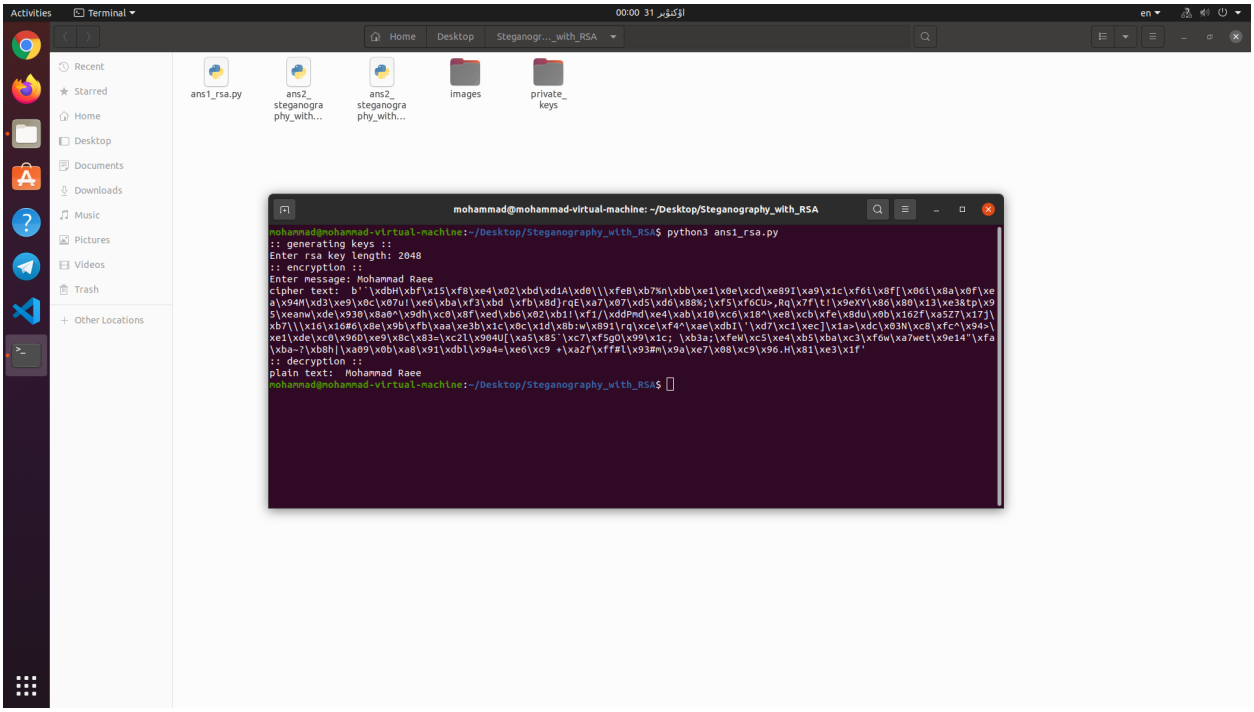
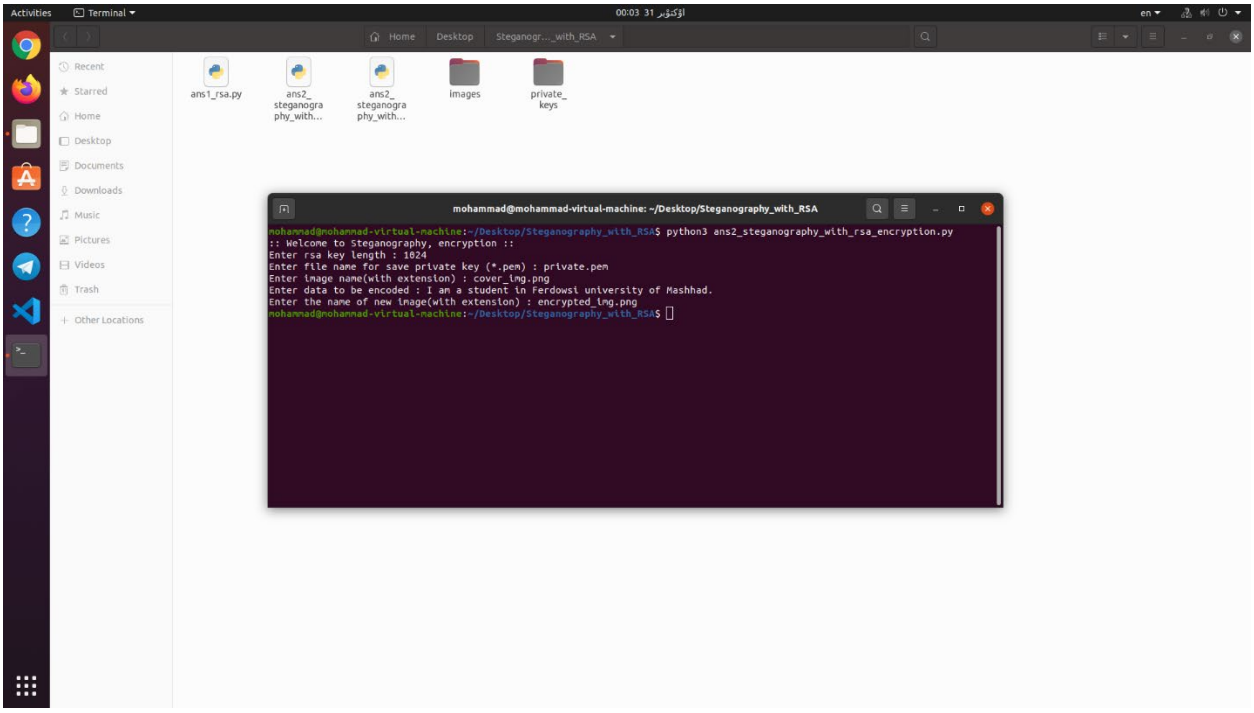
Part 1 : The answer to this question was placed in the **1_rsa.py** file. In this file we have 3 functions. The first function is generate_key, which takes the length of the rsa key and creates two public and private keys. The second function is encryption, which takes the public key and receives a text from the user to encrypt and display cipher_text in the output. And the third function is decryption, which displays the original text or plain_text by receiving the private key and encrypted text.

