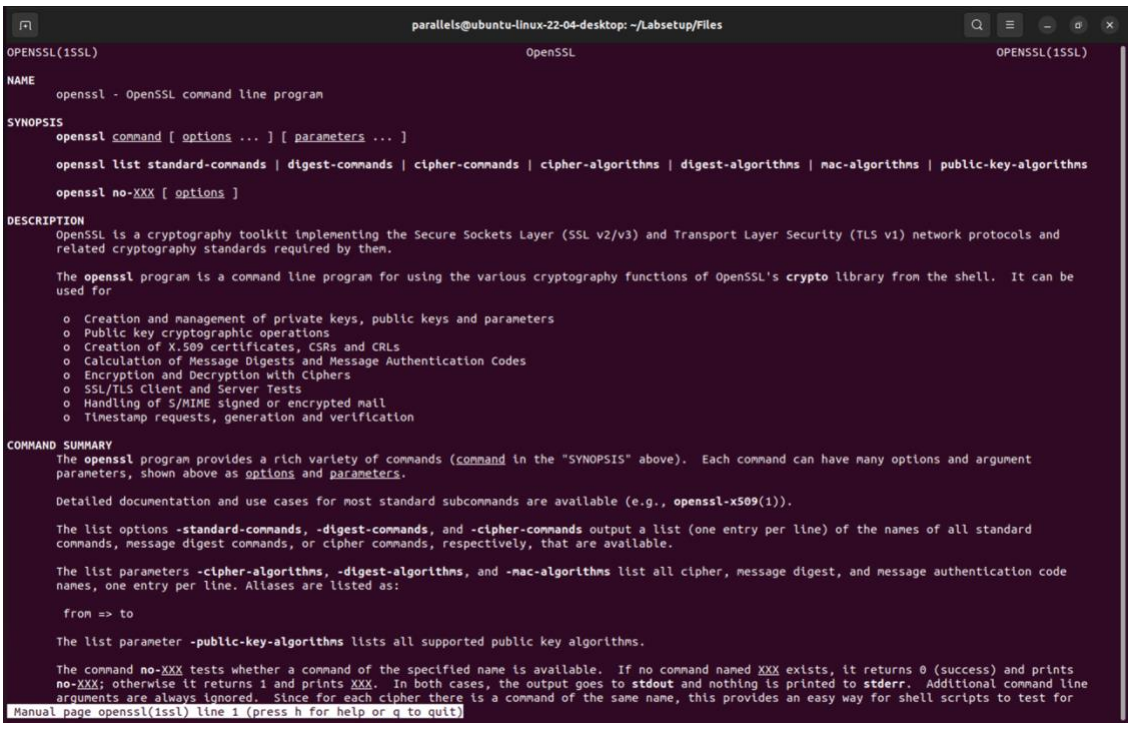


Screenshots: Task 2

Screenshot17: (Return to text)

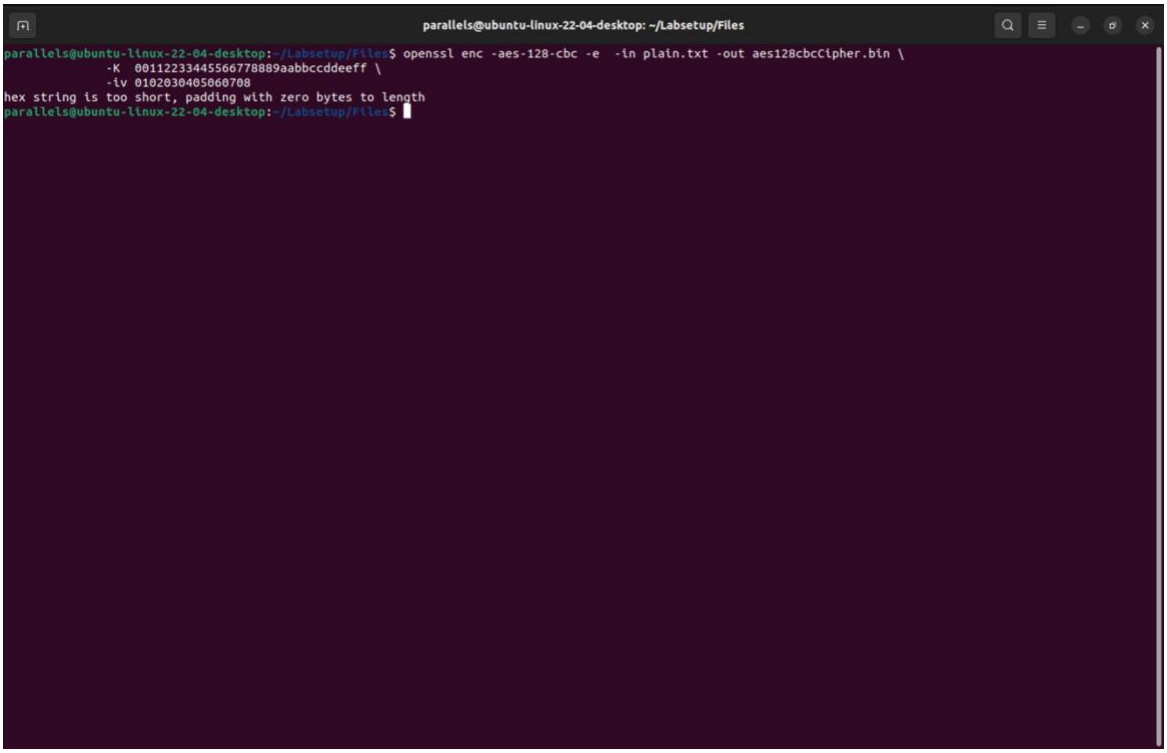


Screenshot18: (Return to text)



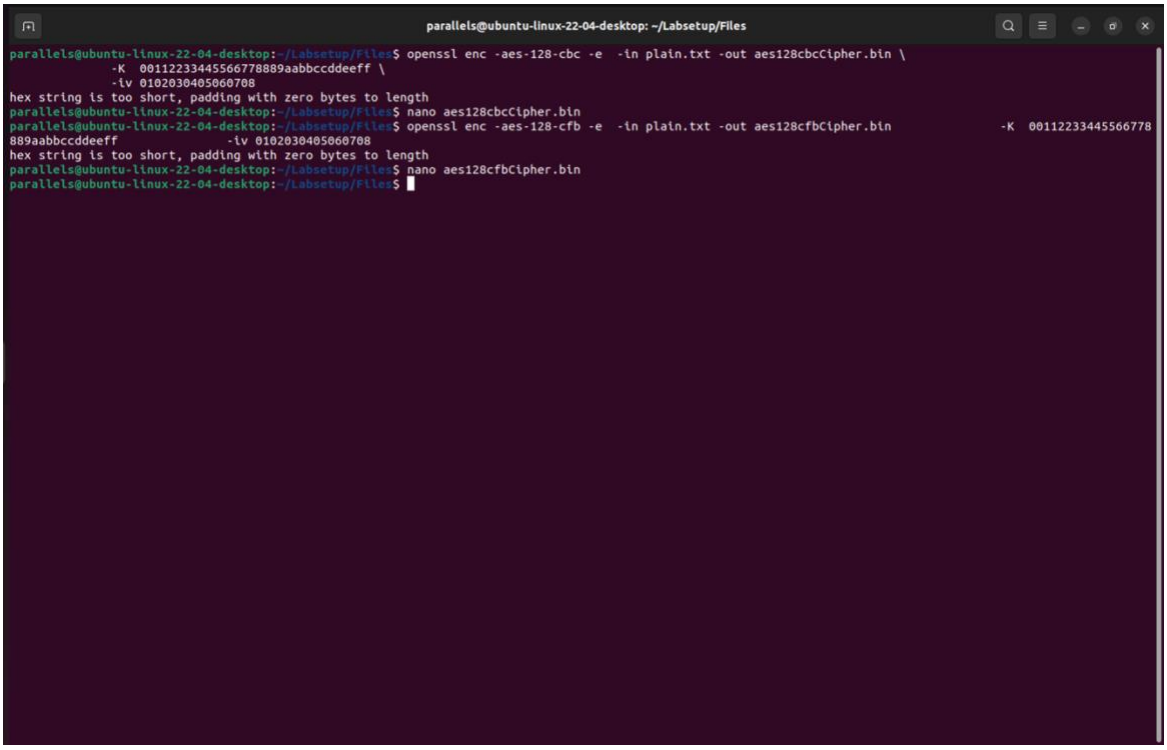
Screenshot19: (Return to text)

Screenshot20: (Return to text)



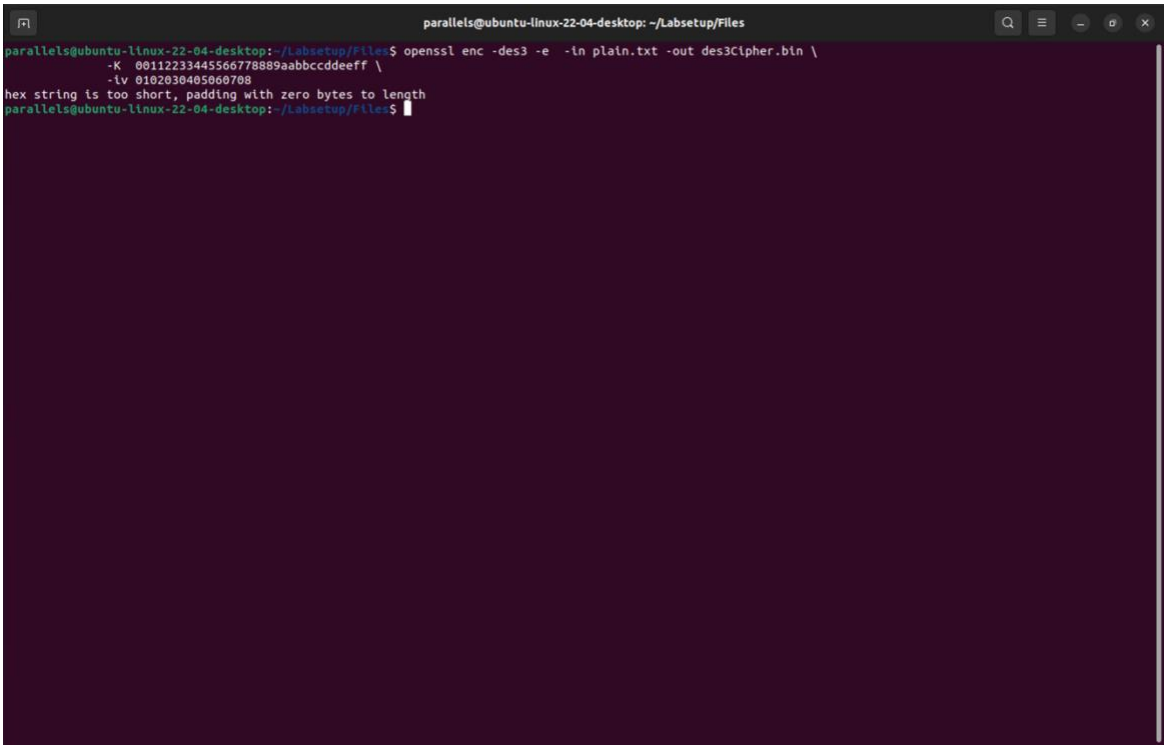
Screenshot21: (Return to text)

Screenshot22: (Return to text)



Screenshot23: (Return to text)

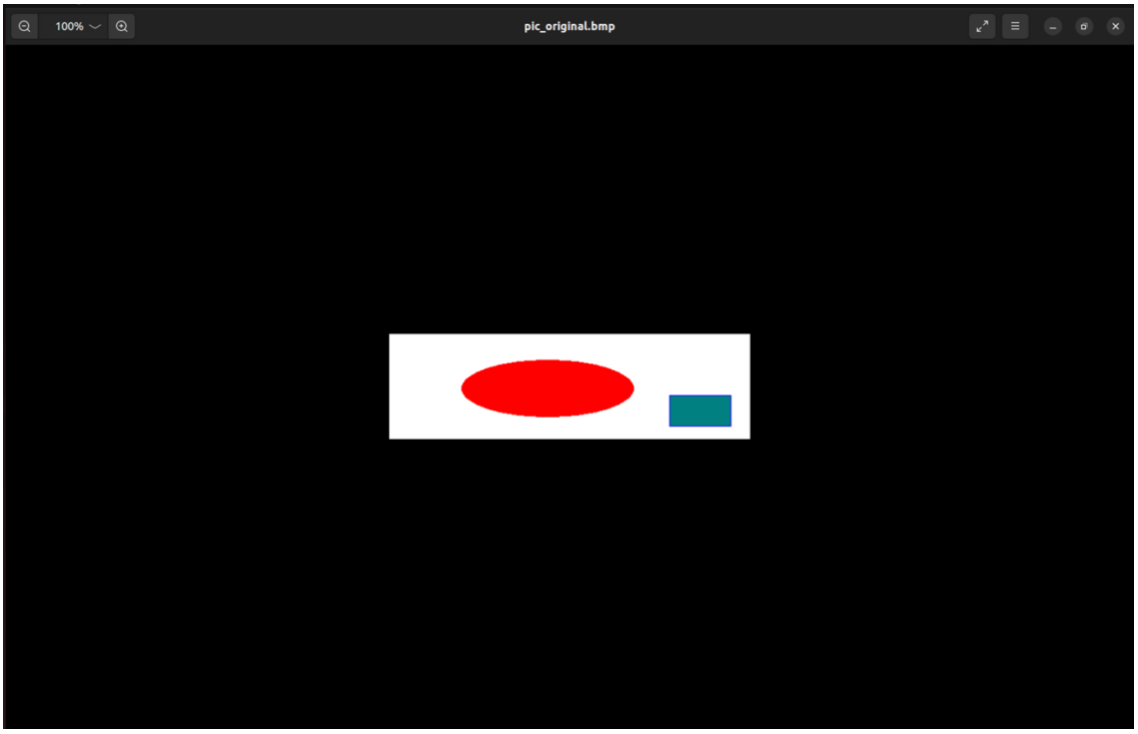
Screenshot24: (Return to text)



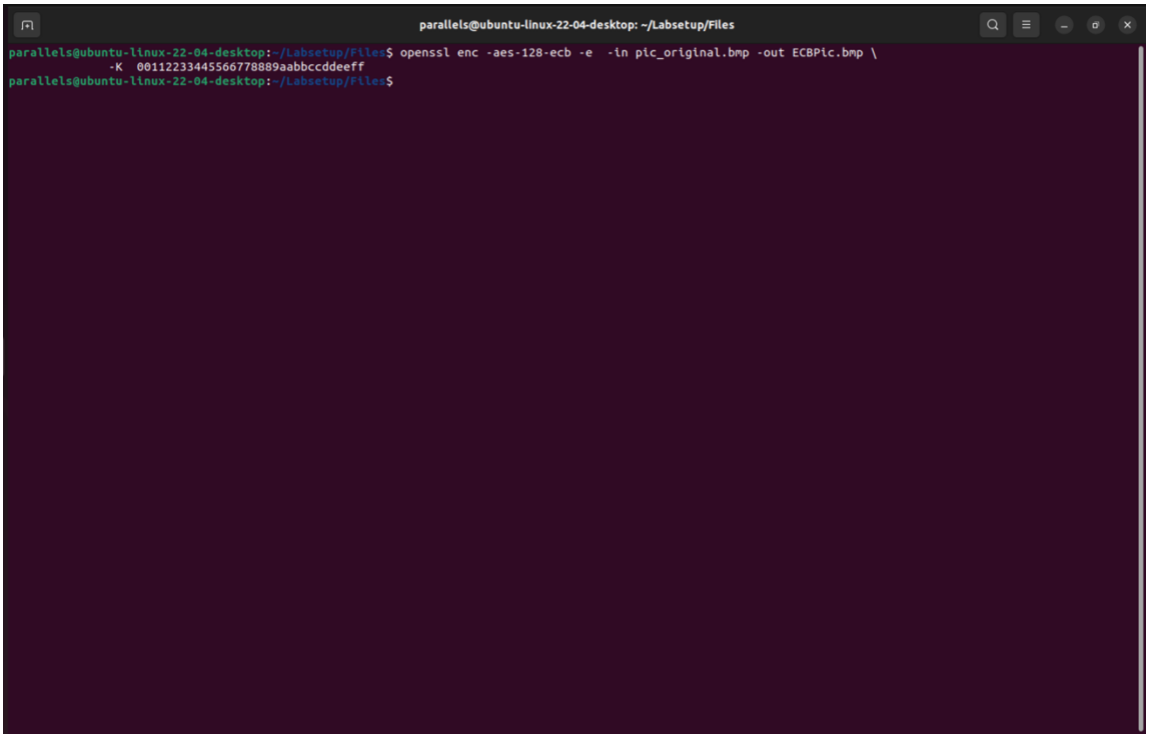
Screenshot25: (Return to text)

