# 5CS031 Network Security

# Workshop 2 Cryptographic Tools

**Student Name:\_\_Loucas Xiourouppa\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Number:\_\_\_2007307\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**This is an assessed workshop. You will need to complete the workshop tasks, answer the questions and then submit the Word file, complete with your screenshots and answers, before the deadline posted on the workshops submission page, on Canvas.**

**Background**

Cryptographic tools are used to:

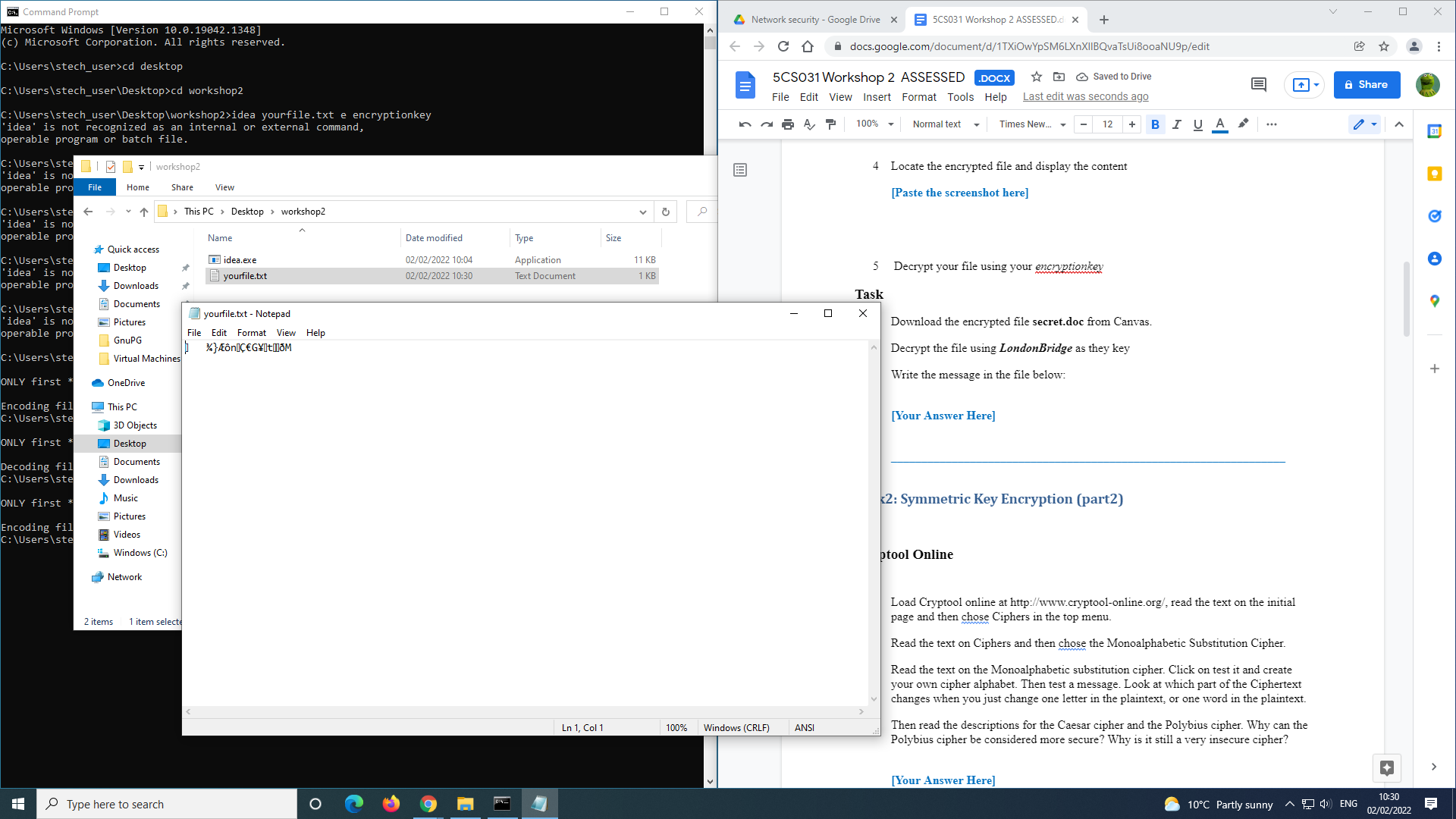
1. Securely transmit information between two parties to counteract passive attacks (where eavesdroppers extract private information.
2. Authenticate messages, thereby counteracting active attacks, where a third party attempts to impersonate a trusted friend to extract information known to be useful through conversation.

