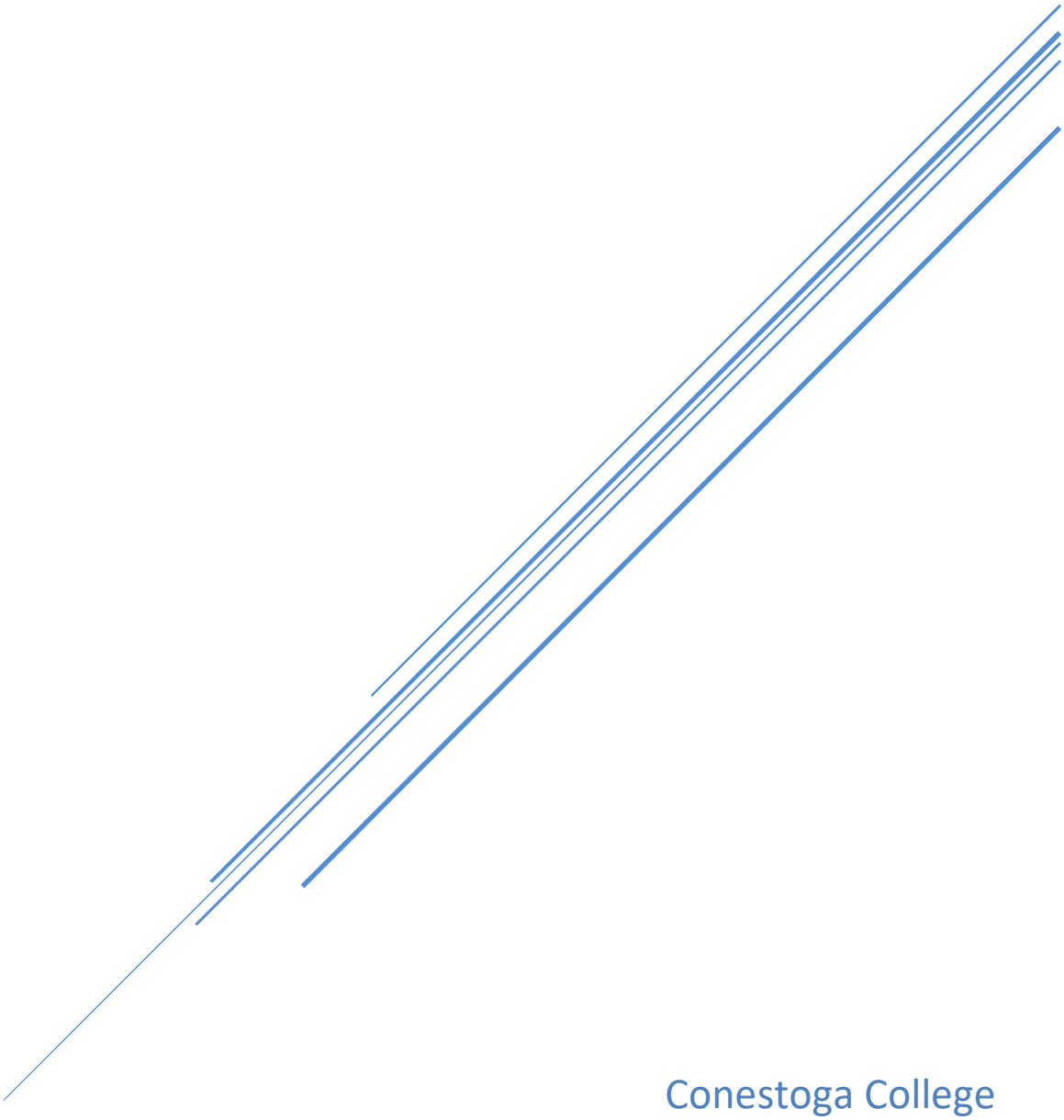
Assignment 2: Securing Communication with GPG

Name: Twinkle Akhilesh Mishra

Student Id : 8894858



Conestoga College

INFO8965: Computer and Network Security

Conestoga College	
Course	**INFO8965: Computer and Network Security**
Activity	**Securing Communication with GPG**
Student Name	**Twinkle Akhilesh Mishra**
Date performed	**04-March-2025**

Objectives

- Security transfer data using public key cryptography
- Authenticate Message Integrity and Sender Identity
- Manage and Exchange Cryptographic Keys

Resources

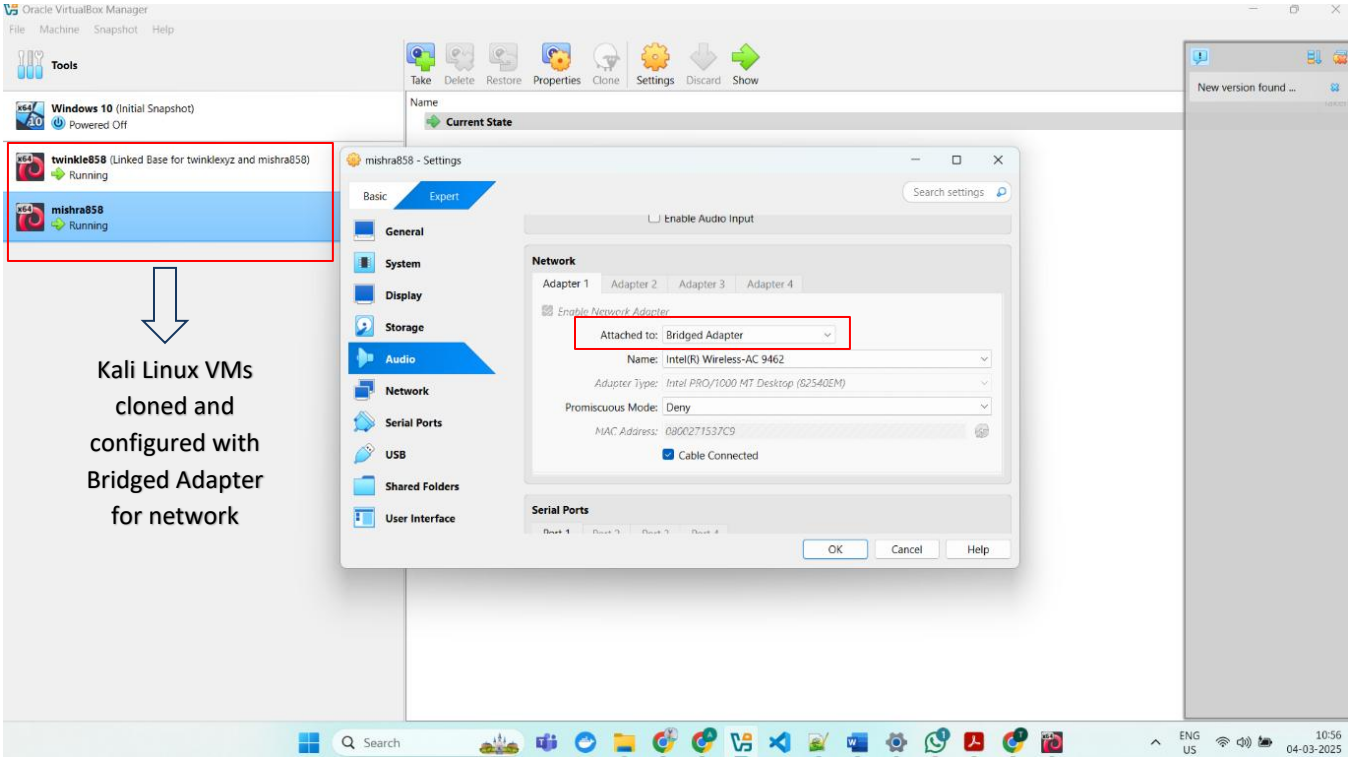
- PC / Laptop
- Virtual Box: https://www.virtualbox.org/wiki/Downloads
- Linux Kali
- GPG (GNU Privacy Guard), SSH