Screenshots: Task 3

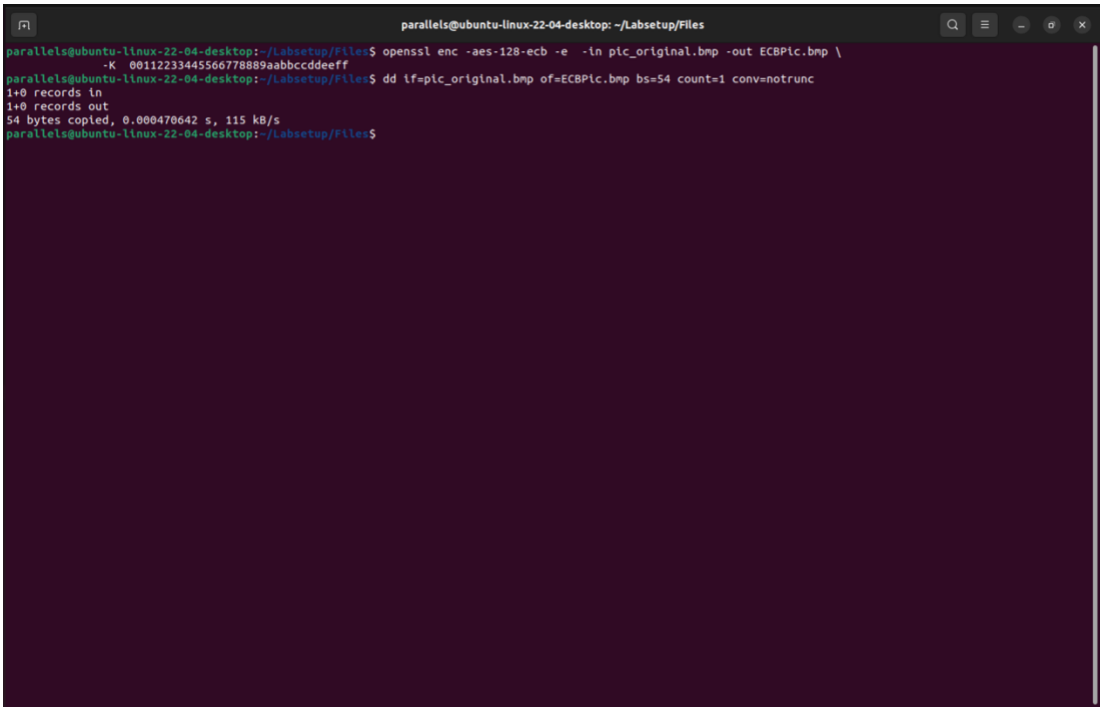
Screenshot26: (Return to text)



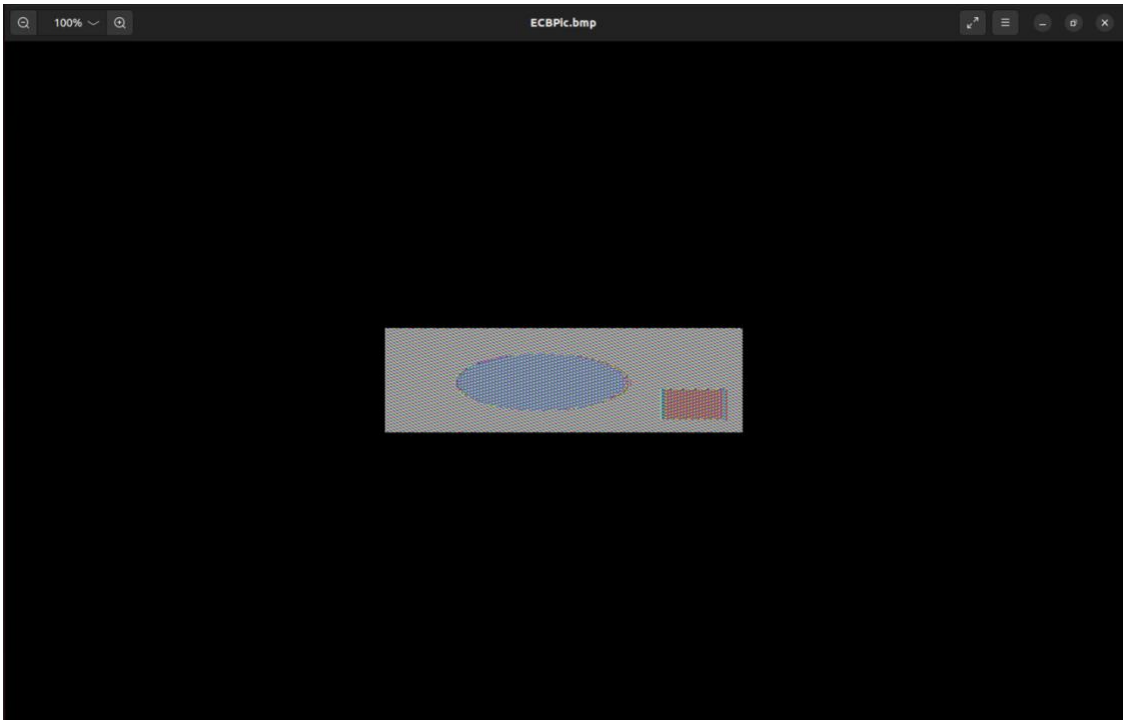
Screenshot27: (Return to text)



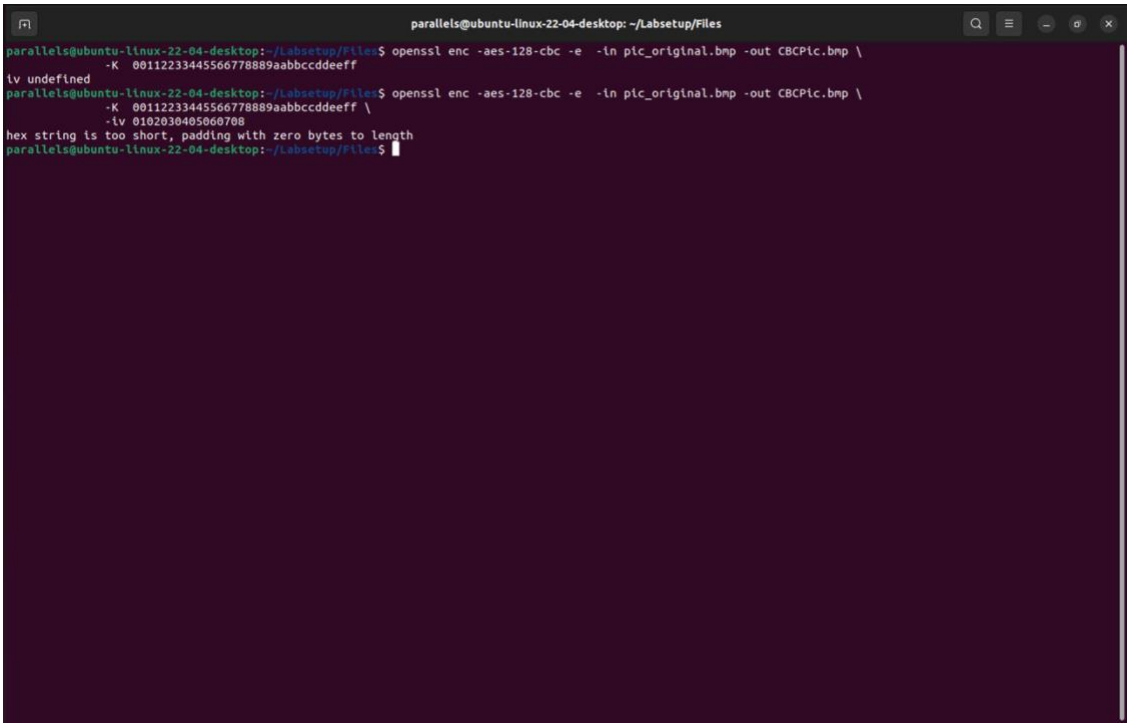
Screenshot28: (Return to text)



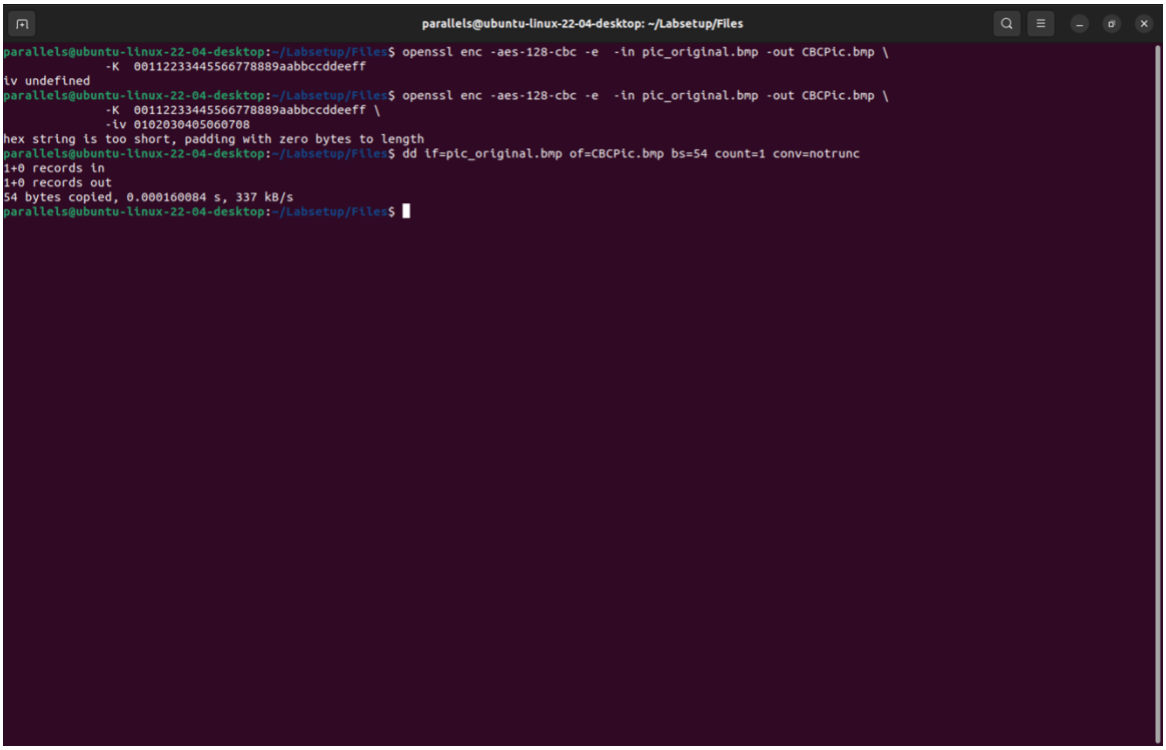
Screenshot29: (Return to text)



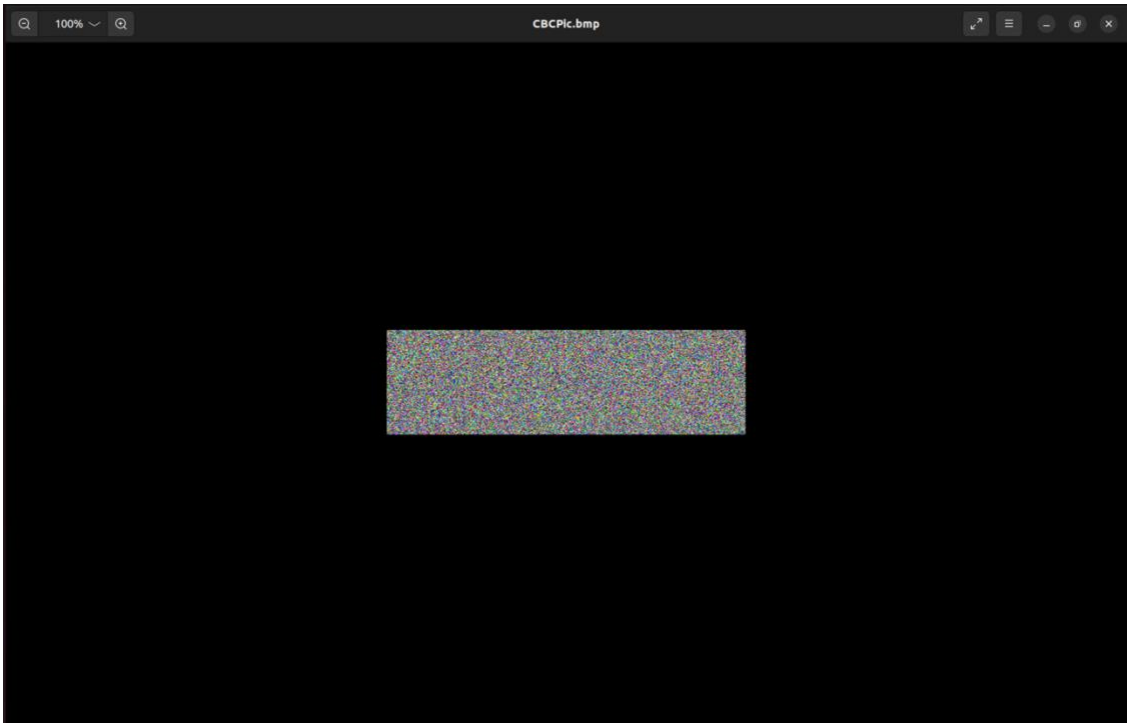
Screenshot30: (Return to text)



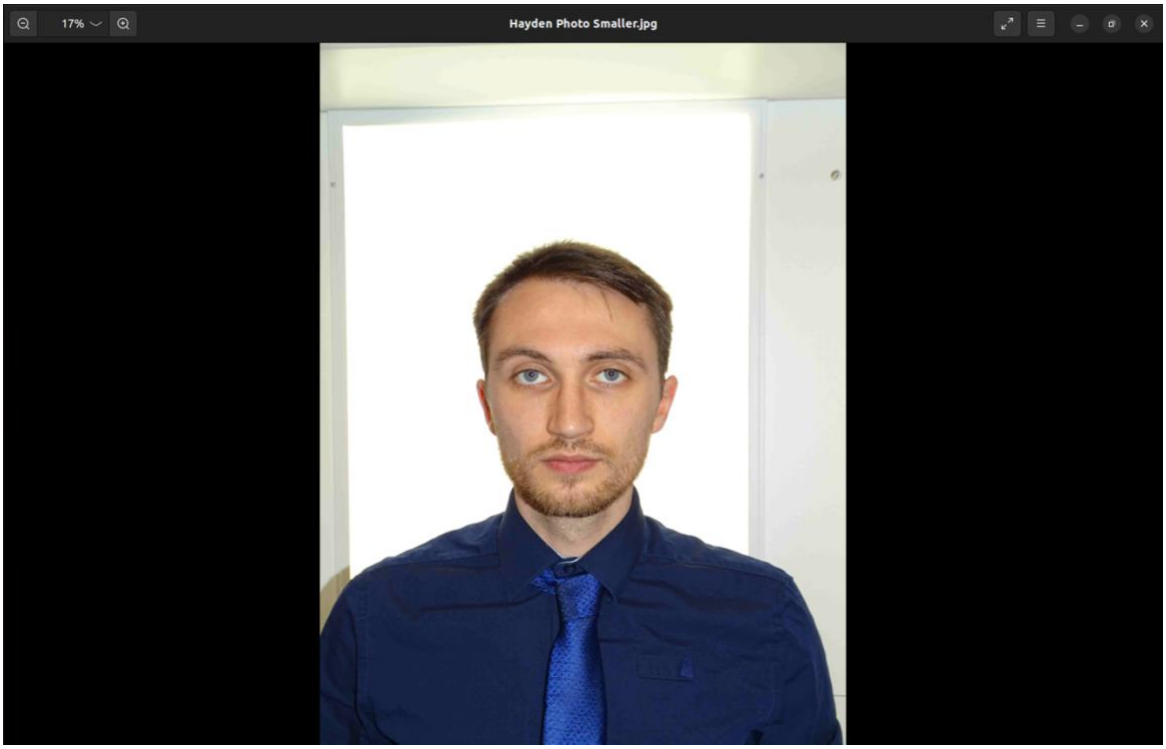
Screenshot31: (Return to text)