This workshop will investigate the methods discussed, beginning with simple techniques before progressing to more recent implementations.

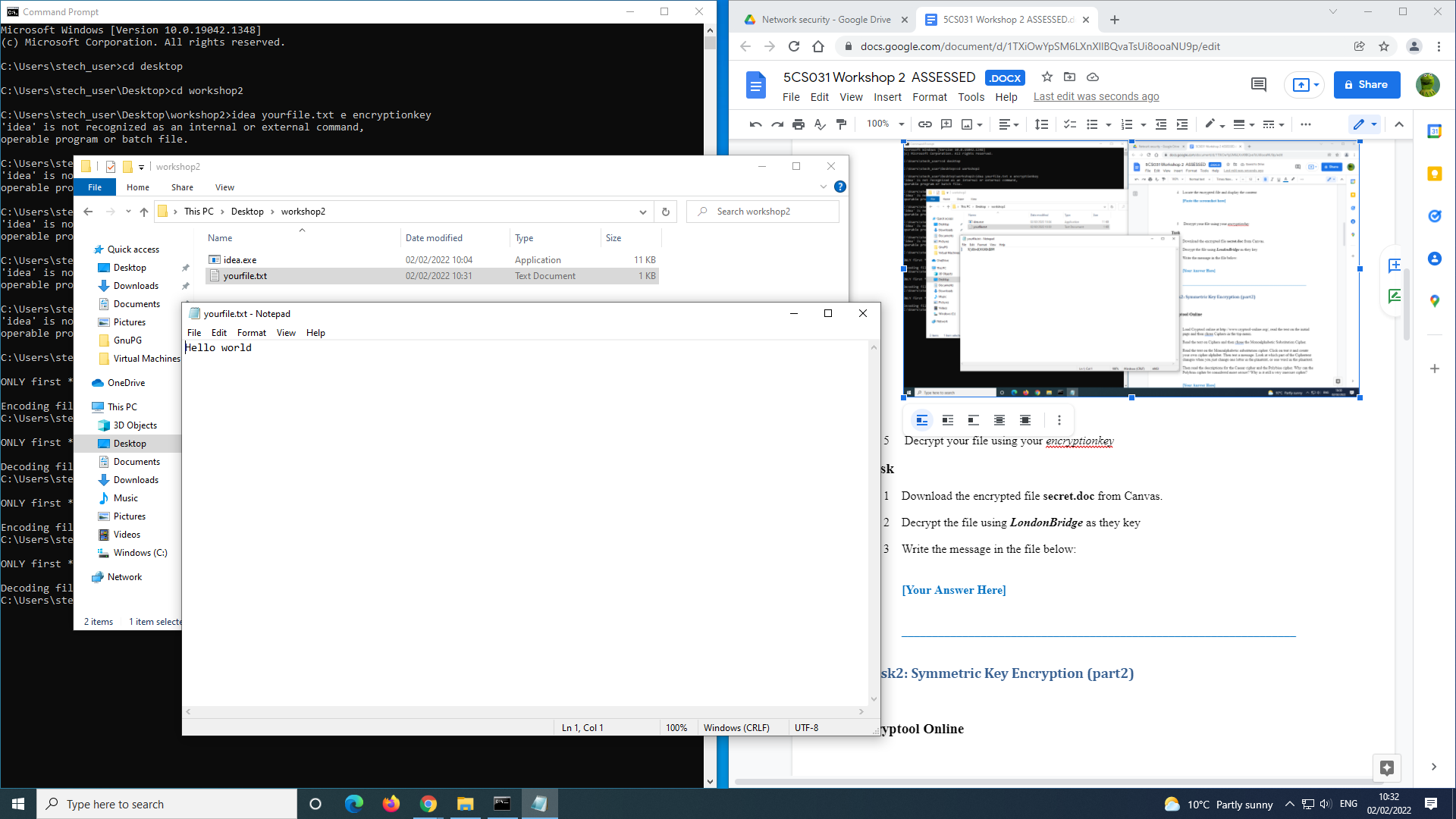
# Task1: Symmetric Key Encryption (part1)