Note: In this lab activity, replace 'FirstName' with your actual first name(Twinkle), 'LastName(Mishra)' with your actual last name, and 'XYZ(858)' with the last three digits of your student ID.

(A) Create clones

- Open VirtualBox and select Kali Linux.
- Right-click → **Clone**.
- Create **two clones**:
  - **Sender VM** → Name it FirstNameXYZ (**twinkle858**).
  - **Receiver VM** → Name it LastNameXYZ (**mishra858**).

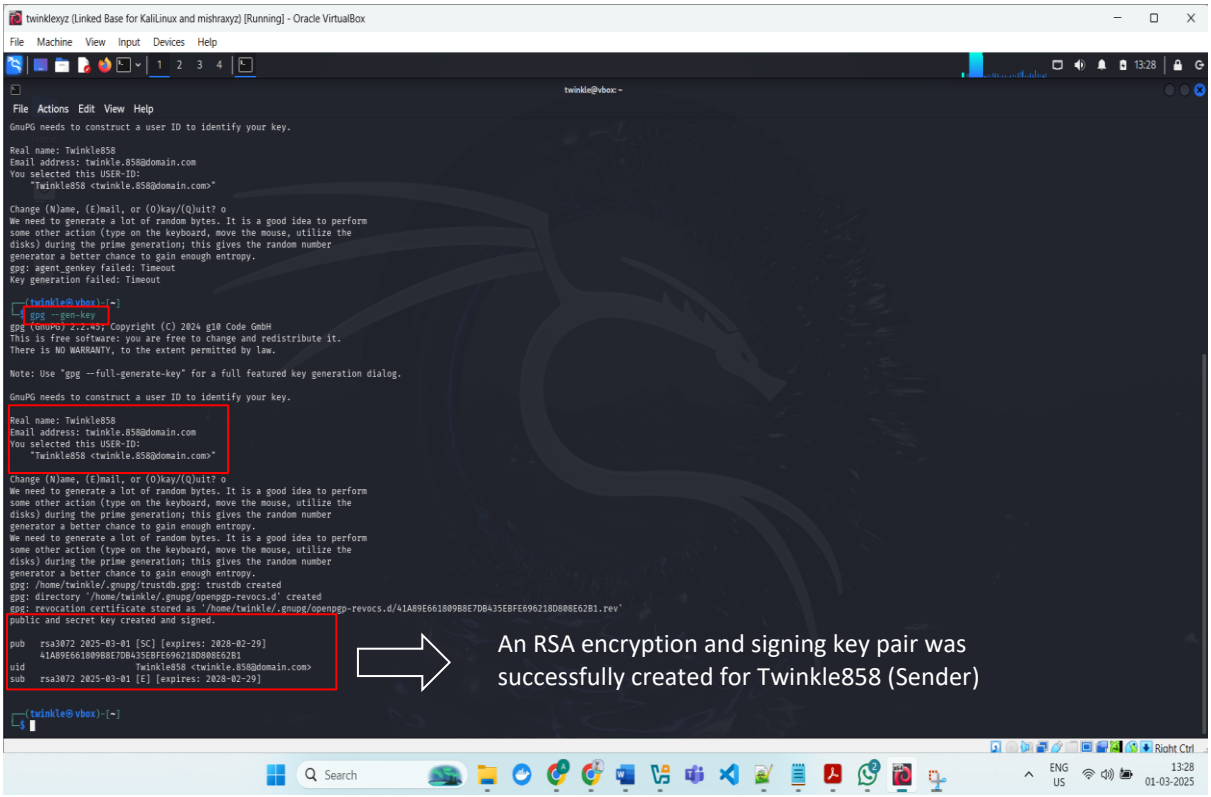
- **Choose Linked Clone** (to save disk space).
- Click **Finish**.