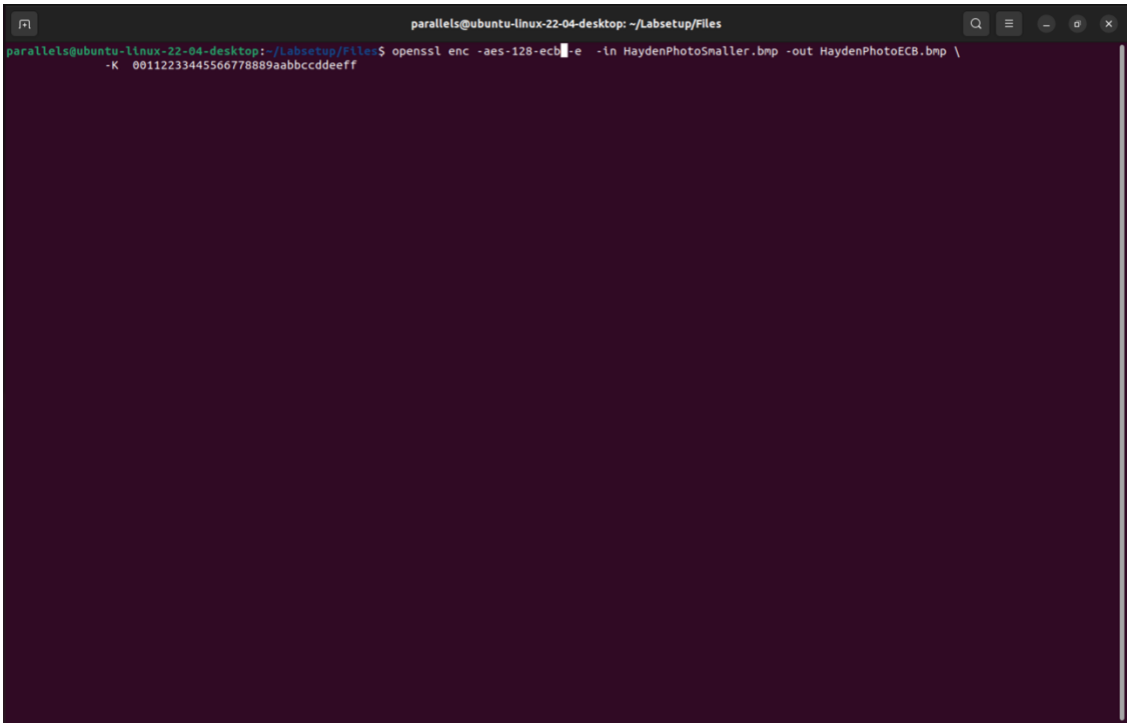
Screenshot32: (Return to text)



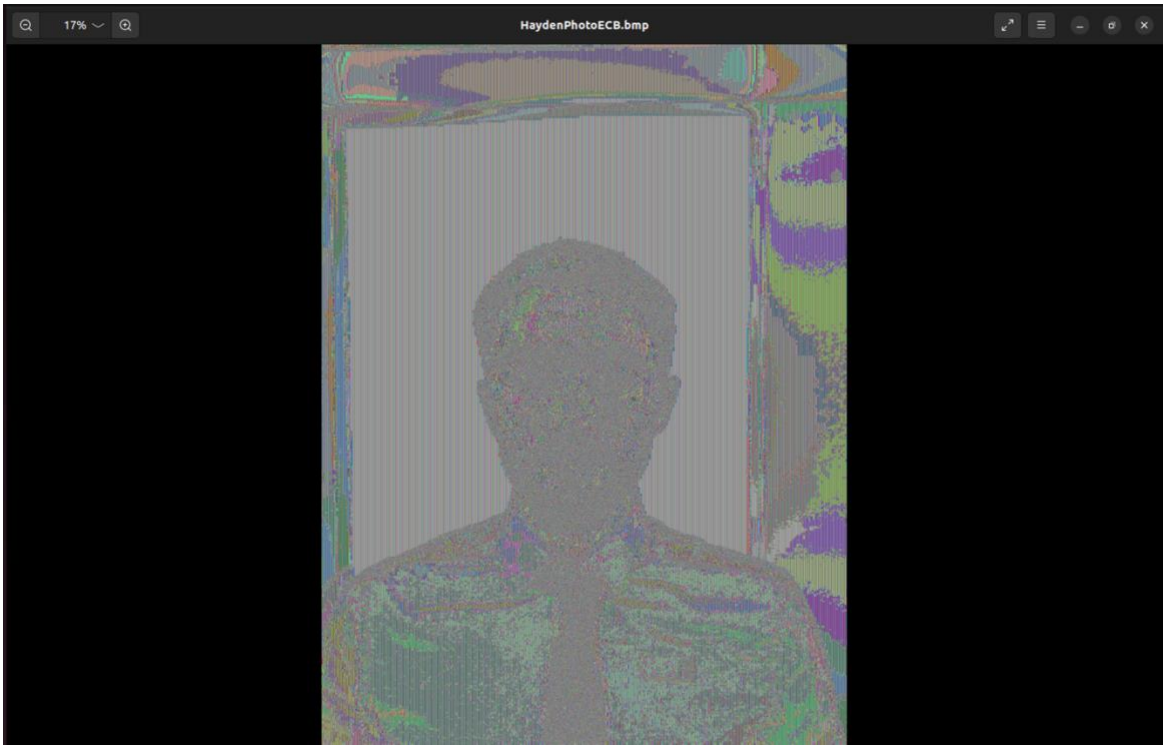
Screenshot33: (Return to text)



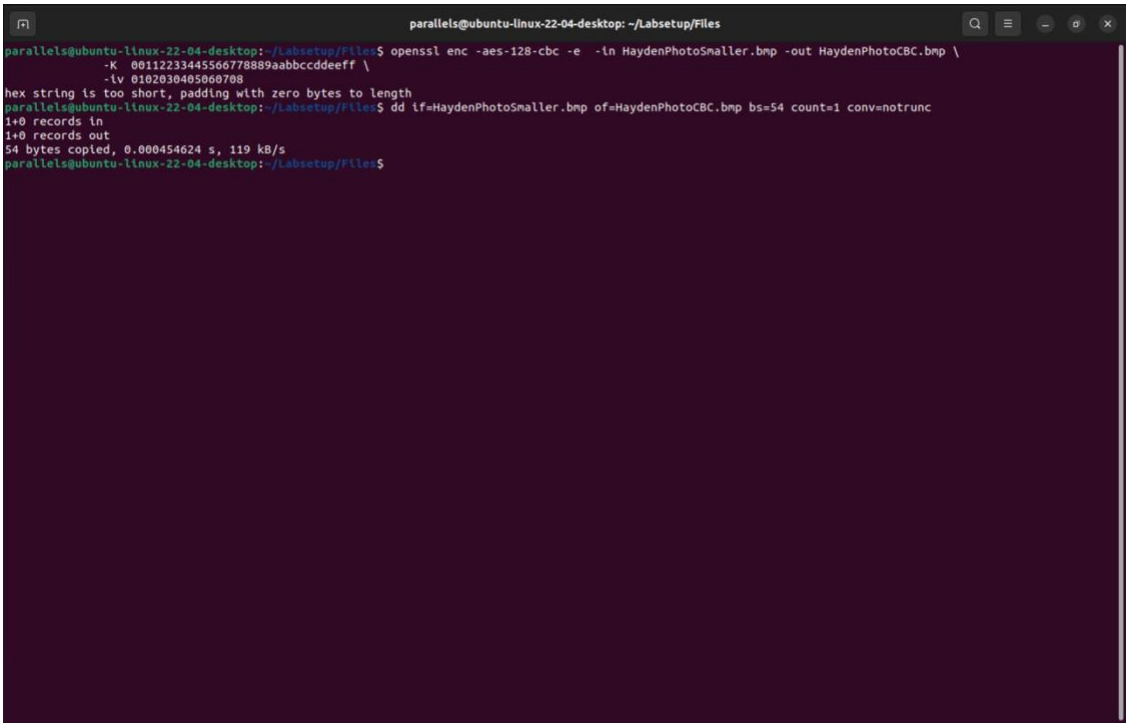
Screenshot34: (Return to text)



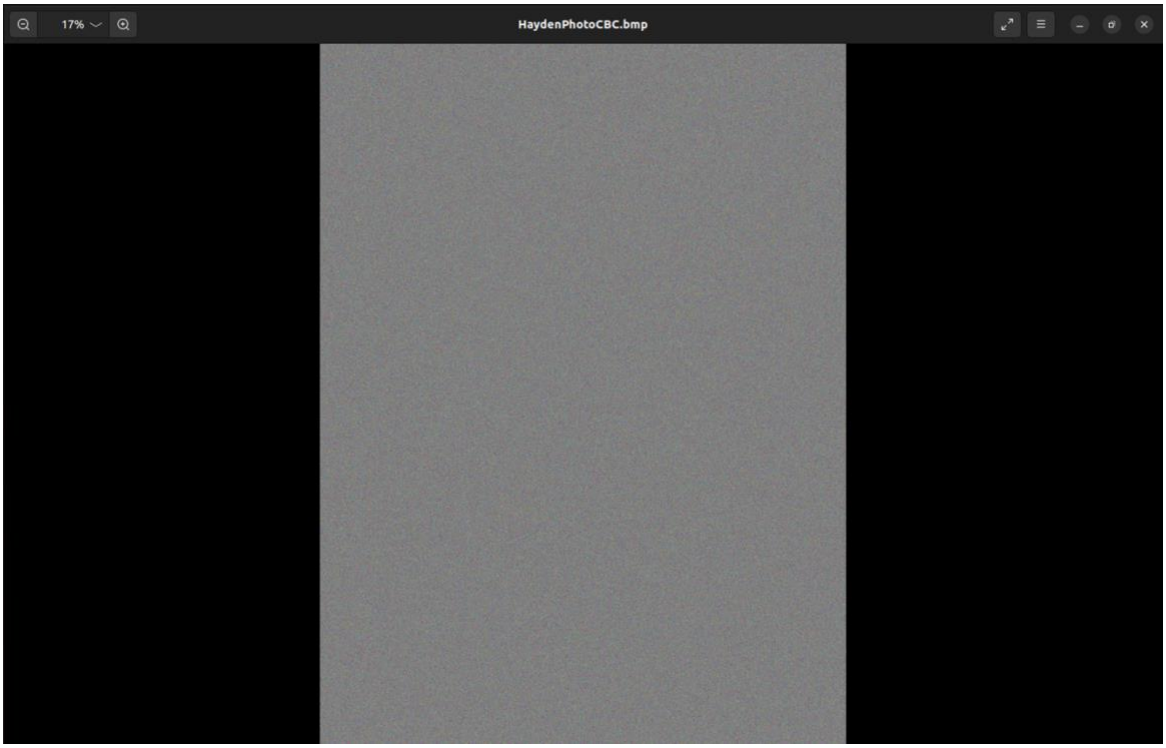
Screenshot35: (Return to text)



Screenshot36: (Return to text)

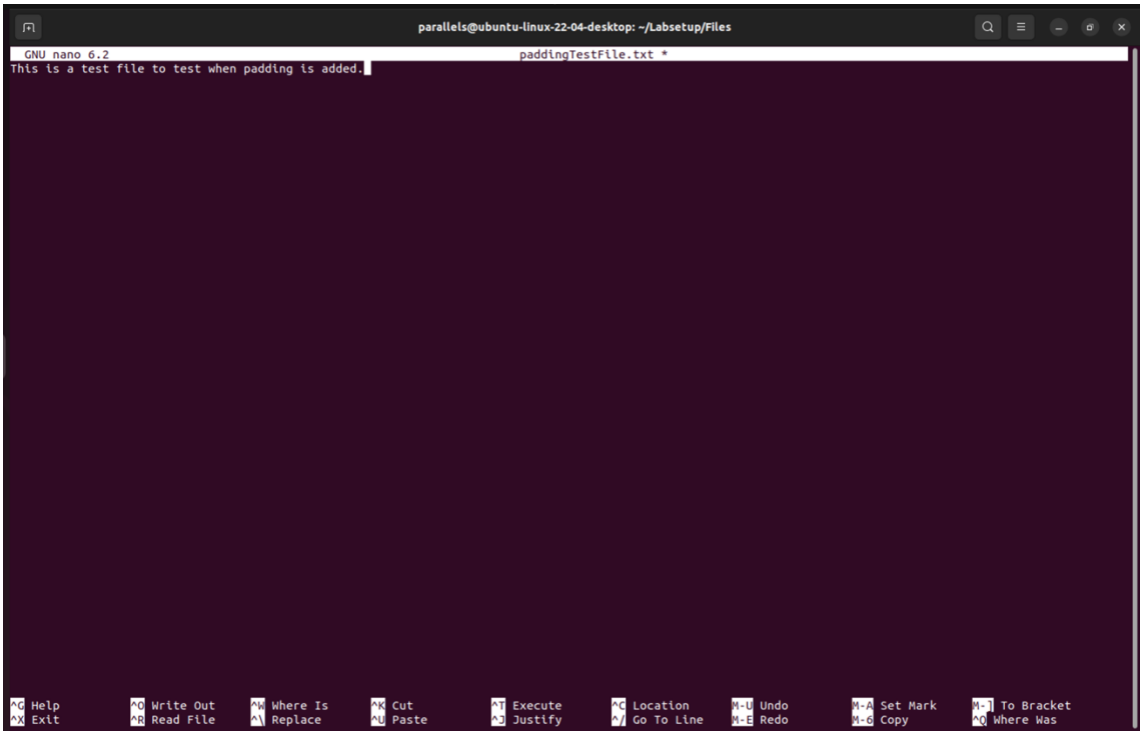


Screenshot37: (Return to text)