1. Download **idea.exe** from Resources in Canvas, and save it to a directory of your choosing. Create a plain-text file in the same directory.
2. Open a Command Prompt and navigate to the directory containing idea.exe and your file.
3. There are two basic commands:  
     
   To encrypt: idea *yourfile.txt* e *encryptionkey*To decrypt: idea *yourfile.txt* d *encryptionkey*Make sure you remember what the key is, and don't use any spaces for the filenames or the keys.
4. Locate the encrypted file and display the content

**[Paste the screenshot here]**

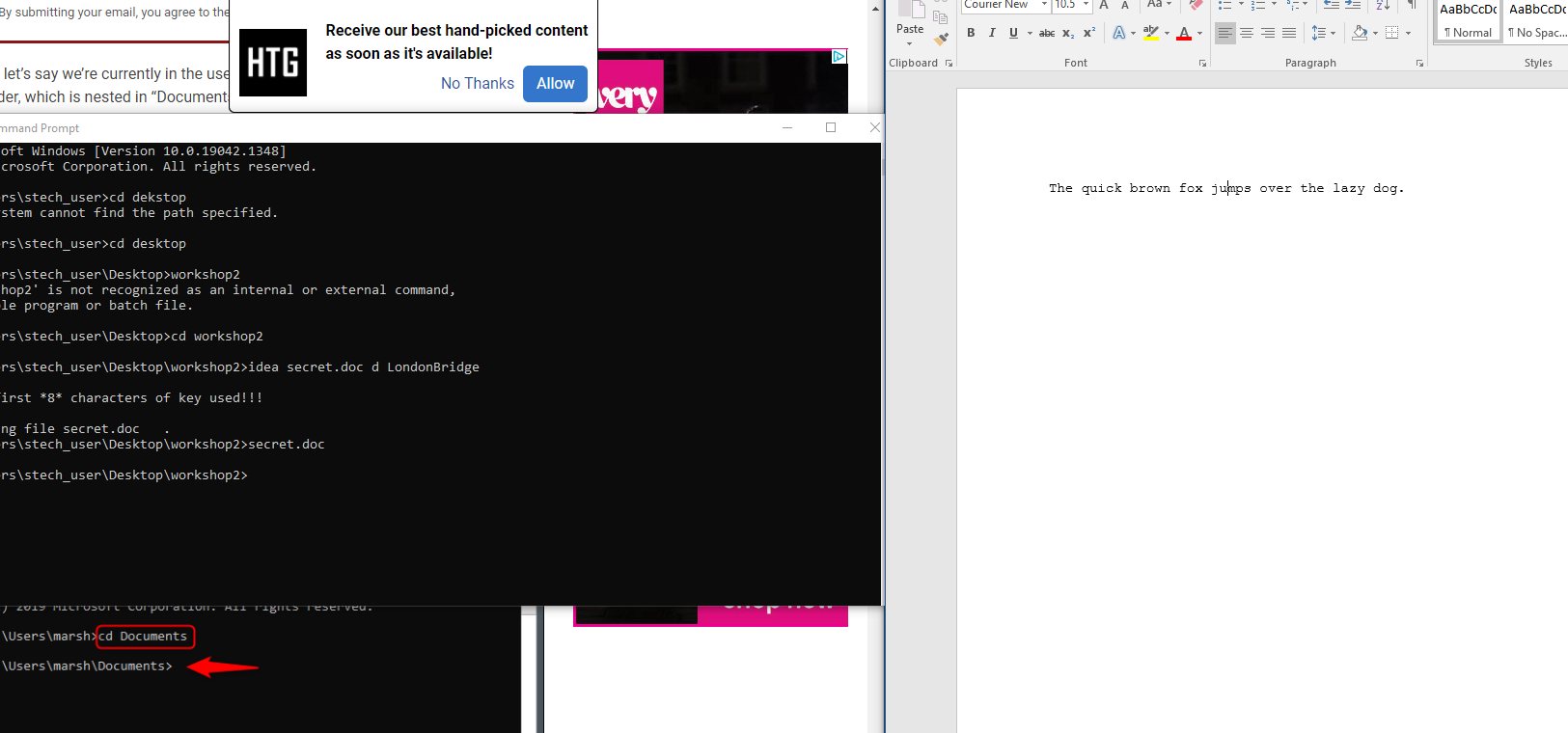
****

AFTER DECRYPTION



1. Decrypt your file using your *encryptionkey*

#### Task

1. Download the encrypted file **secret.doc** from Canvas.
2. Decrypt the file using ***LondonBridge*** as they key
3. Write the message in the file below:
4. 

**[Your Answer Here]**

**THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Task2: Symmetric Key Encryption (part2)

#### Cryptool Online

1. Load Cryptool online at http://www.cryptool-online.org/, read the text on the initial page and then chose Ciphers in the top menu.
2. Read the text on Ciphers and then chose the Monoalphabetic Substitution Cipher.
3. Read the text on the Monoalphabetic substitution cipher. Click on test it and create your own cipher alphabet. Then test a message. Look at which part of the Ciphertext changes when you just change one letter in the plaintext, or one word in the plaintext.