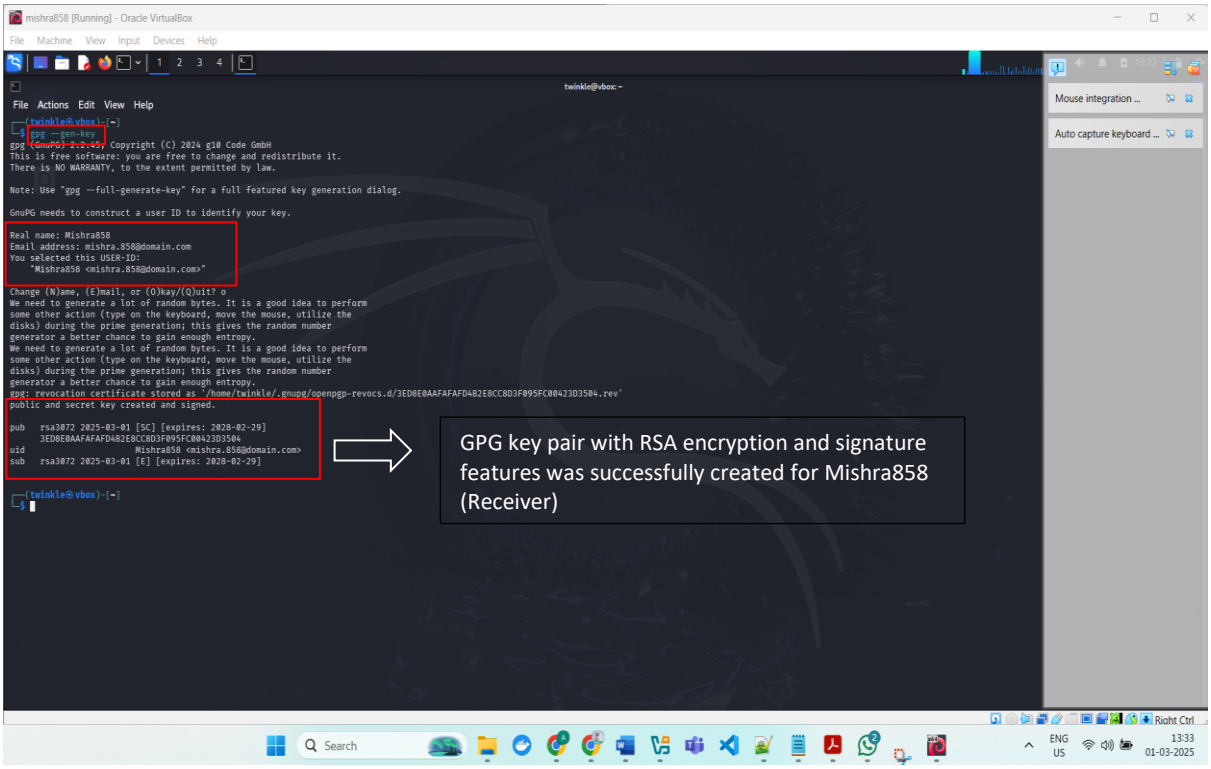
(B) Create GPG Key Pair

➢ **On the Sender VM (Twinkle858)**

- Open the terminal and type:
  gpg --gen-key
- Enter the following details when prompted:
  - Name: Twinkle858
  - Email: twinkle.858@domain.com
  - Algorithm: RSA and RSA (default)
  - Key Size: 4096
  - Expiration: Press Enter (default)
  - Passphrase: Set a secure passphrase.
  - Wait for the system to generate the key.



- **On the Receiver VM (Mishra858)**
  - Open the terminal and type:
    gpg --gen-key
  - Enter the following details when prompted:
    - Name: Mishra858
    - Email: mishra.858@domain.com
    - Algorithm: RSA and RSA (default)
    - Key Size: 4096
    - Expiration: Press Enter (default).
    - Passphrase: Set a secure passphrase.
  - Wait for the system to generate the key.

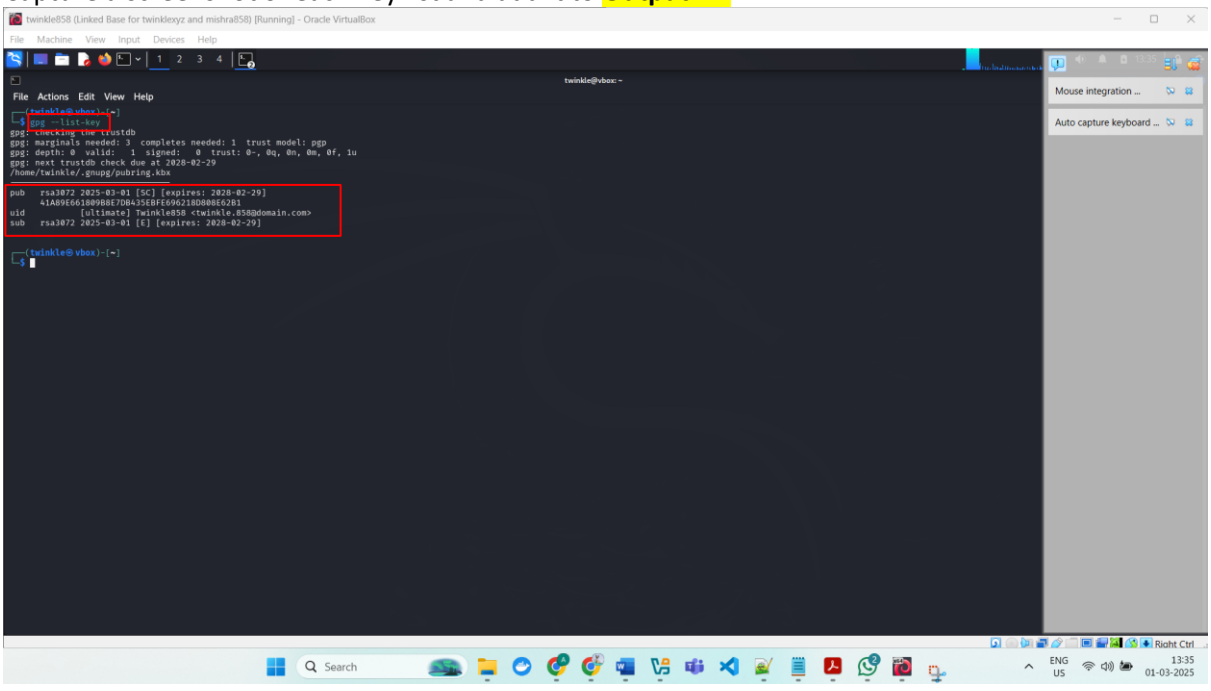


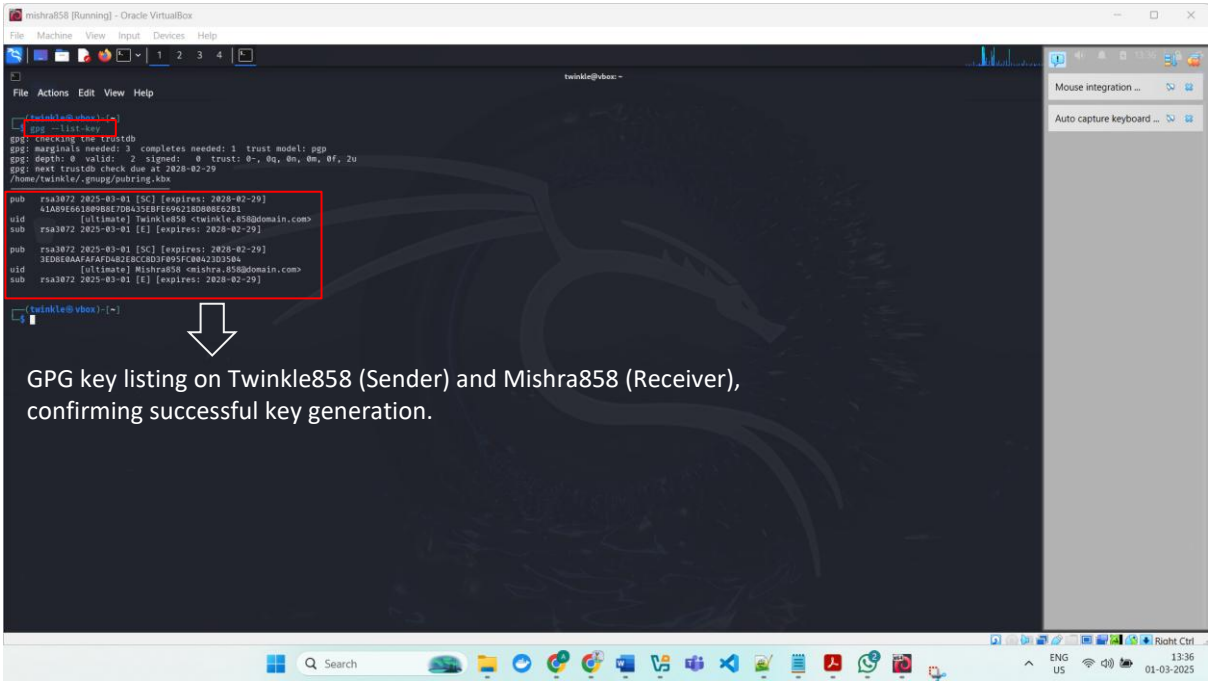
(C) List and Export Public Keys

On both VMs, list the keys with:

`gpg --list-keys`

Capture a screenshot of each key list and add it to **Output # 2**.