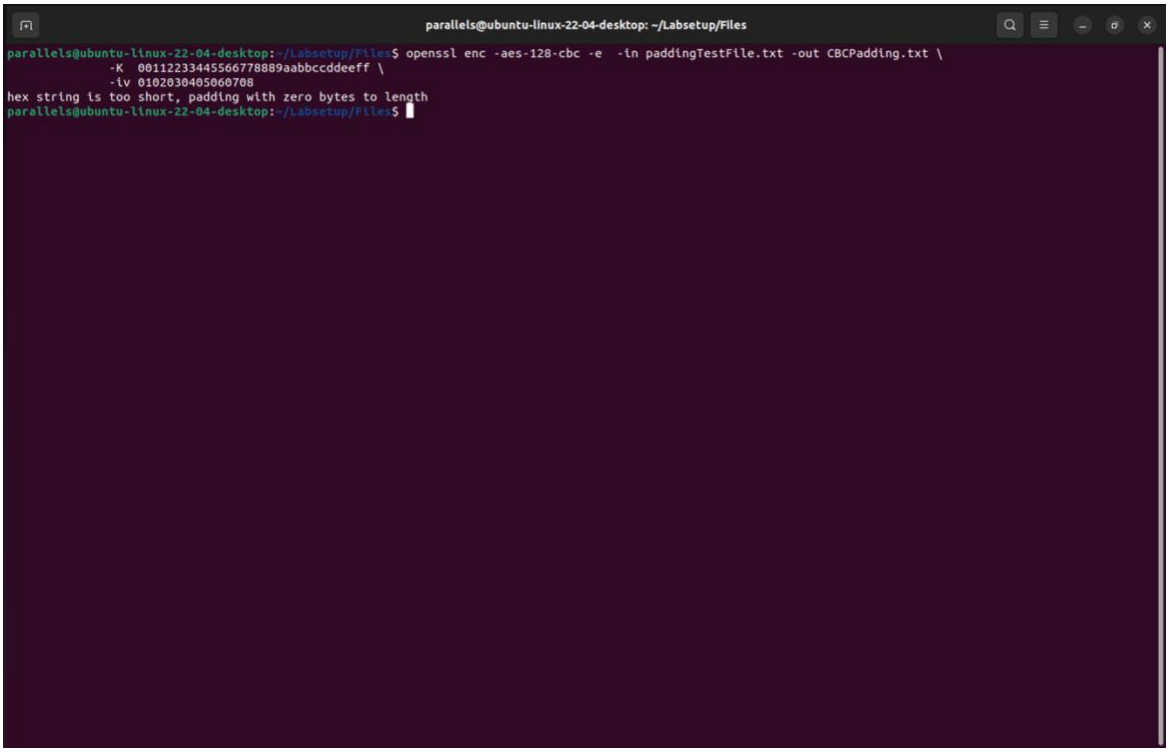


Screenshots: Task 4

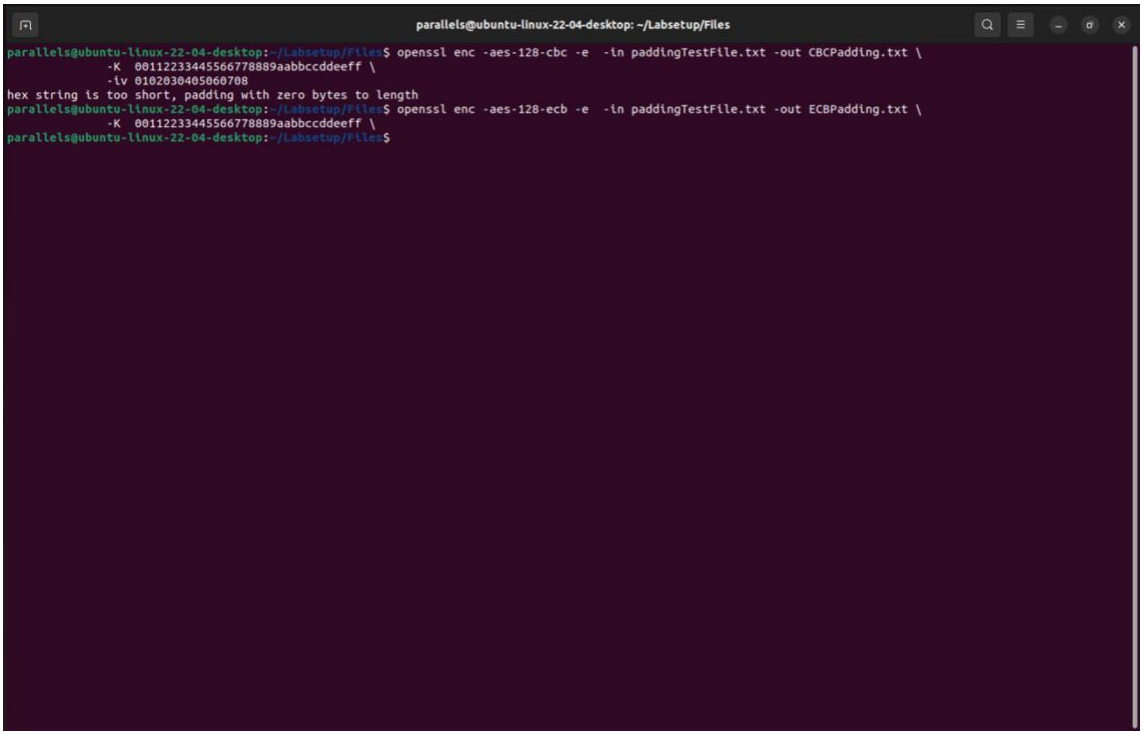
Screenshot38: (Return to text)



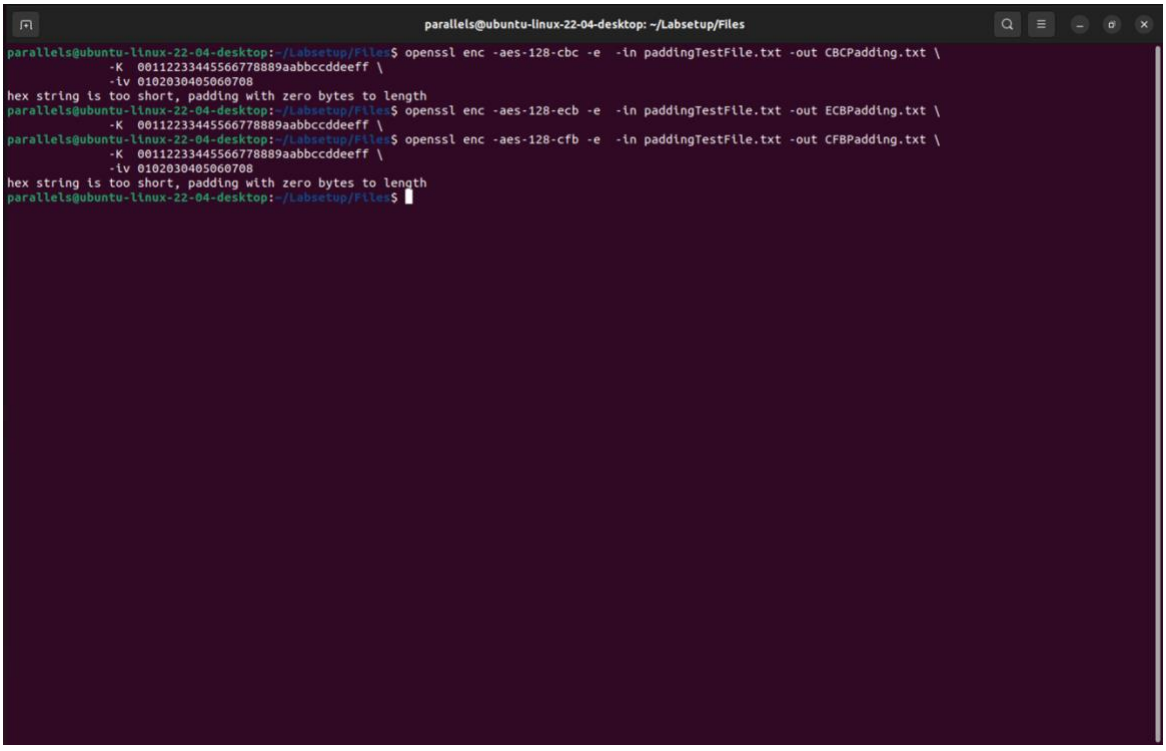
Screenshot39: (Return to text)



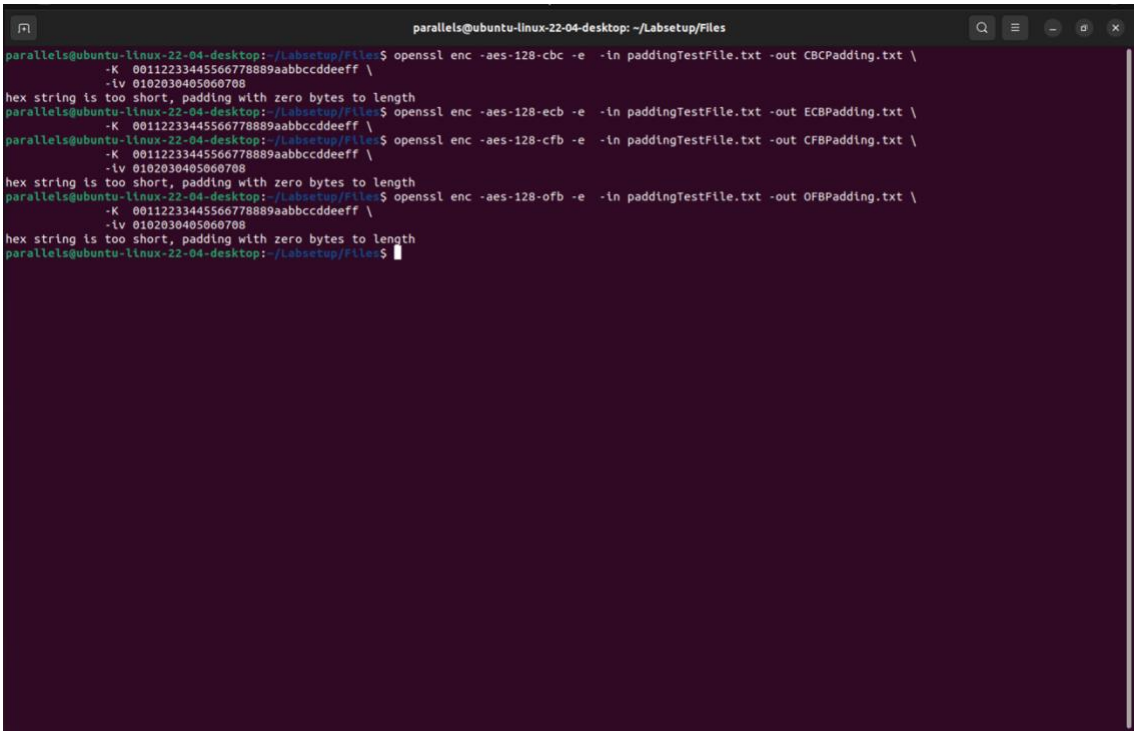
Screenshot40: (Return to text)



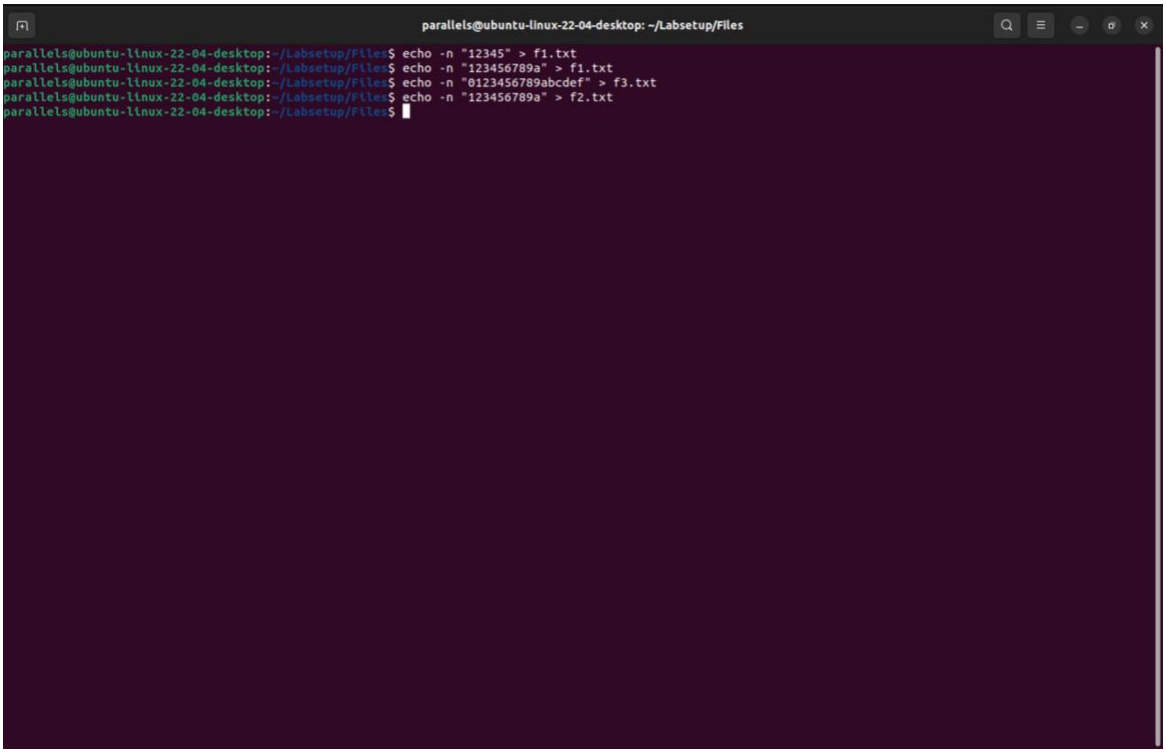
Screenshot41: (Return to text)



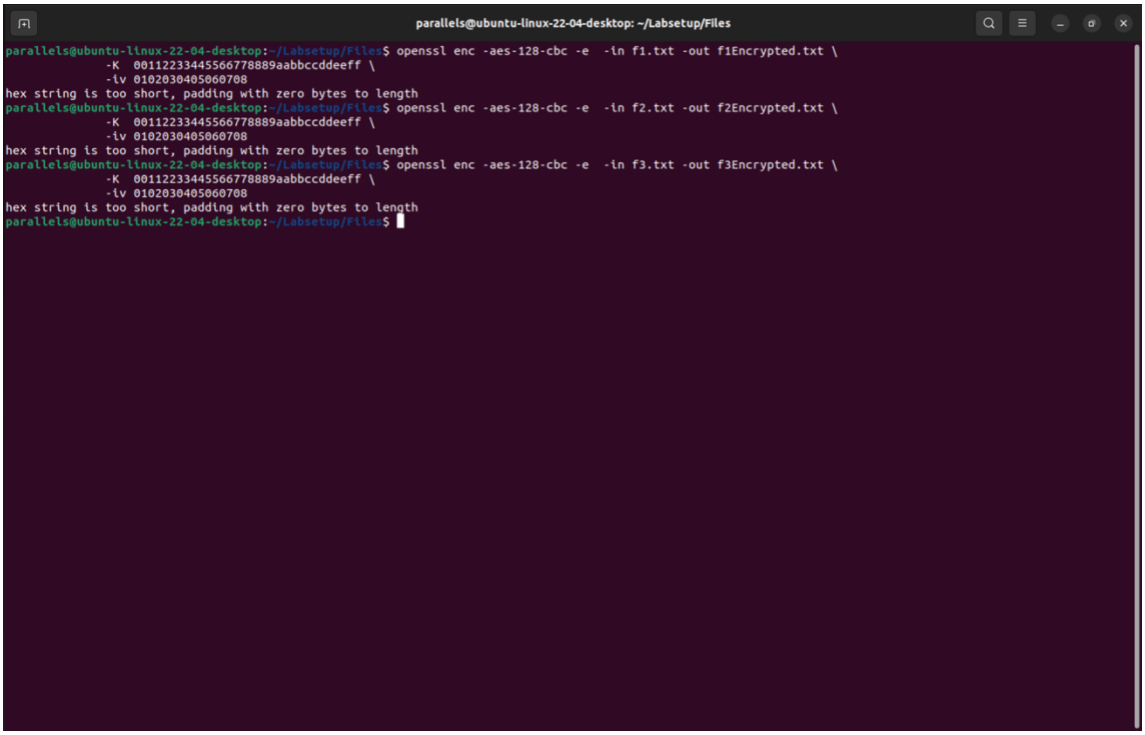
Screenshot42: (Return to text)