4. Then read the descriptions for the Caesar cipher and the Polybius cipher. Why can the Polybius cipher be considered more secure? Why is it still a very insecure cipher?

**[Your Answer Here]**

**The polybuis cipher uses a 5x5 grid to encrypt letters to numbers and still keep meaning to the sentence whereas the caesar cipher just replaces a letter with the letter after it in the alphabet which is really easy to decipher if you know how it works.**

1. The following ciphertext is derived from an English plaintext using a Monoalphabetic Substitution Cipher. It is not case sensitive, “,” and “.” are unencrypted, blanks are not deleted:

**xdat kyagsx udyiq qd gshzk$sm**

**zd xtt $j $z ugx bdxx$oit zd qd $z.**

**$j hdy byz g ig{mt xu$z&k $s xdat &gvt**

**xdatukt{t, u$zk g x$ms ds $z xgh$sm:**

**tsq-dj-zkt-ud{iq xu$z&k. bitgxt qd sdz zdy&k,**

**zkt bg$sz udyiq sdz tvts kgvt z$at zd q{h.**

1. Which approach would you chose to start an analysis of this ciphertext? What is the plaintext? Which key is used?

**[Your Answers Here: Find the plaintext]**

**The approach I took was to first google how the monoalphabetic substitution Cipher works. Then use an online editor to figure out the key used.**

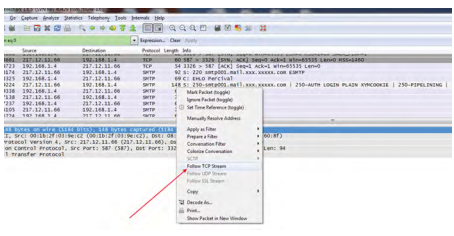
SOME HUMANS COULD DO ANYTHING TO SEE IF IT WAS POSSIBLE TO DO IT. IF YOU PUT A LARGE SWITCH IN SOME CAVE SOMEWHERE, WITH A SIGN ON IT SAYING: END-OF-THE-WORLD SWITCH. PLEASE DO NOT TOUCH, THE PAINT COULD NOT EVEN HAVE TIME TO DRY.