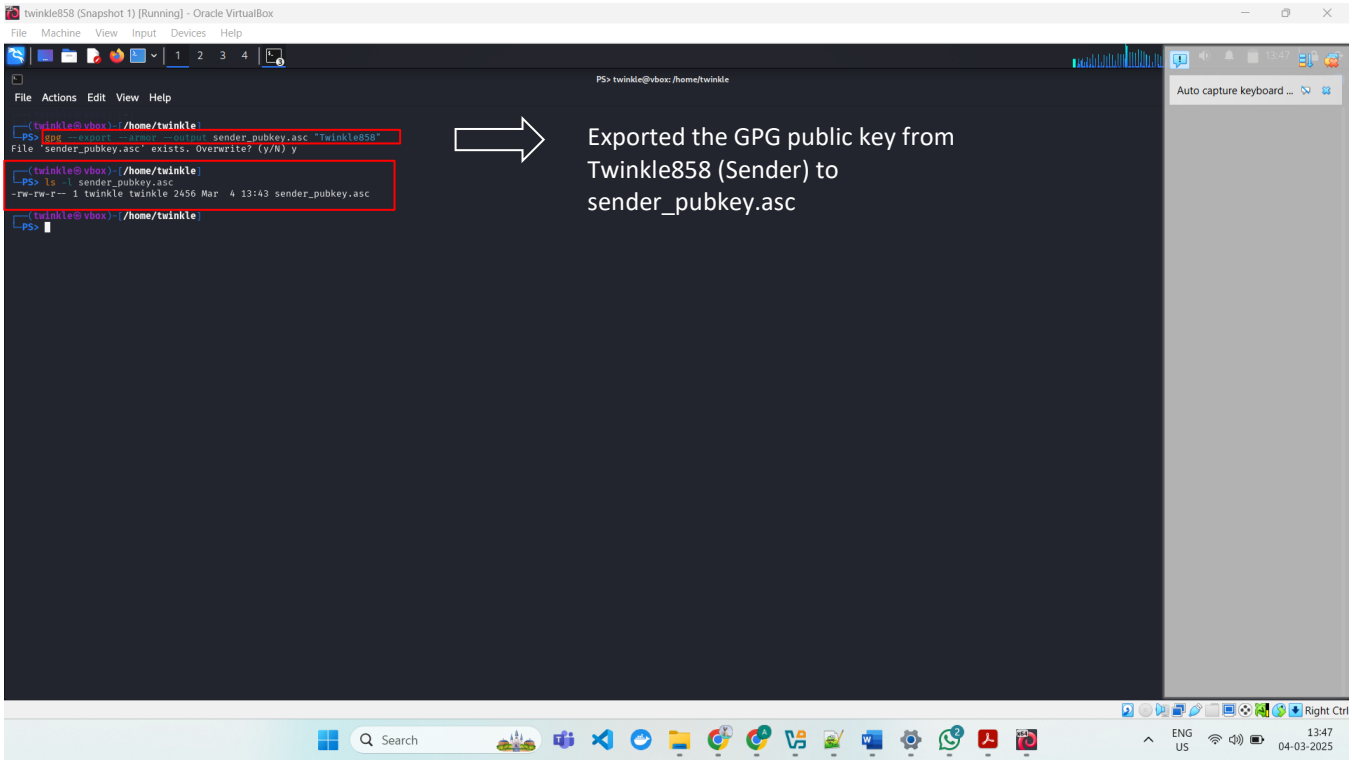


GPG key listing on Twinkle858 (Sender) and Mishra858 (Receiver), confirming successful key generation.

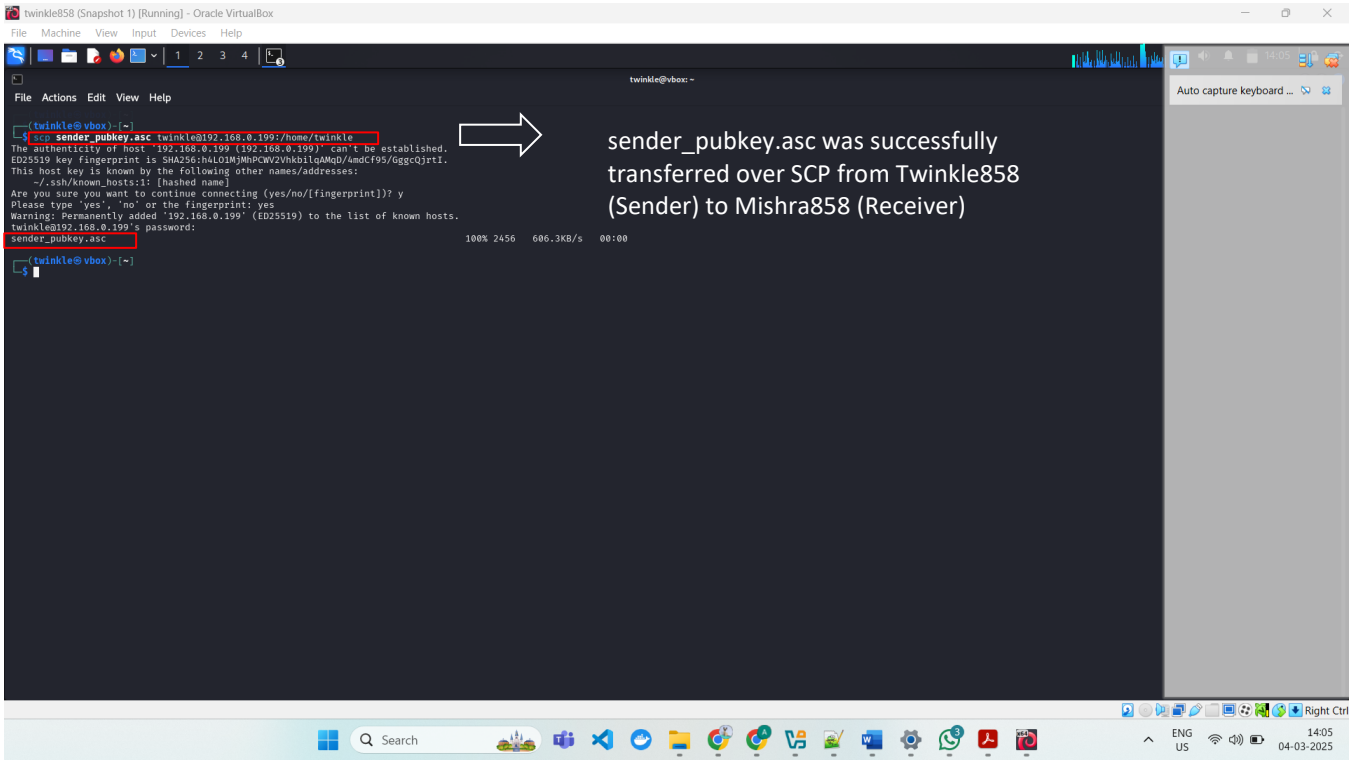
On the Sender VM, export the public key:

`gpg --export --armor --output sender_pubkey.asc "Twinkle858"`

Transfer `sender_pubkey.asc` to the Receiver VM using SCP (Secure Copy Protocol) command. Refer to online resources for more information about SCP command.