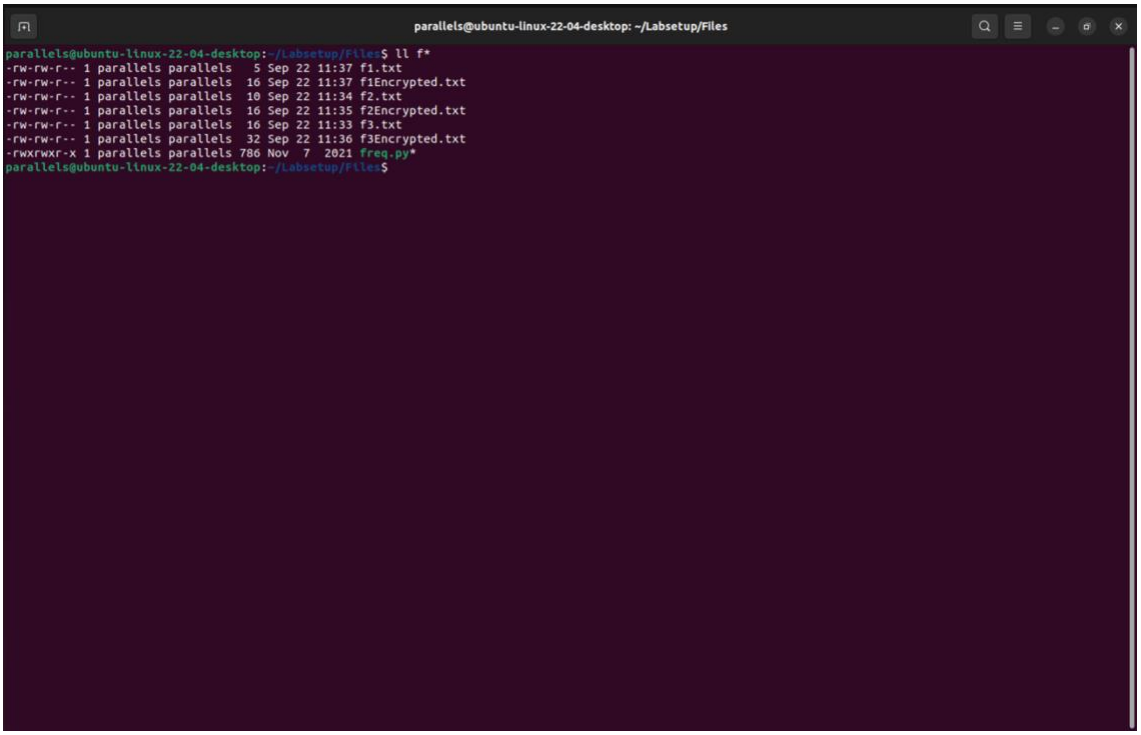
Screenshot43: (Return to text)



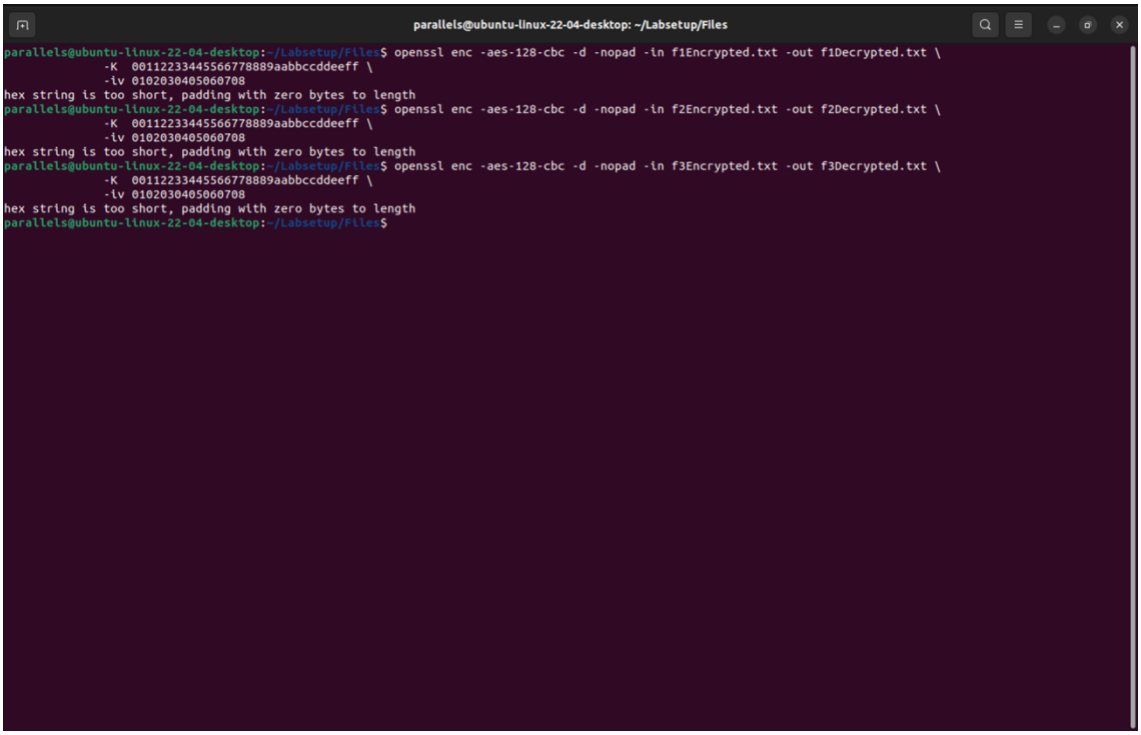
Screenshot44: (Return to text)



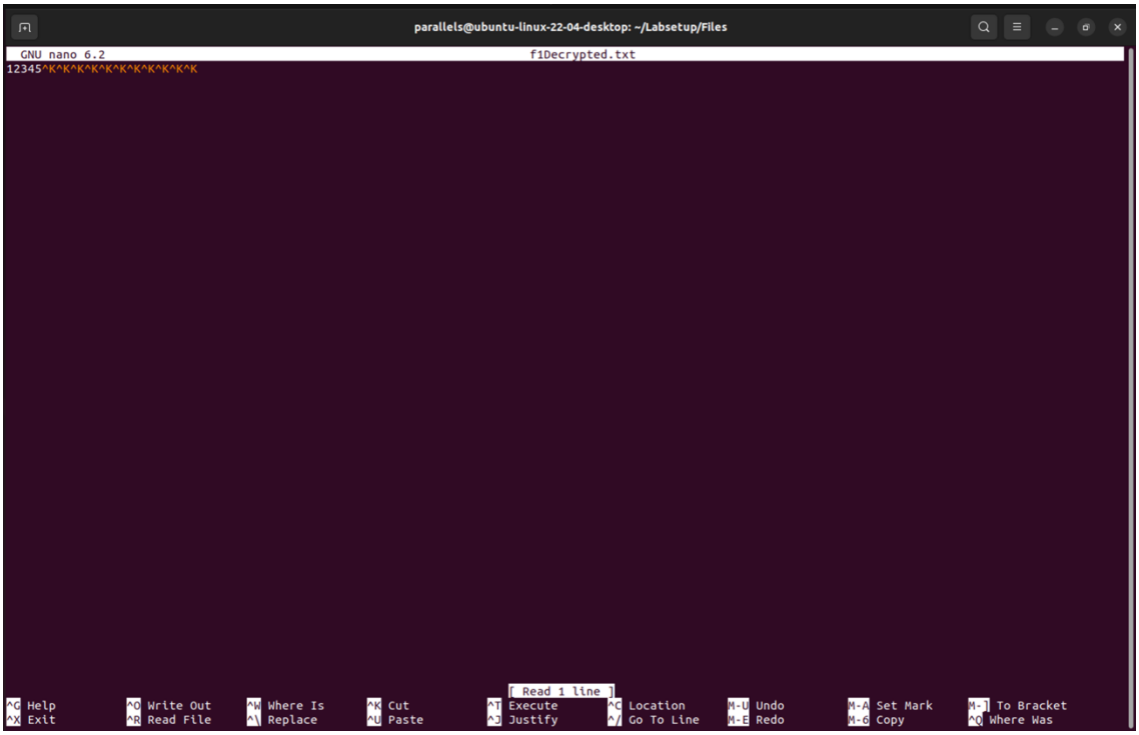
Screenshot45: (Return to text)



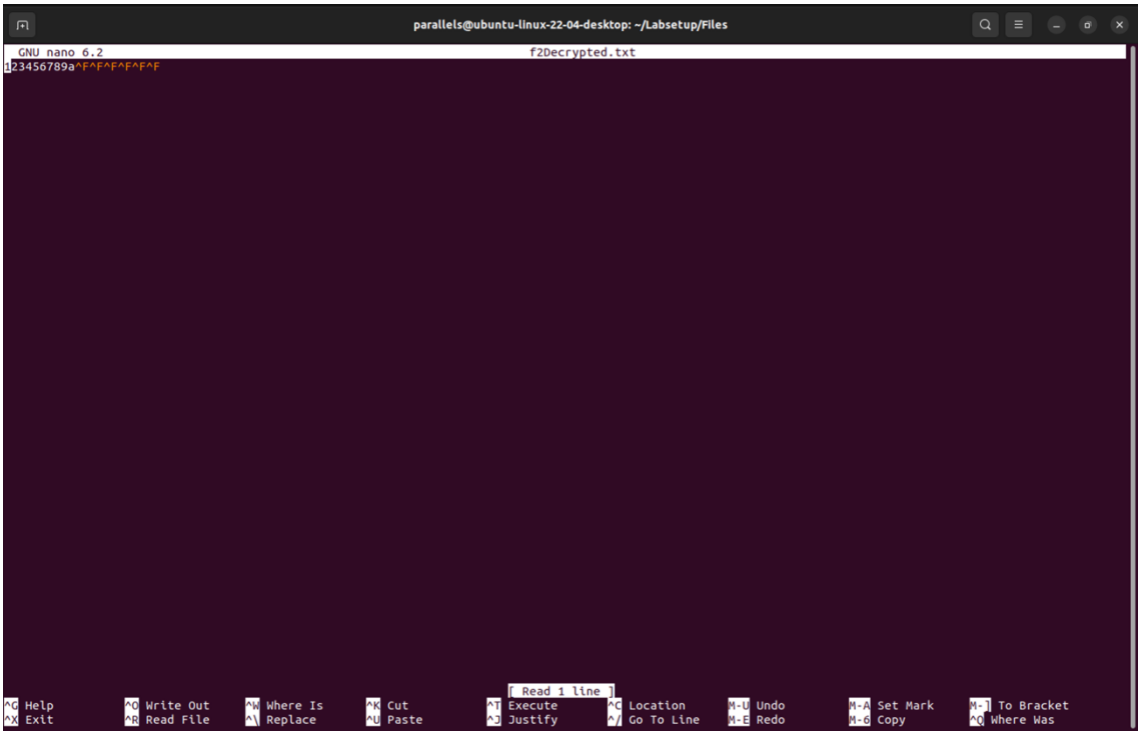
Screenshot46: (Return to text)



Screenshot47: (Return to text)

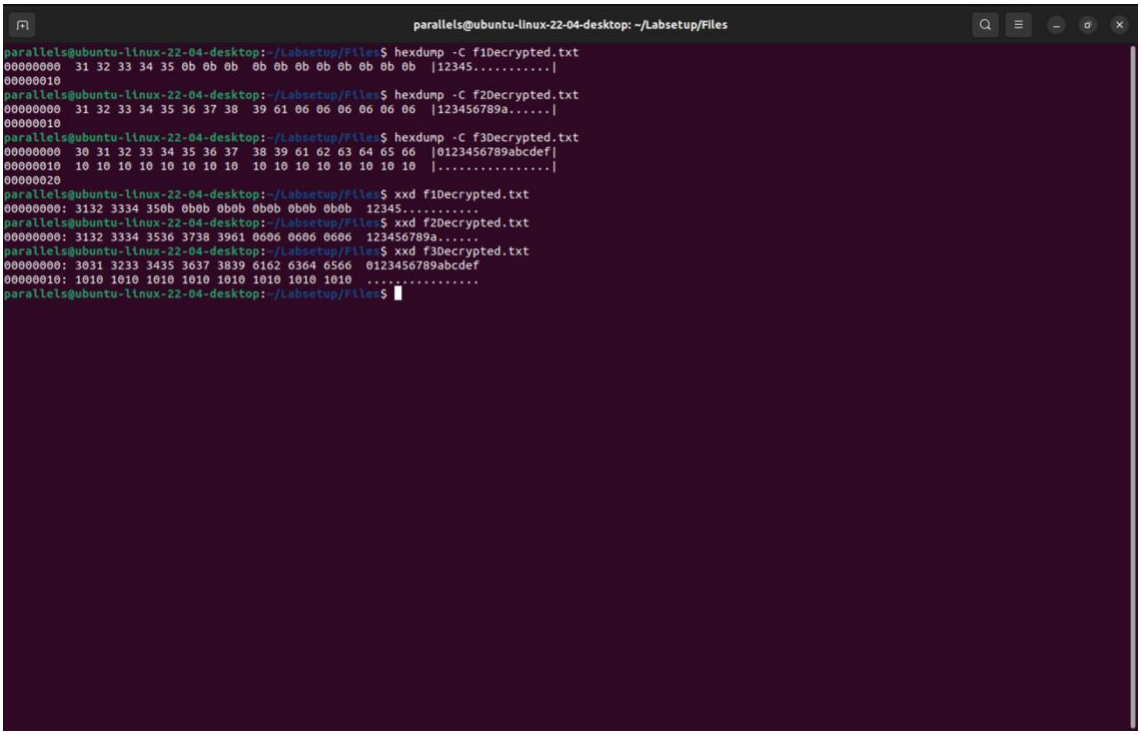


Screenshot48: (Return to text)



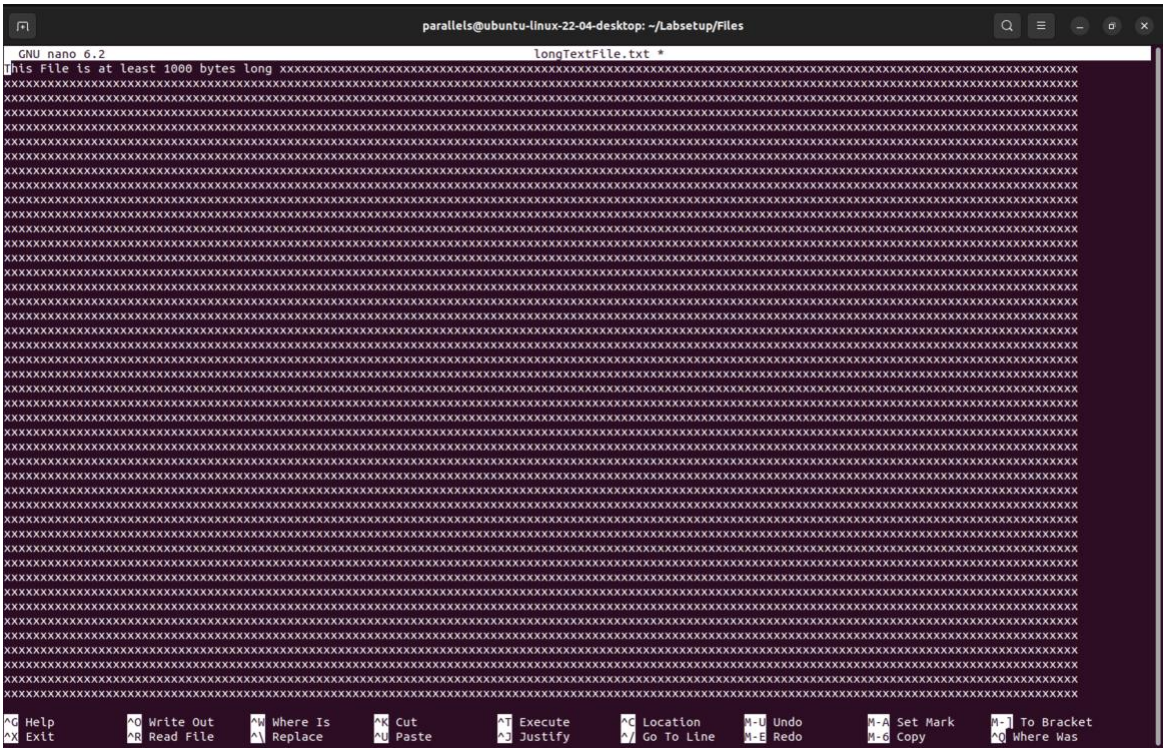
Screenshot49: (Return to text)