We see output 1_rsa.py in the image below:



Part2 : This section was placed on **2_steganography_with_rsa_encryption.py** Comments are complete. In this file, the encoding function is first executed, in which an integer is obtained from the user to generate the RSA key. It then takes the name of the file in which the private key wants to be stored from the user (this file must have a .pem extension). It then takes the input text from the user and gives it to the encryption function along with the public key. In this function, first, the text is encrypted with the public key and then the hash256 function is executed on it and then the hashed value is encrypted again with the public key and this value is attached to cipher_text and given as output. The user then gets the name of the image in which the watermark is to be placed (this image is placed in the images folder and this name is given to the encode_enc function along with the encryption output to replace the right bits of the pixels in the image. It then takes the URL of the image you want the cached image to be stored in from the user (this image is also stored in the images folder.

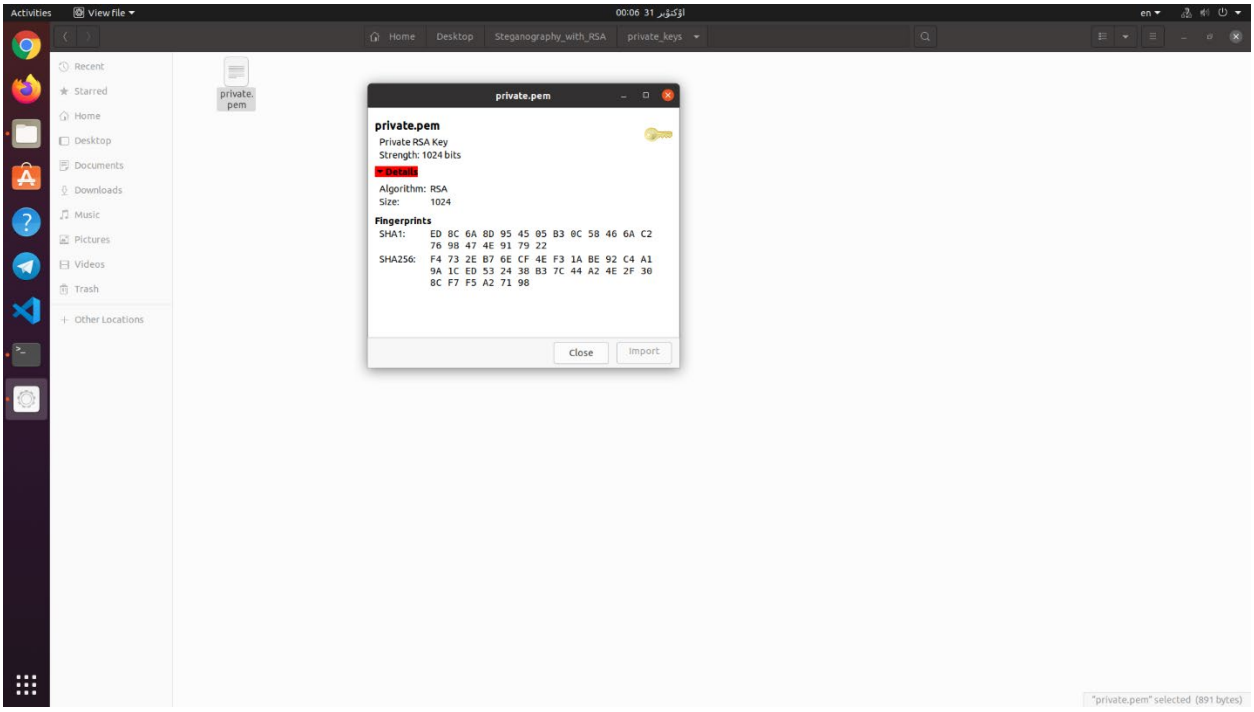
The output of the command line commands 2_steganography_with_rsa_encryption.py can be seen in the following image:



You can see the original image called cover_img.png in the image below:

You can see the encrypted image saved as encrypted image.png in the following image:

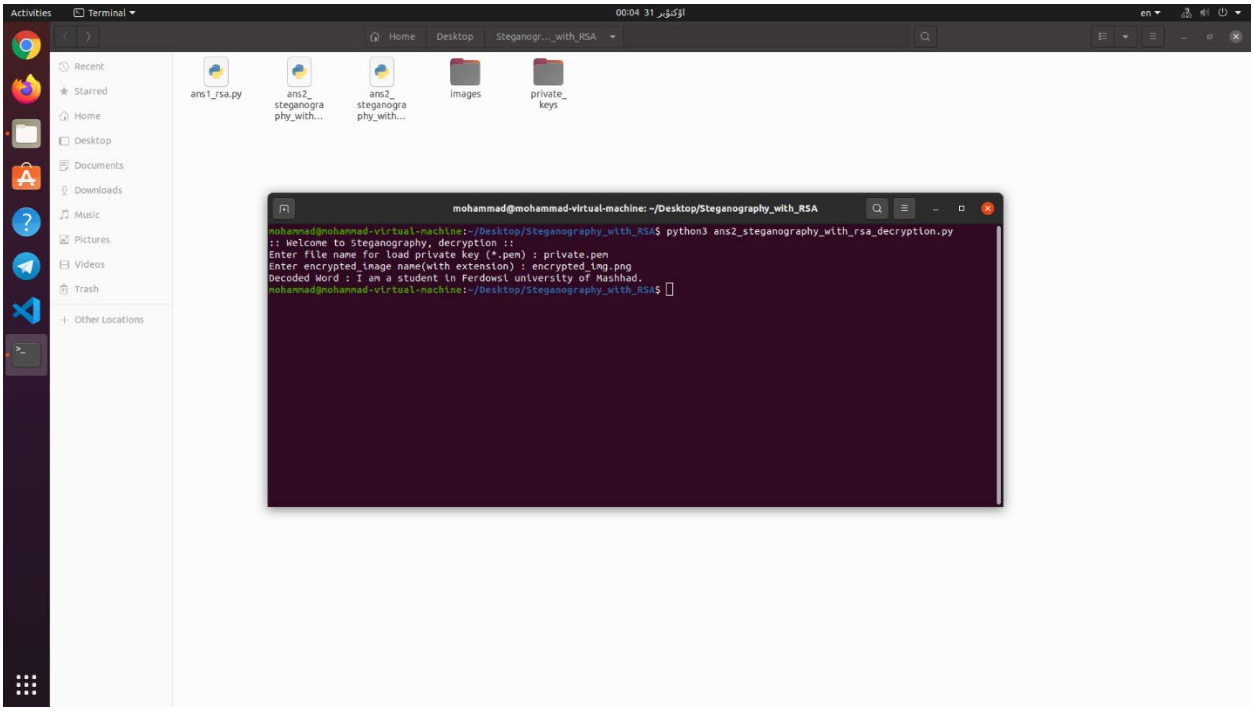
You can also see the image of the file containing the private key saved as private.pem in the image below:



Part 3 : This section was placed in the file **3_steganography_with_rsa_decryption.py**. In this file, first, it asks the user for the name of the image from which the password should be extracted (this image is in the images folder) then it asks the user for the name of the file from

which the private key should be extracted for decryption (this file is in the folder Private_keys). Now it calls the decryption function. This function first extracts the value coded in the image, which is the result of concat two cipher_text values and the cipher_text encrypted value. Well, it separates these two values and first decrypts the second part with private_key and then encrypts the first part with hash256 and compares the two values, if they are one then integrity is confirmed and then cipher_text with private Decrypts and displays, and in case of non-compliance, sends the message of non-confirmation of integrity.

The output of the command line 3_steganography_with_rsa_decryption.py commands can be seen in the following image:



Actual applications of cryptography : In response to **two examples** of cryptographic applications. **The first** use is in the world of economics, cryptocurrency. Currency encryption becomes one of the most important applications of blockchain and uses public and private keys to protect the address of blockchain users. In the case of blockchain encryption, private keys are used as the individual address and public keys are universally visible to all. The private key is a secret value and is used to access and authorize each of those "addresses", which are generally transactions. Digital signatures are used specifically for digital currencies. They are used to evaluate transactions with their signature securely (offline) and also for multi-signature digital wallet contracts in Lakchin.

The second application is about the daily use of cryptography in-home Wi-Fi and Internet disclosure. As you know, your first connection to the world is through Wi-Fi networks, so it is the first part that needs to be encrypted and protected by these home networks. WiFi networks

are encrypted by protection protocols such as WPA and WPA2. Which protects your WiFi networks from your information traffic. Modern search engines also use a protocol called Secure Sockets Layer (SSL) to secure your searches and exchanges. SSL works by using one key to encrypt and another key to decrypt, which is the same as cryptography. When you see the phrase HTTPS in the URL typing location, it means that SSL is securing the Internet away from your eyes.