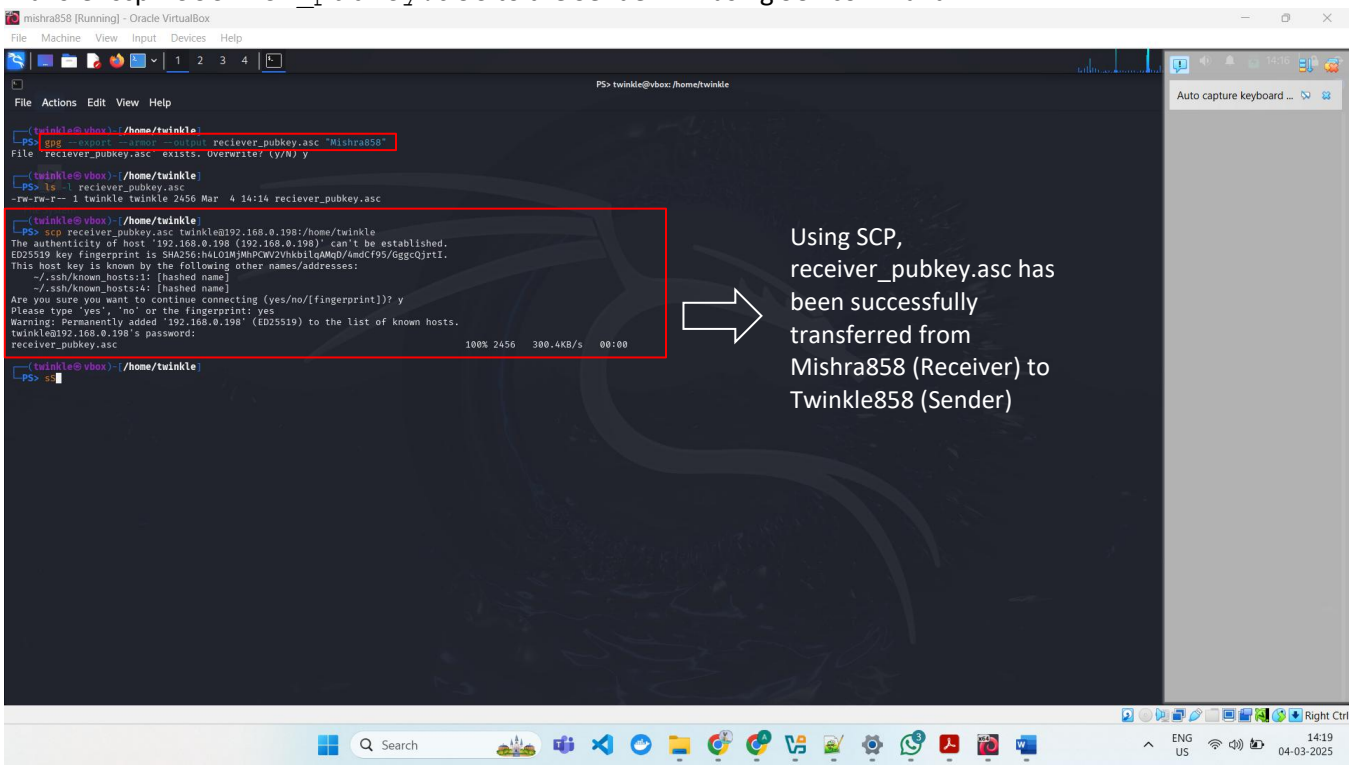
Exported the GPG public key from Twinkle858 (Sender) to sender_pubkey.asc



On the Receiver VM, export the public key:

```
gpg --export --armor --output receiver_pubkey.asc "Mishra858"
```

Transfer scp `receiver_pubkey.asc` to the Sender VM using SCP command.

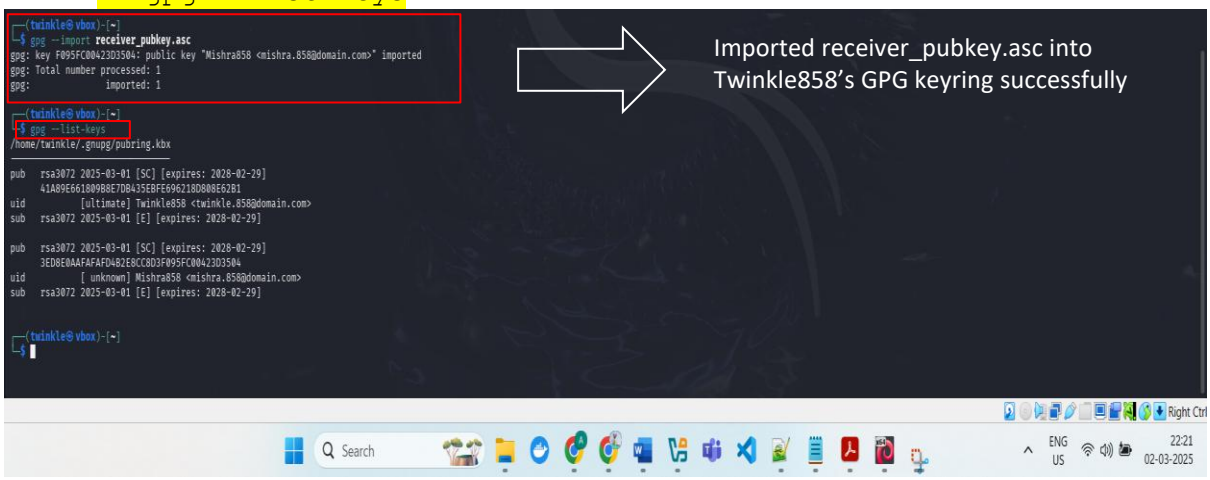


(D) Import public keys

Output#3

- On the Sender VM, import the Receiver's public key:
  - `gpg --import receiver_pubkey.asc`