Screenshot50: (Return to text)

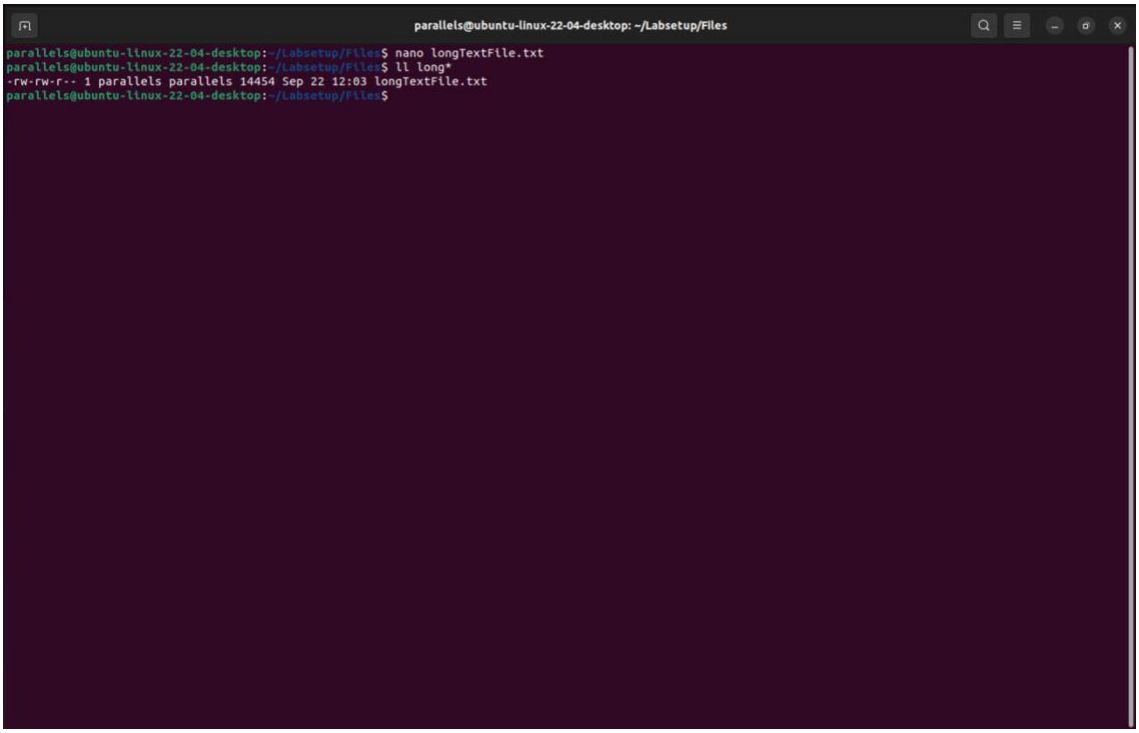


Screenshots: Task 5

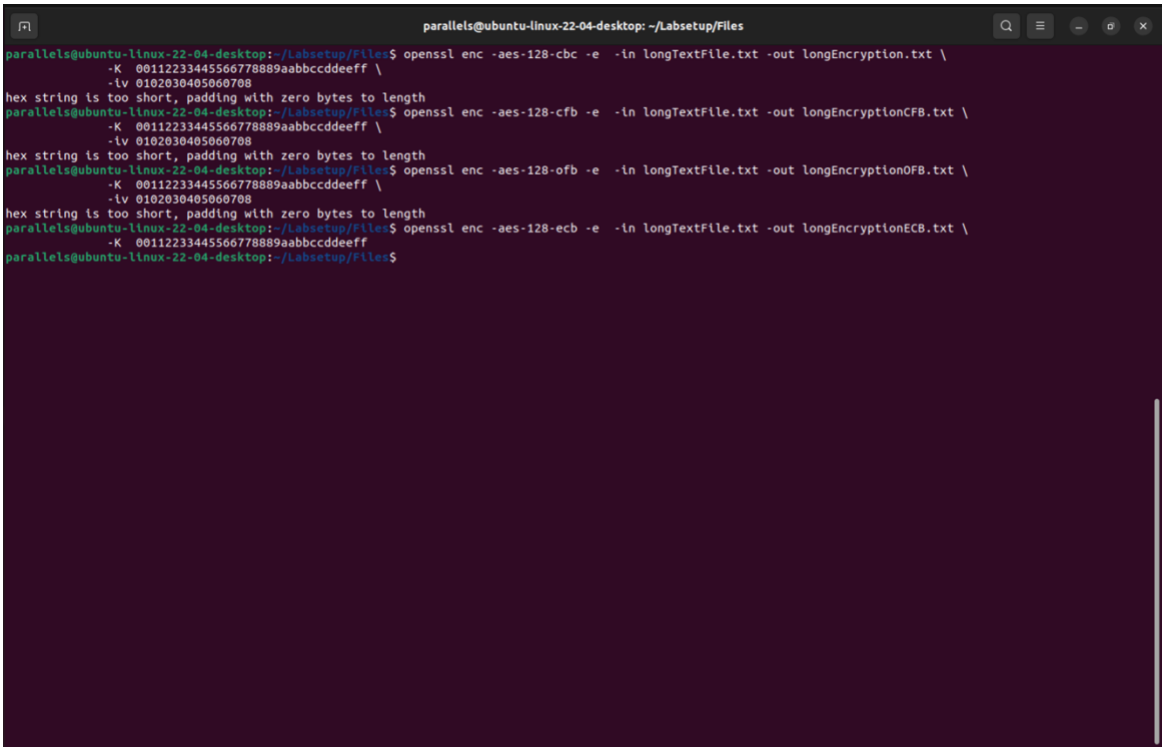
Screenshot51: (Return to text)



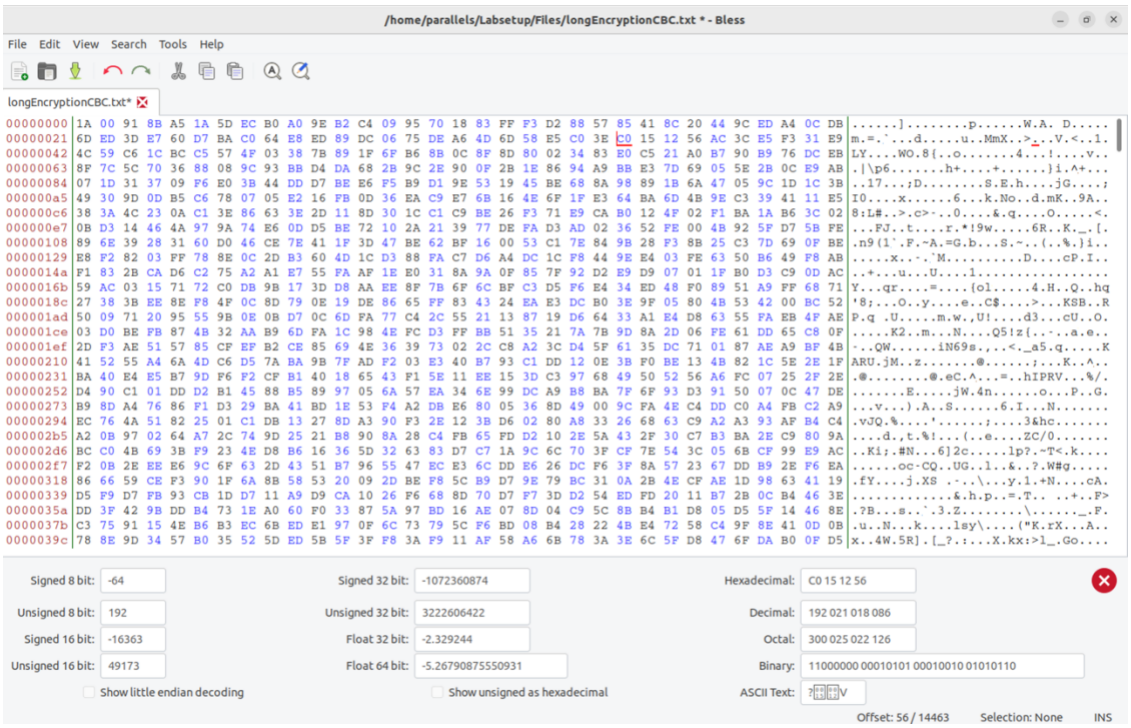
Screenshot52: (Return to text)



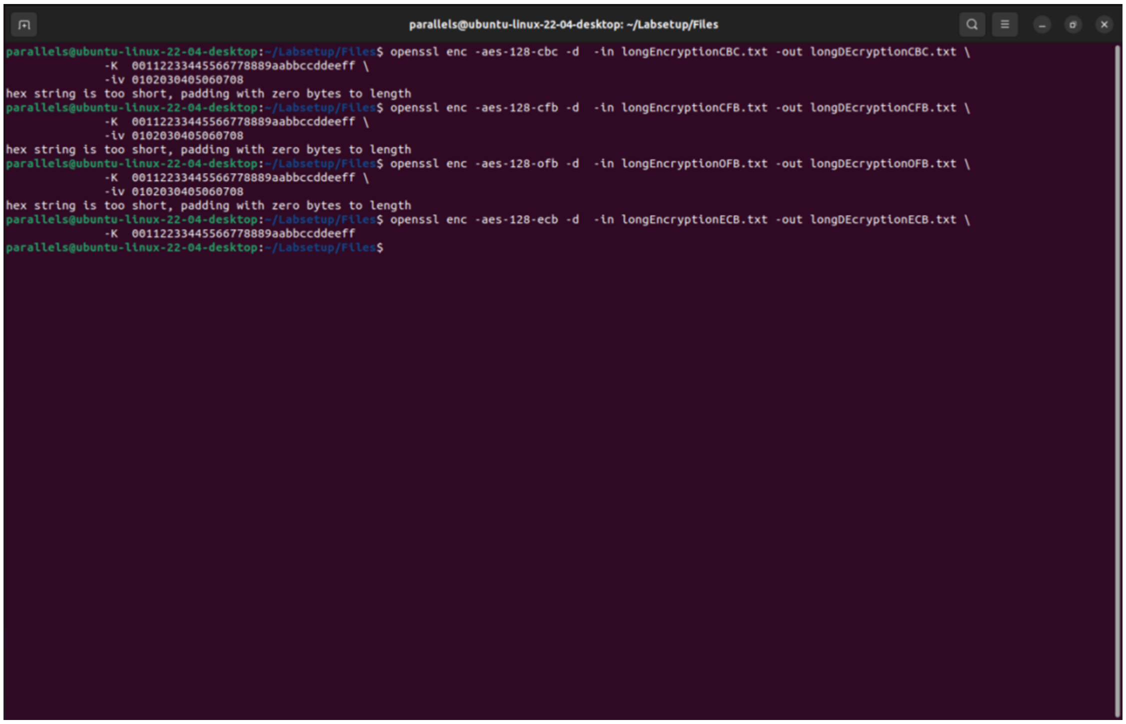
Screenshot53: (Return to text)



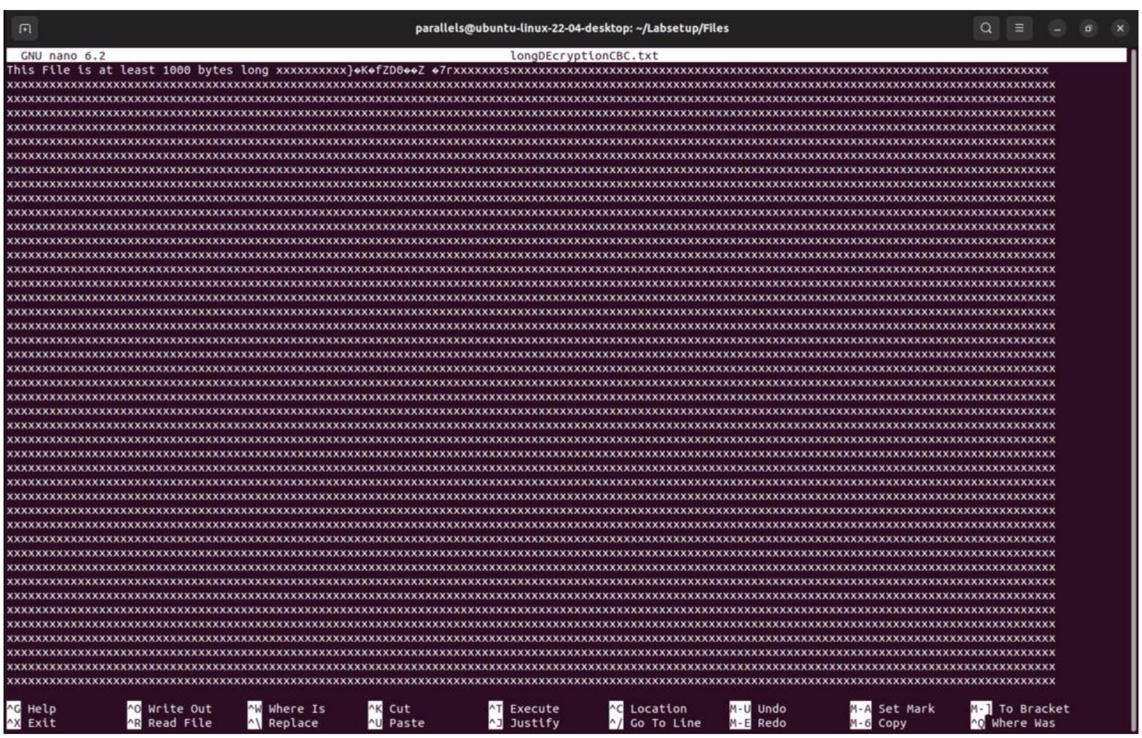
Screenshot54: (Return to text)



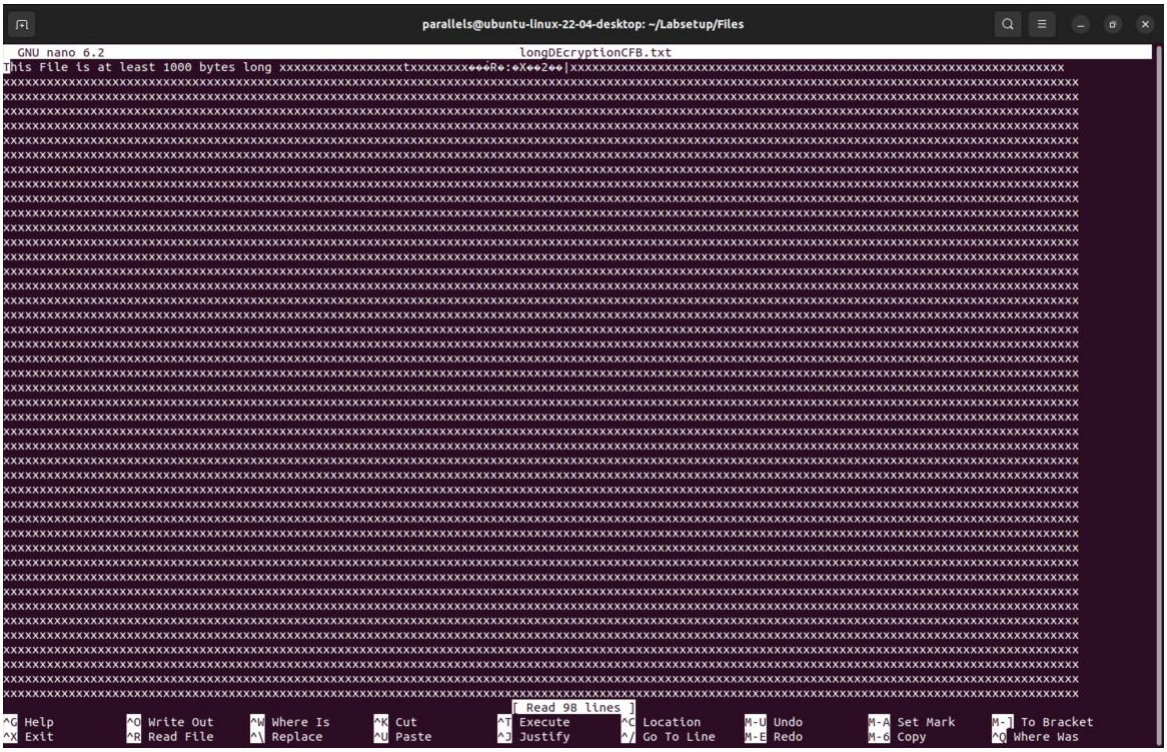
Screenshot55: (Return to text)