# Task3: Extract an Email Username and Password.

Some email services don’t use strong encryption. In this exercise, you will extract a username and password from an SMTP capture.

You will use the **smtp.pcap** file available from the resources on Canvas.

1. Download and install **Wireshark** if you don’t already have it on your computer
2. Start Wireshark and open the **smtp.pcap**.
3. Right-click any packet and choose Follow TCP Stream, as shown below



1. Examine the stream and look for the values that represent the username and password.
2. **Hint:** These values are shown as Base64-encoded string.

**[Write down the decoded username and password]**

**username - galunt**

**password - V1v1tr0n**

# Task4: Encrypting and Decrypting Data Using OpenSSL

OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library. In this lab, you will use OpenSSL to encrypt and decrypt text messages.

1. **Required Resources**

* Kali Linux: You can download Kali linx VM if you don’t have it already
  1. <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>
* Internet access

1. **Encrypting Messages with OpenSSL**

OpenSSL can be used as a standalone tool for encryption. While many encryption algorithms can be used, this lab focuses on AES. To use AES to encrypt a text file directly from the command line using OpenSSL, follow the steps below:

* + 1. **Encrypting a Text File**

1. Log into Kali linux VM.
2. Open a terminal window. And check if you have OpenSSL installed by using the command **openssl version –a.** If it’s not installed research how to install openssl.
3. Create a directory in your home directory and give it a name, example **5cs031labs**
4. In this directory create a file called **letter\_to\_grandma.txt** and type some content in it.
5. Display the content of the file by using the following command

$ **cat letter\_to\_grandma.txt**

**[Paste the screenshot here]**

### 

1. From the same terminal window, issue the command below to encrypt the text file. The command will use AES-256 to encrypt the text file and save the encrypted version as **message.enc**. OpenSSL will ask for a password and for password confirmation. Provide the password as requested and be sure to remember the password.

$ **openssl aes-256-cbc -in letter\_to\_grandma.txt -out message.enc**

enter aes-256-cbc encryption password:

Verifying - enter aes-256-cbc encryption password:

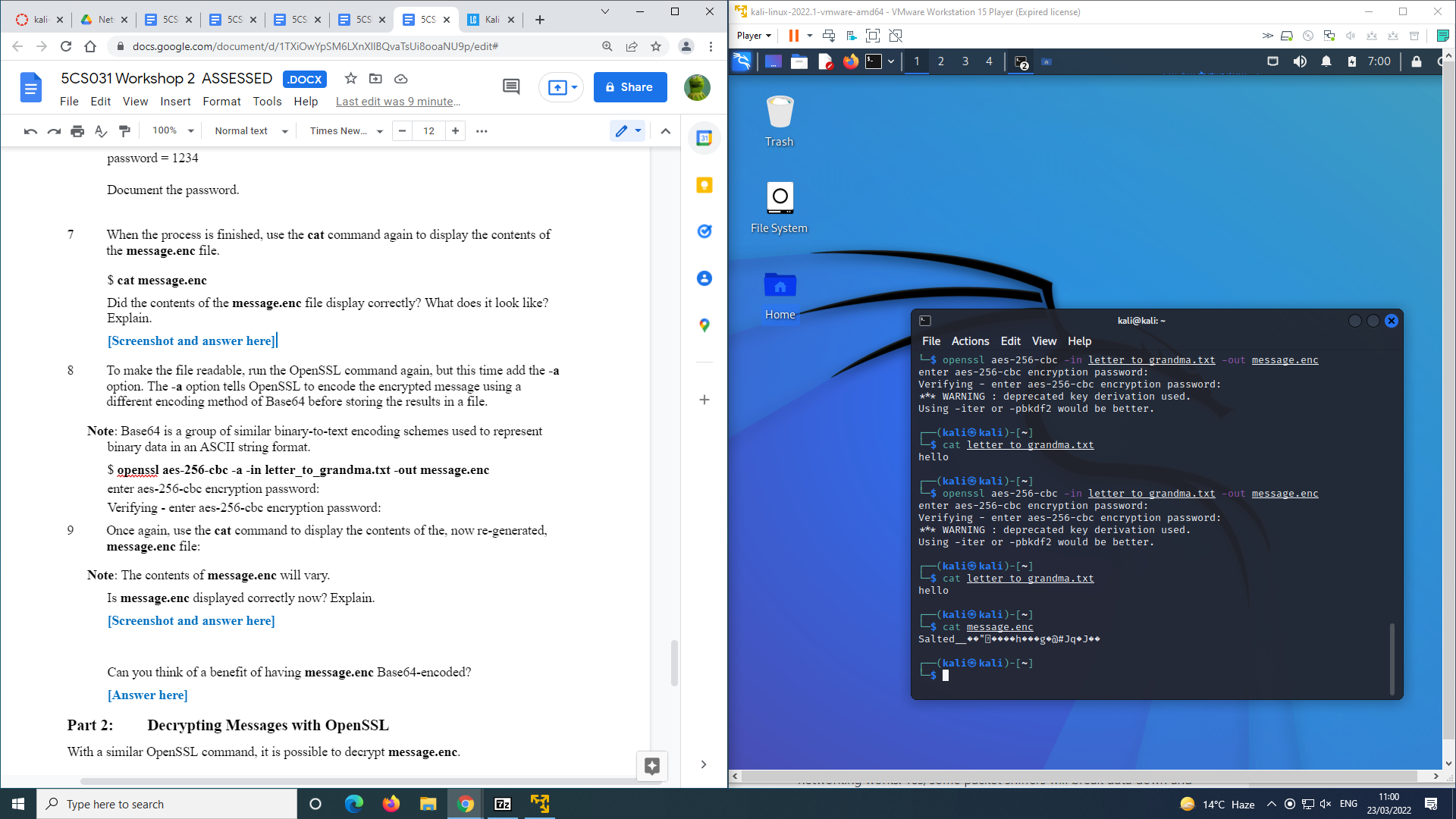
password = 1234  
  
Document the password.

1. When the process is finished, use the **cat** command again to display the contents of the **message.enc** file.

$ **cat message.enc**

Did the contents of the **message.enc** file display correctly? What does it look like? Explain.

**the texts in it look encrypted**

****

1. To make the file readable, run the OpenSSL command again, but this time add the **-a** option. The **-a** option tells OpenSSL to encode the encrypted message using a different encoding method of Base64 before storing the results in a file.

**Note**: Base64 is a group of similar binary-to-text encoding schemes used to represent binary data in an ASCII string format.

$ **openssl aes-256-cbc -a -in letter\_to\_grandma.txt -out message.enc**

enter aes-256-cbc encryption password:

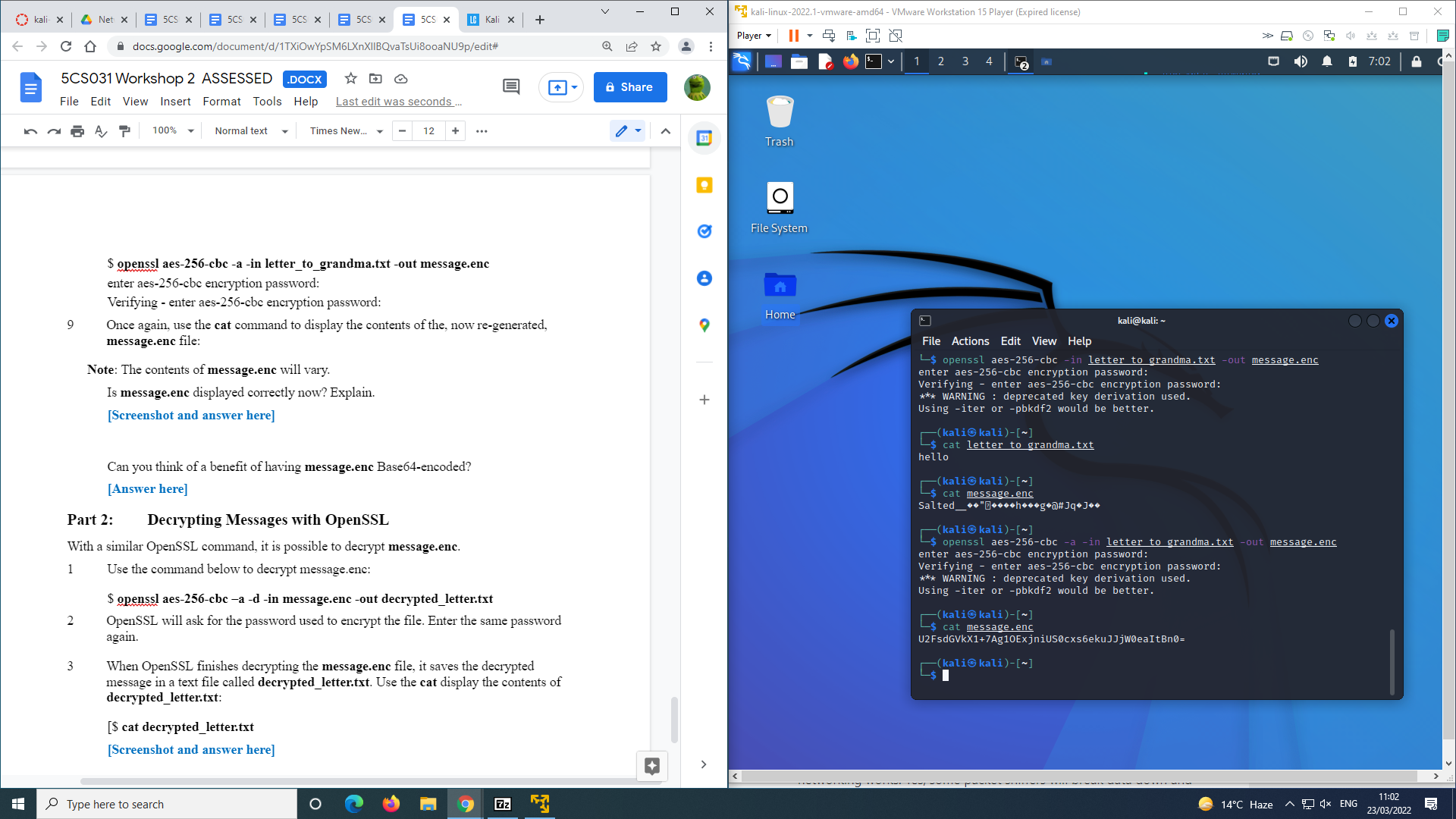
Verifying - enter aes-256-cbc encryption password:

1. Once again, use the **cat** command to display the contents of the, now re-generated, **message.enc** file:

**Note**: The contents of **message.enc** will vary.

Is **message.enc** displayed correctly now? Explain.

The text isn't encrypted anymore but the text has jumbled around.

****

Can you think of a benefit of having **message.enc** Base64-encoded?

**For someone who might not know base 64 encryption, this will stop them from getting sensitive data while the information can still have meaning with a base 64 decoder.**