- Verify the imported key with:
  - `gpg --list-keys`

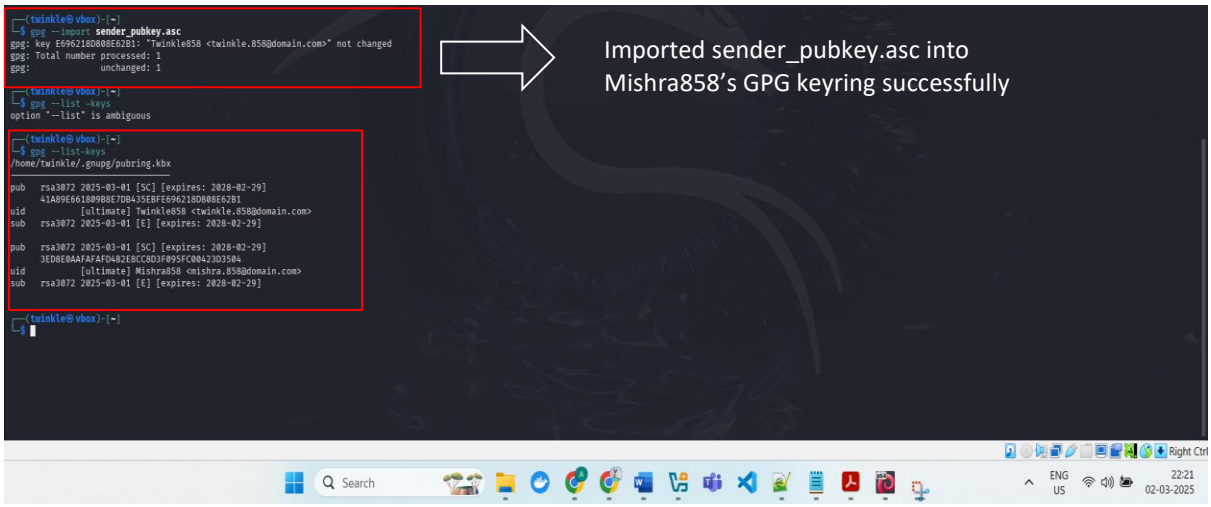


On the Receiver VM, import the Sender's public key:

- `gpg --import sender_pubkey.asc`

Verify the imported key:

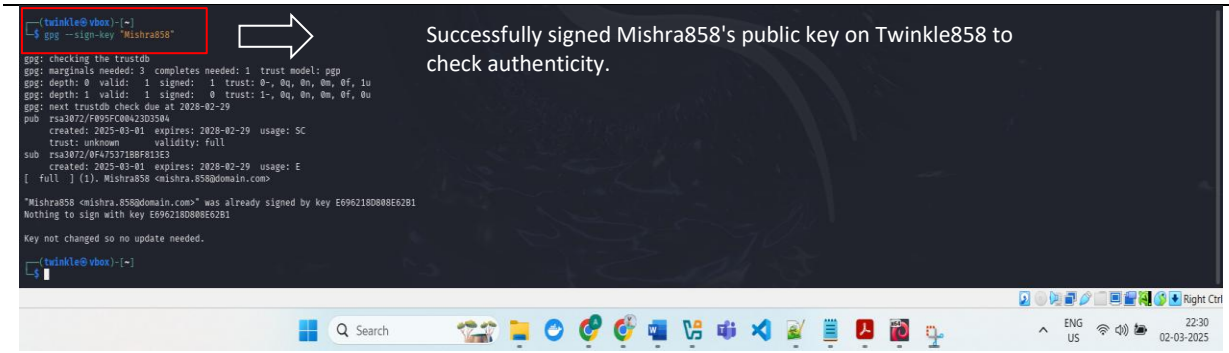
- `gpg --list-keys`



(E) Sign Receiver's public key

- On the Sender VM, sign the Receiver's public key to verify its authenticity:
  - **`gpg --sign-key "Mishra858"`**

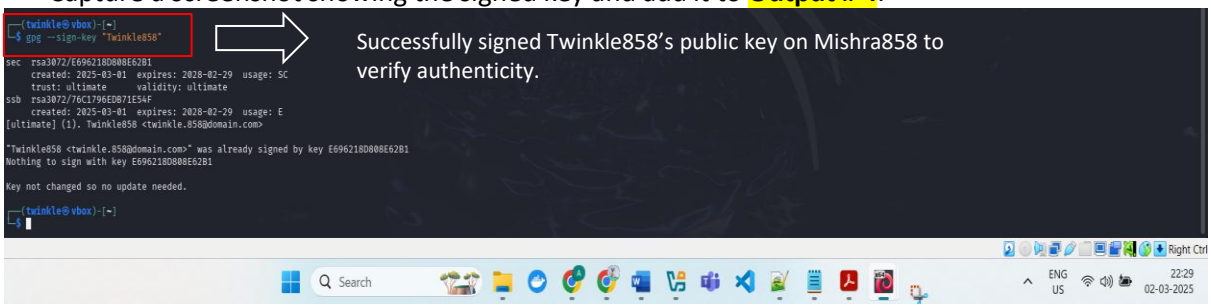
Capture a screenshot showing that the key has been signed and add it to **Output # 4**.



On the Receiver VM, sign the Sender's public key:

- **`gpg --sign-key "Twinkle858"`**

Capture a screenshot showing the signed key and add it to **Output # 4**.



- Explain the advantage of signing receiver's public key in this context. [2 mark]
- Signing a public key ensures that:
  - ✓ Authenticity is Verified
    - It confirms that the key truly belongs to the claimed identity.
    - Prevents man-in-the-middle attacks where someone intercepts and replaces keys.
  - ✓ Trust Establishment
    - The recipient can see that you have verified and trusted this key.
    - Helps build a web of trust between users.
  - ✓ Prevents Spoofing
    - No one can impersonate the key owner without a valid signature.

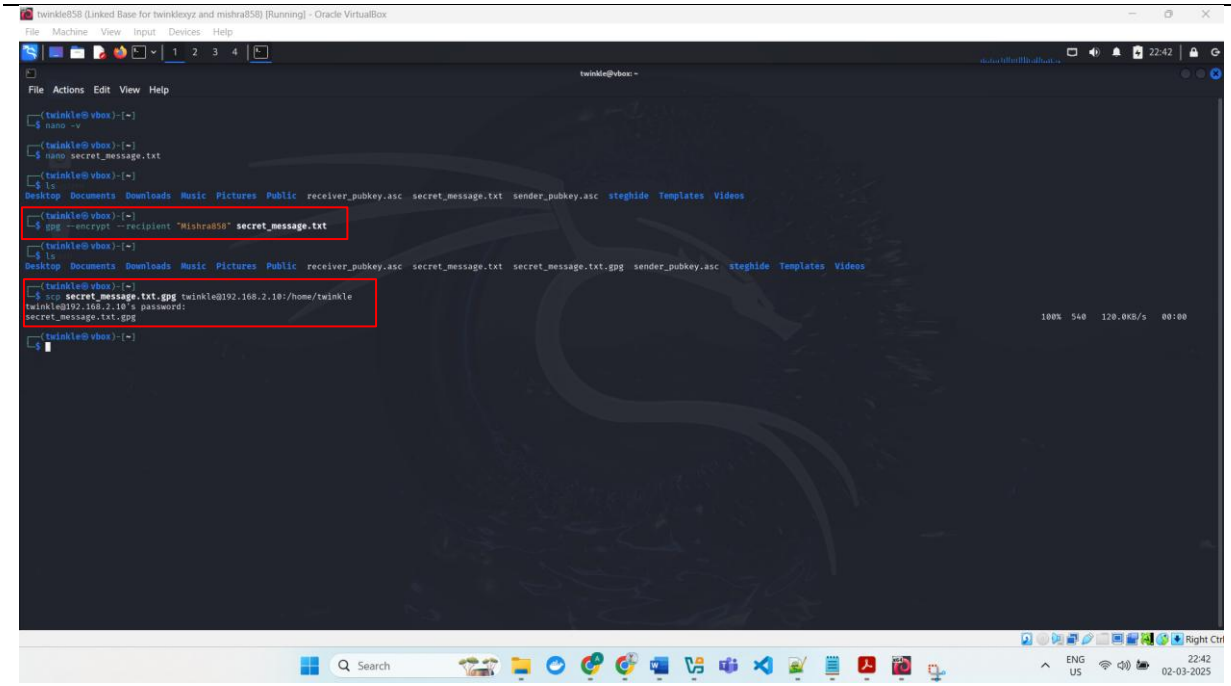
  Signing with public key certifies its legitimacy and establishes a trust network. It will help to protect against man-in-the-middle attacks and ensures secure communication between the sender and receiver.