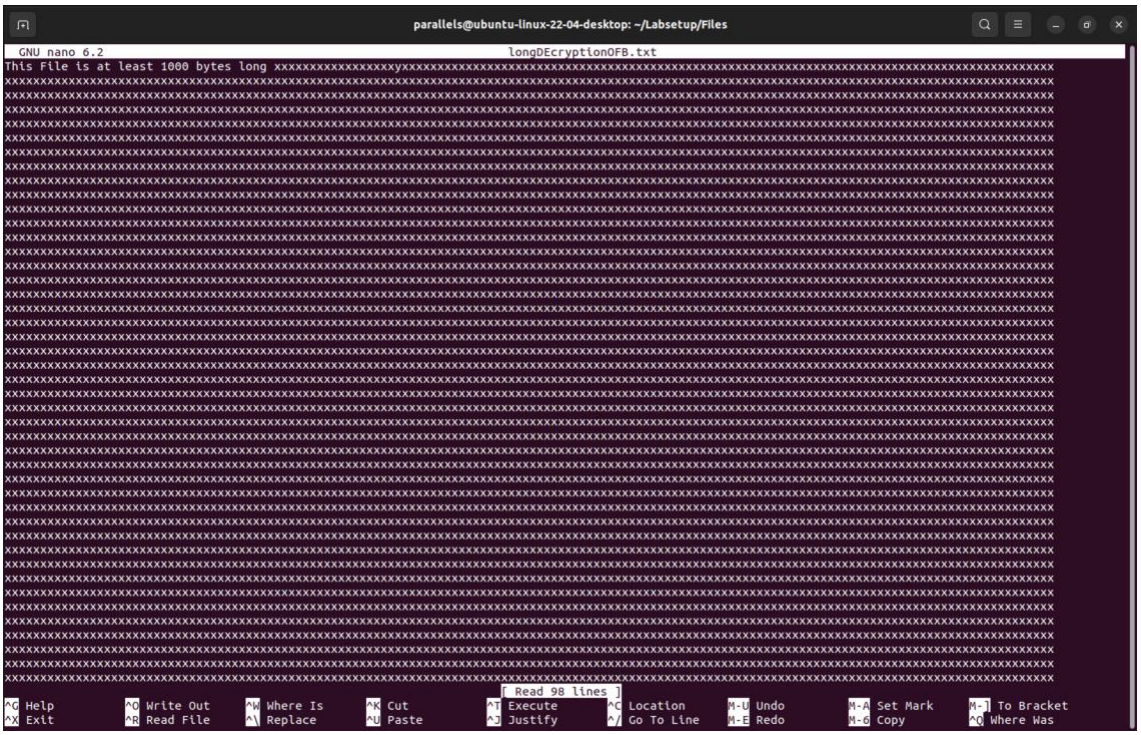
Screenshot56: (Return to text)



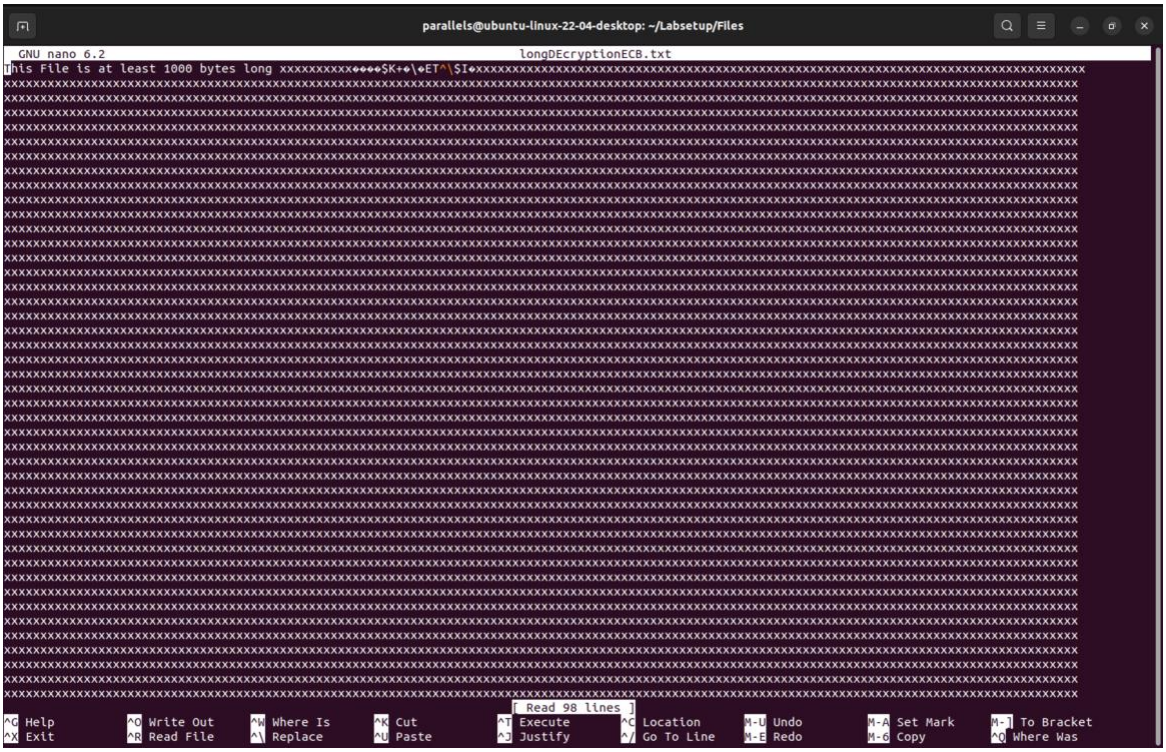
Screenshot57: (Return to text)



Screenshot58: (Return to text)

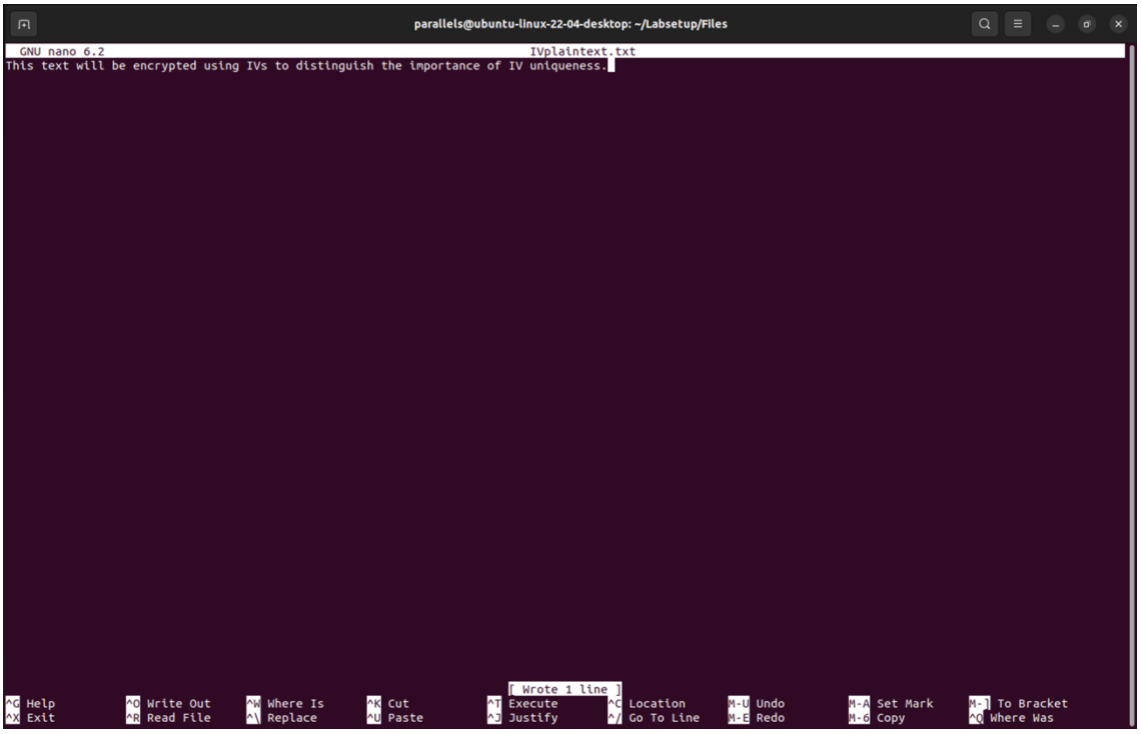


Screenshot59: (Return to text)

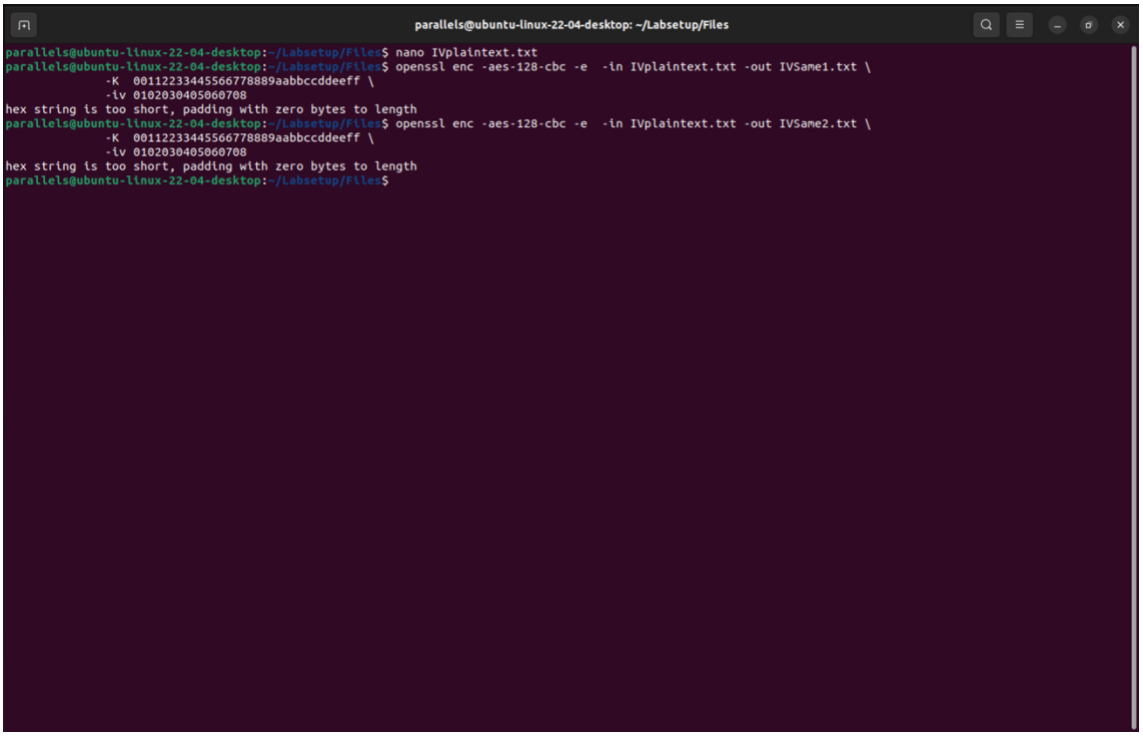


Screenshots: Task 6

Screenshot60: (Return to text)



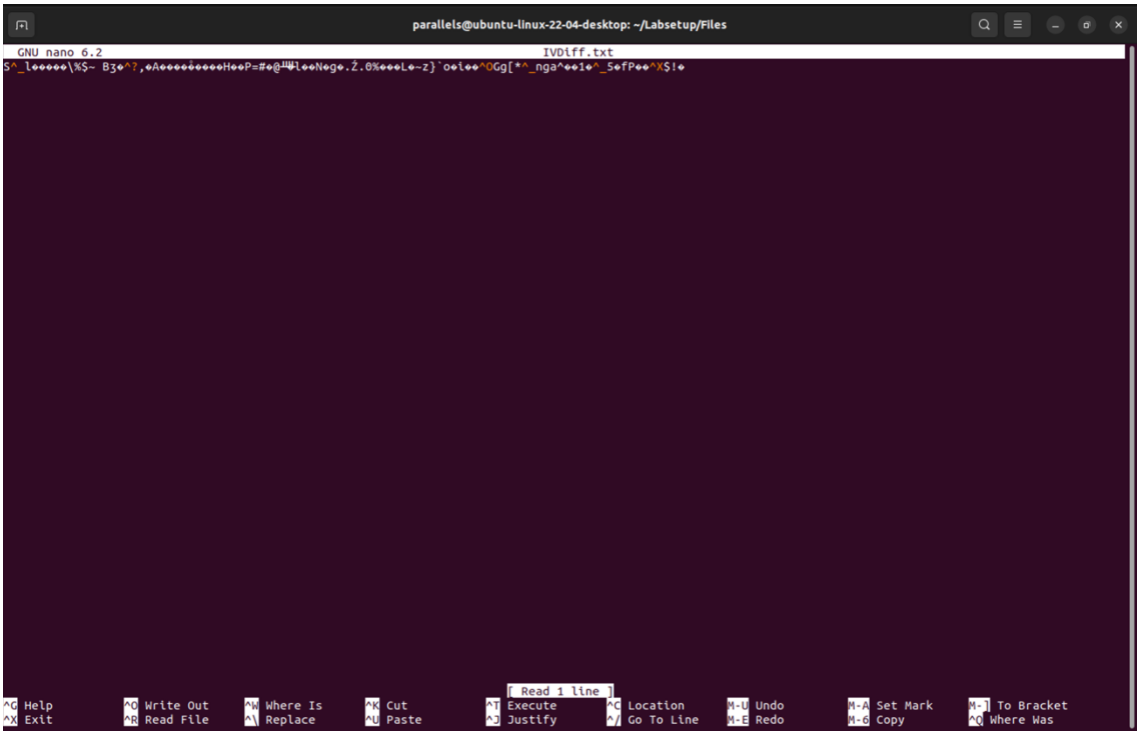
Screenshot61: (Return to text)