1. **Decrypting Messages with OpenSSL**

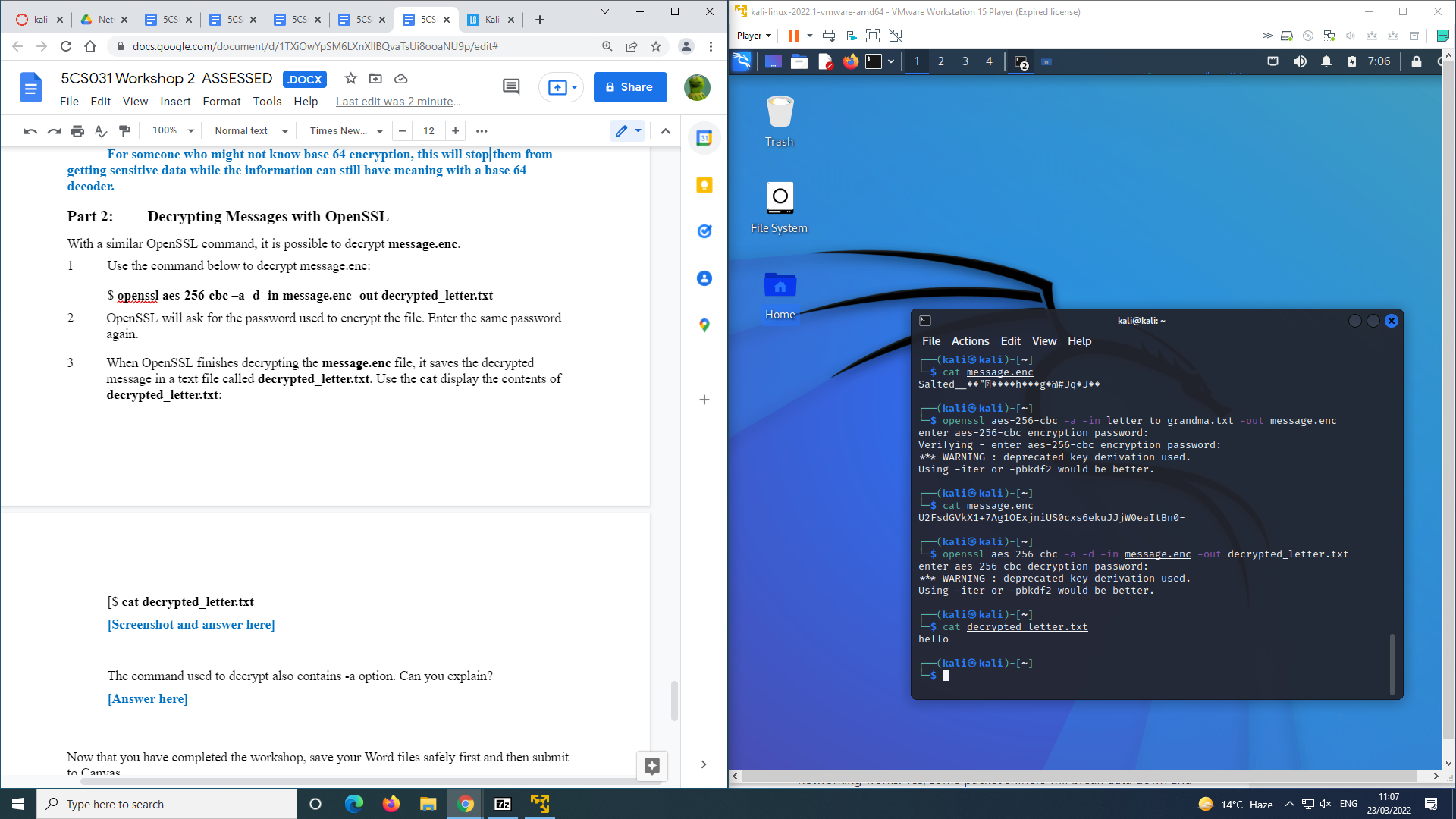
With a similar OpenSSL command, it is possible to decrypt **message.enc**.

1. Use the command below to decrypt message.enc:

$ **openssl aes-256-cbc –a -d -in message.enc -out decrypted\_letter.txt**

1. OpenSSL will ask for the password used to encrypt the file. Enter the same password again.
2. When OpenSSL finishes decrypting the **message.enc** file, it saves the decrypted message in a text file called **decrypted\_letter.txt**. Use the **cat** display the contents of **decrypted\_letter.txt**:

[$ **cat decrypted\_letter.txt**

****

The command used to decrypt also contains -a option. Can you explain?

**[Answer here]**

**The a- tells openssl to encode it using base 64 encryption. We also use it to decrypt the file using base 64 because its encrypted that way.**

# Now that you have completed the workshop, save your Word files safely first and then submit to Canvas.