(F) Encrypt and Decrypt a File

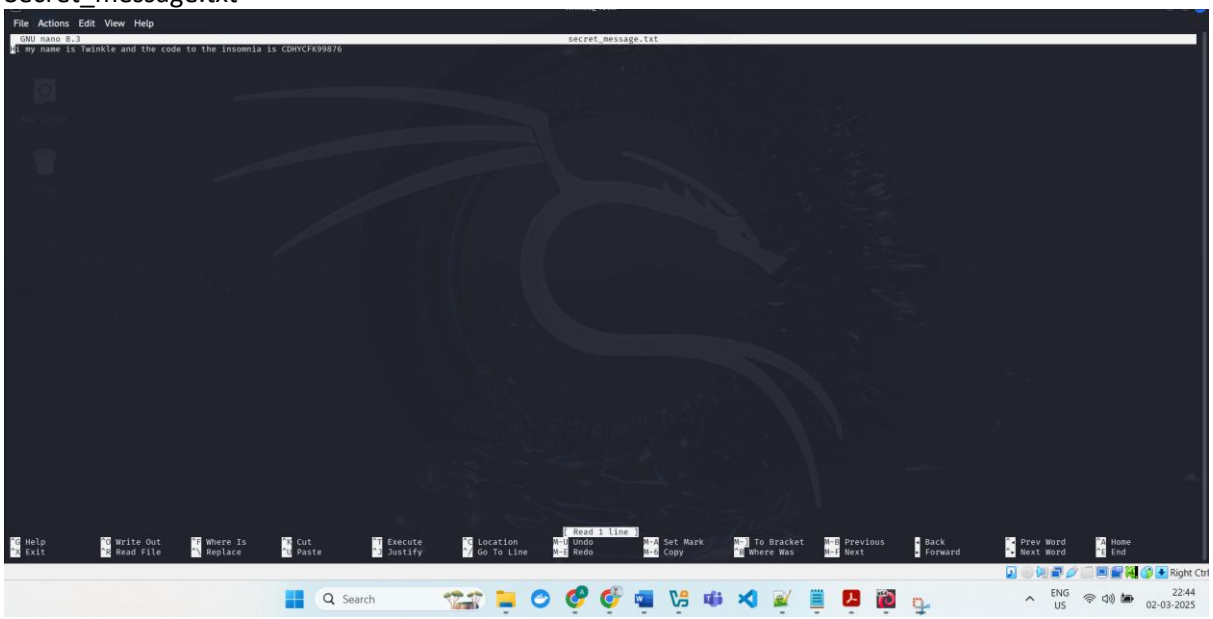
On the Sender VM, create a text file named secret_message.txt. Add some confidential message to this file. Encrypt the file using the Receiver's public key:

- `gpg --encrypt --recipient "Mishra858" secret_message.txt`

This will create an encrypted file `secret_message.txt.gpg`. Capture a screenshot of the encrypted file and add it to **Output # 5**.



Secret_message.txt



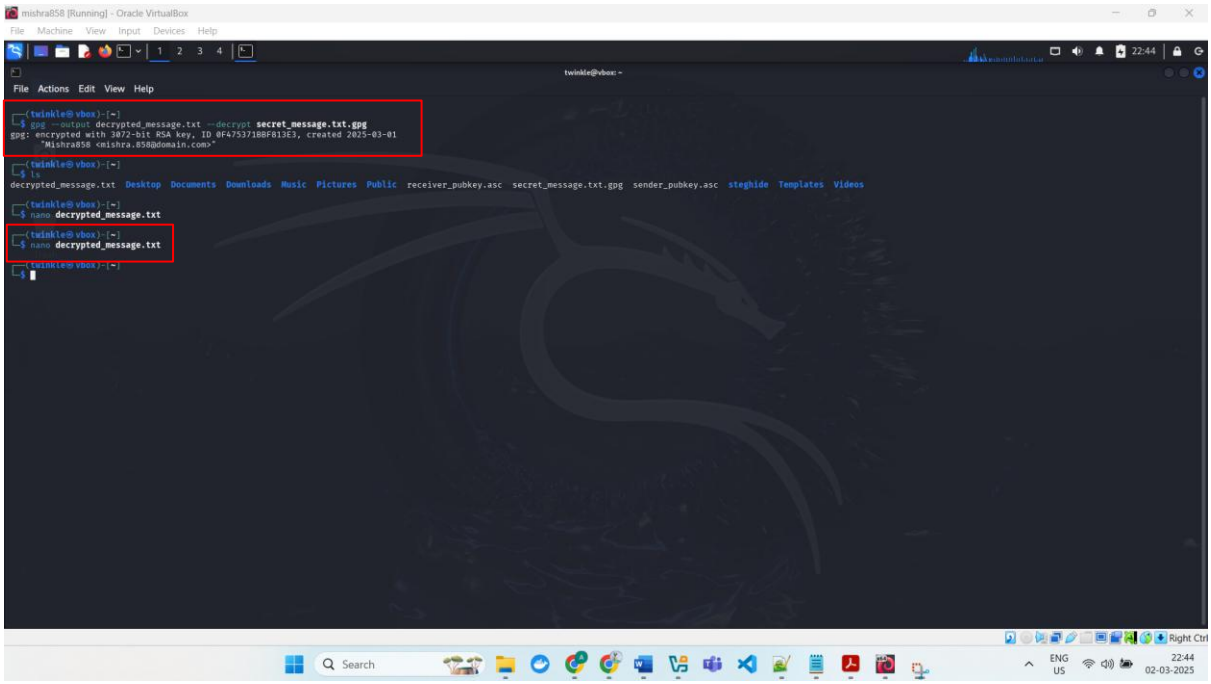
Transfer `secret_message.txt.gpg` from the Sender VM to the Receiver VM using SCP.