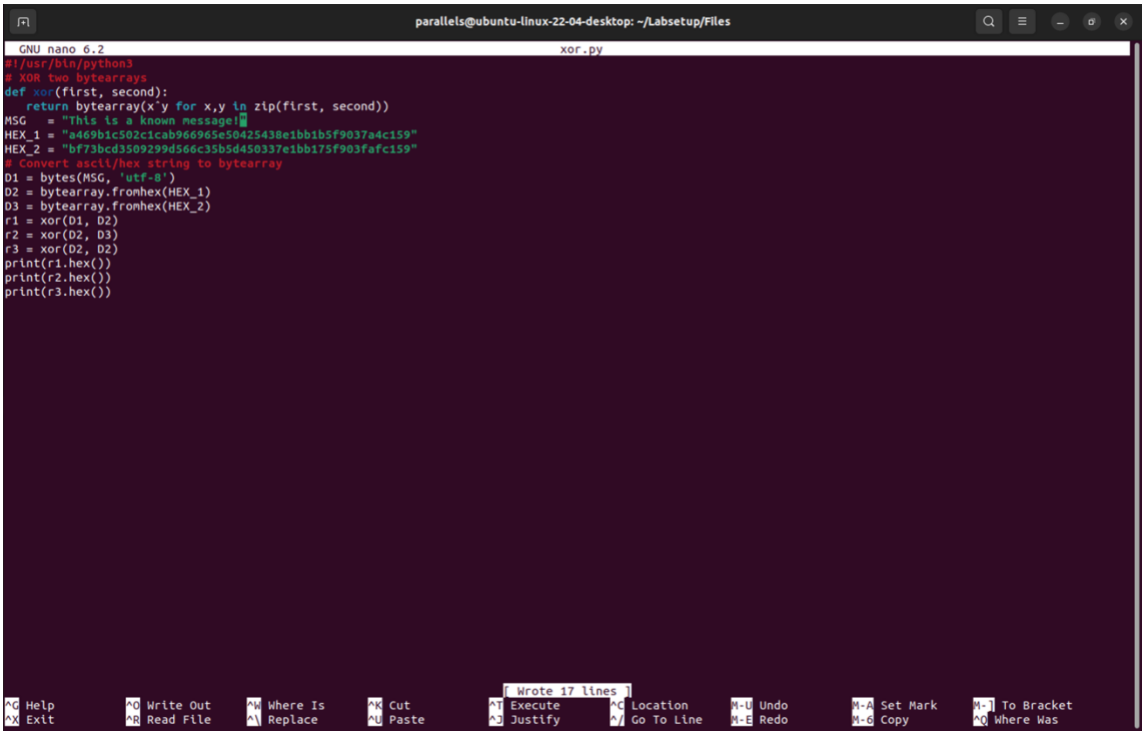
Screenshot62: (Return to text)

Screenshot63: (Return to text)

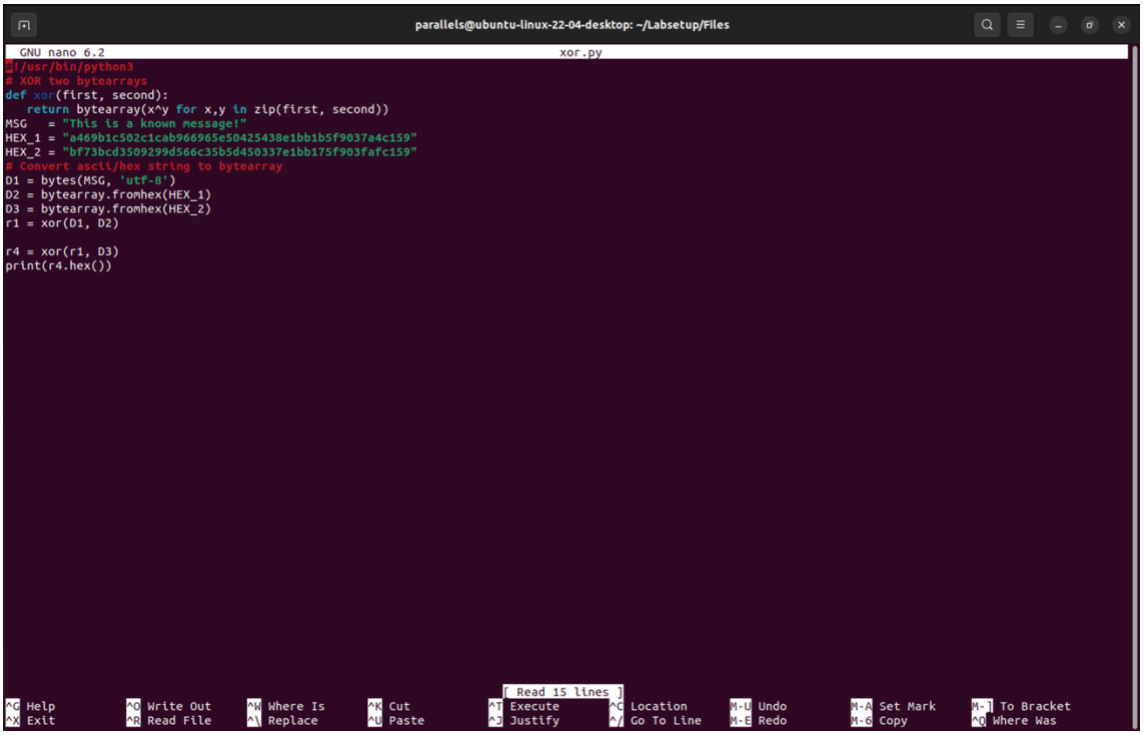
Screenshot64: (Return to text)



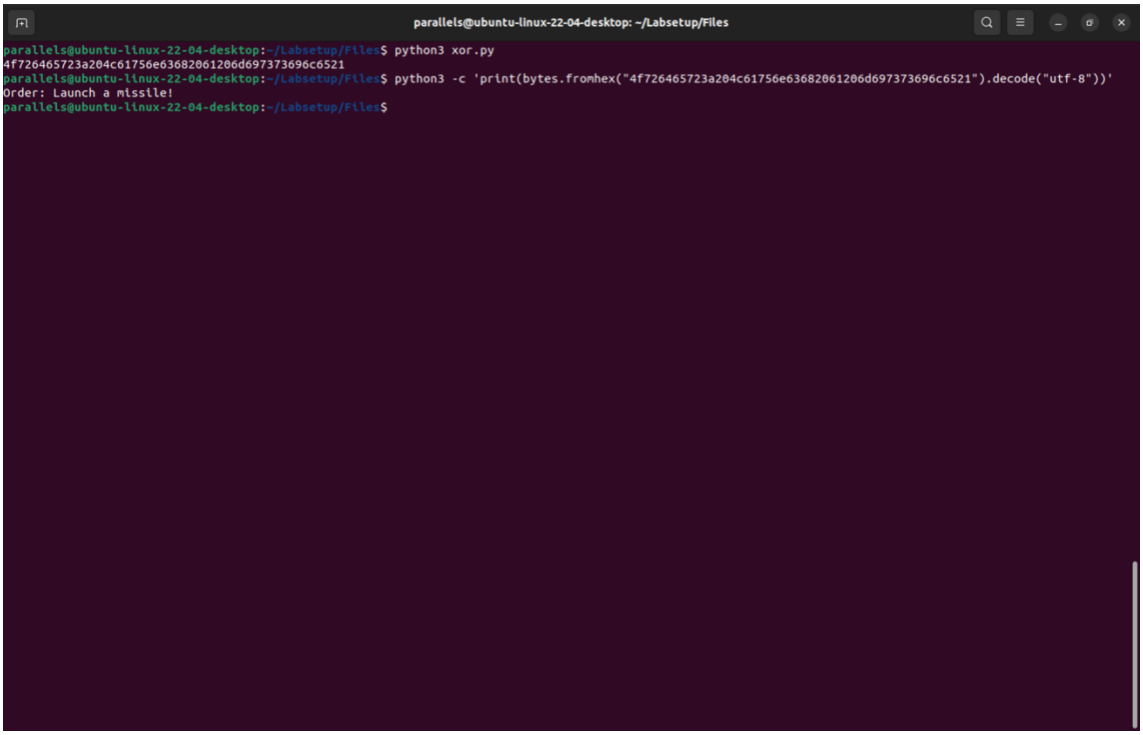
Screenshot65: (Return to text)



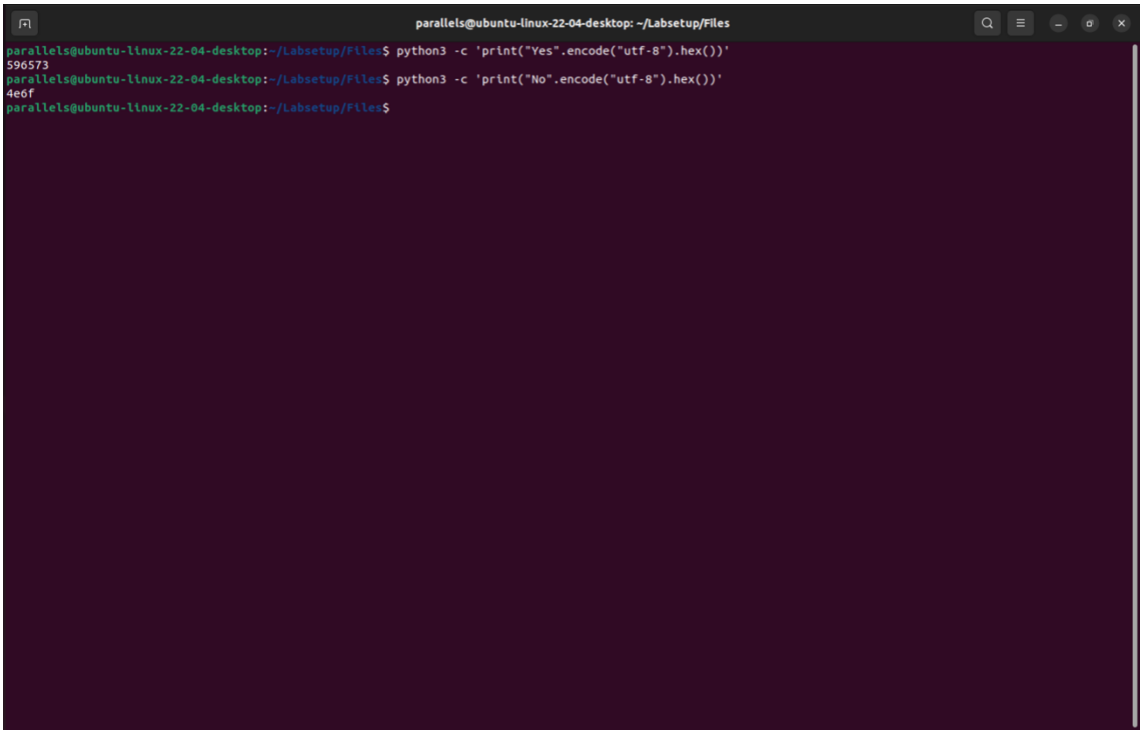
Screenshot66: (Return to text)



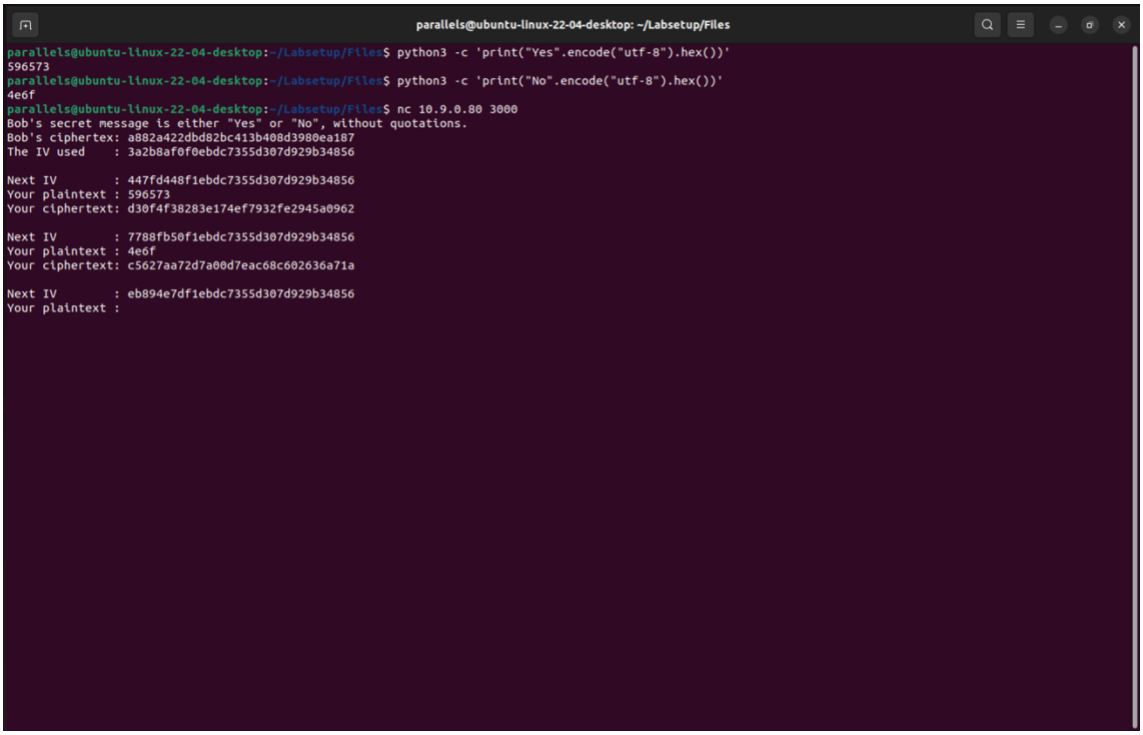
Screenshot67: (Return to text)



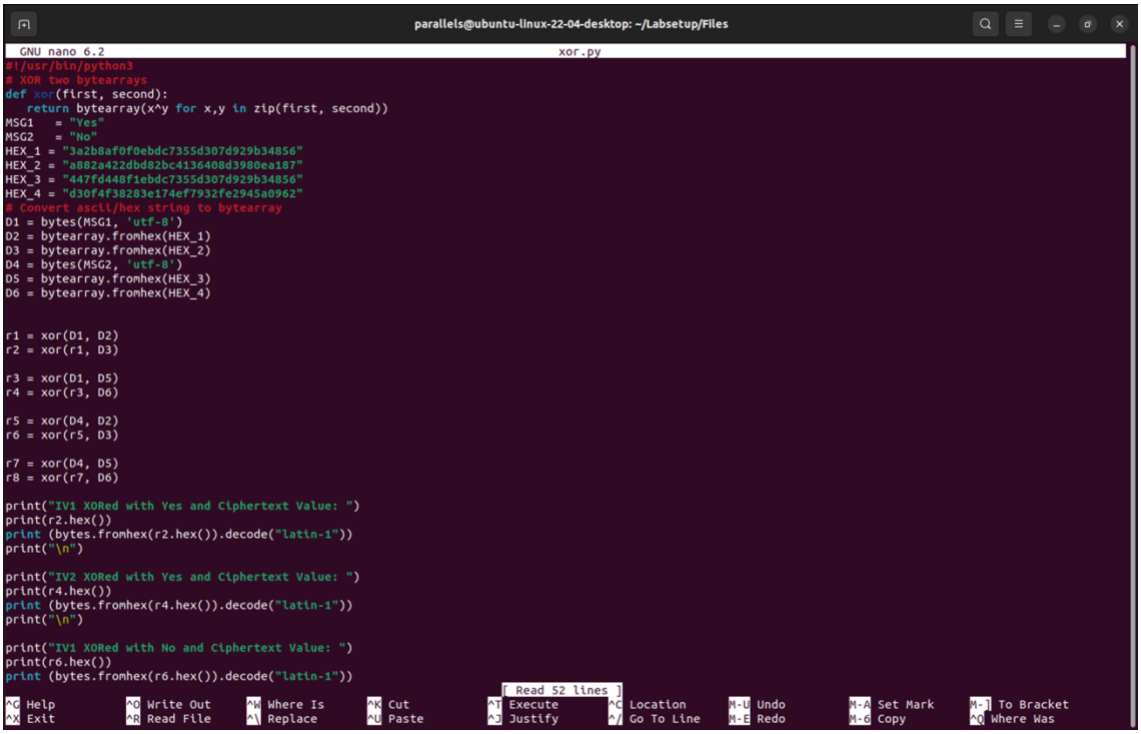
Screenshot68: (Return to text)



Screenshot69: (Return to text)



Screenshot70: (Return to text)



Screenshot71: (Return to text)