On the Receiver VM, decrypt the file with the command:

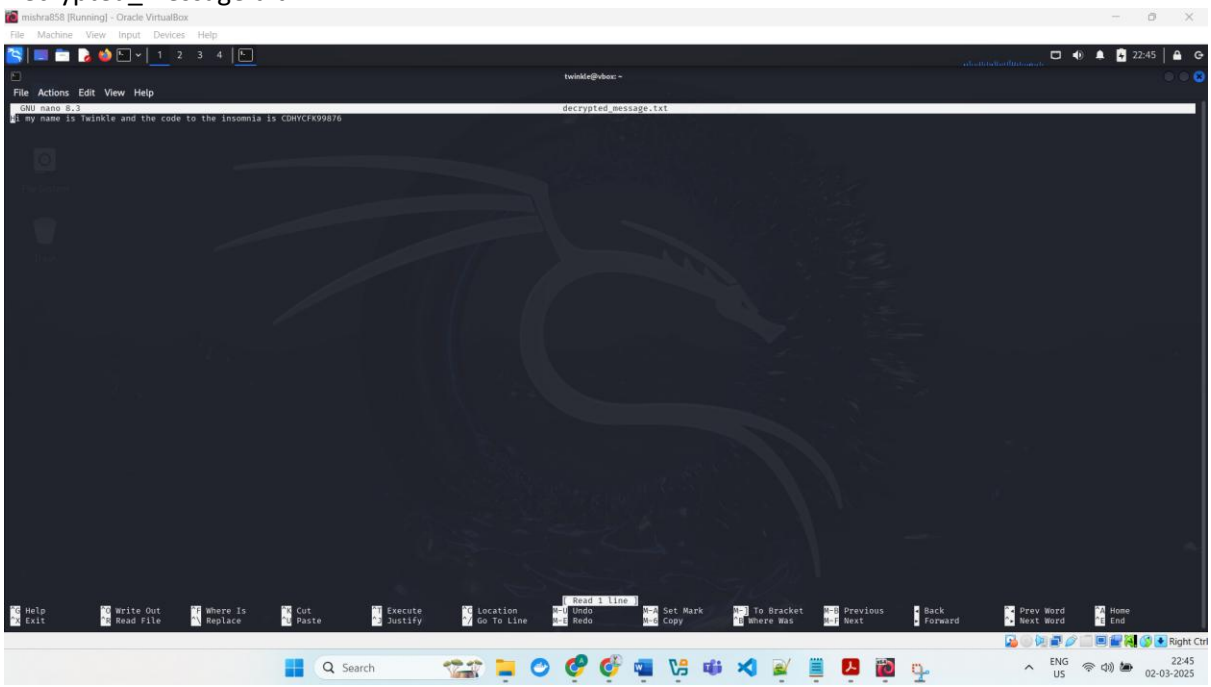
- `gpg --output decrypted_message.txt --decrypt secret_message.txt.gpg`

Capture a screenshot of the decrypted file content and add it to **Output # 5** [2 marks]

Receiver VM



Decrypted_message.txt



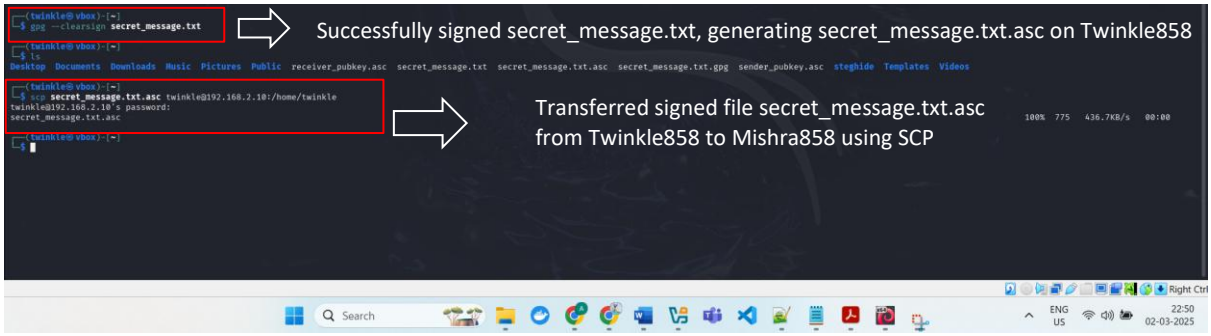
(F) Verify File Signature

- On the Sender VM, sign the secret_message.txt file:

- **`gpg --clearsign secret_message.txt`**

Transfer the signed file `secret_message.txt.asc` to the Receiver VM.

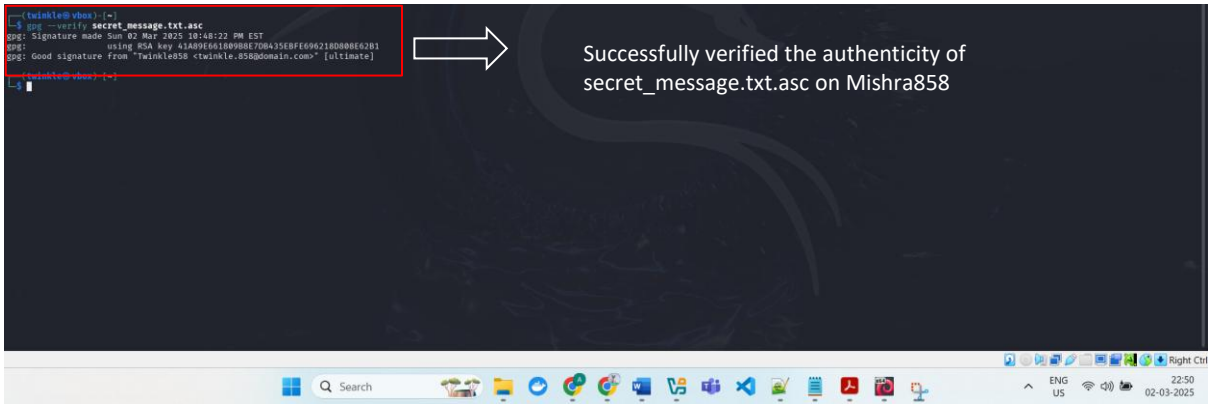
Sender VM



- On the Receiver VM, verify the signature:

- **`gpg --verify secret_message.txt.asc`**

Capture a screenshot showing successful verification of the signature and add it to **Output # 6**.



- Explain the purpose and reason of verifying file signatures in this context. [2 marks]
  The following are the purpose and reason of verifying file signatures:

  ✓ Ensures File Integrity
  - Verifying the signature confirms that file has not been modified during transfer.

  ✓ Authenticates the Sender
  - It ensures that Twinkle858 (Sender) is the actual creator of the signed file.

  ✓ Preventing Attacks and Tampering
  - If the file is not verified, the attacker may change or replace it without anyone knowledge.

  Verifying file signatures ensures the sender's validity as well as the file's integrity. It avoids manipulation and certifies that the file was not changed during transmission.