# Cryptography Algorithm Documentation

## Introduction

This document introduces a unique cryptographic algorithm designed to ensure complete security of encrypted data and encryption keys. Even if the encrypted key falls into the hands of a third party, it remains impossible to decrypt the data. This is achieved by encrypting the key itself using the algorithm's internal mechanisms and subsequently securing it with the AES-128 standard.

The algorithm is implemented in C# and adheres to modern security requirements, offering high performance and flexibility.

## Key Features

1. \*\*Secure Key Encryption\*\*  
 - The key is generated and encrypted using a default key and the algorithm's internal mechanisms.  
 - The encrypted key is then further secured with the AES-128 standard.

2. \*\*Complete Data Protection\*\*  
 - Decrypting the data is impossible without access to the encrypted key and the proprietary algorithm.  
 - The unique structure of the algorithm ensures security at every stage.

3. \*\*Advanced Encryption Standard (AES-128)\*\*  
 - The final encryption of the key uses the AES-128 standard.  
 - AES-128 provides a widely recognized balance of speed and security.

4. \*\*Recursive Encryption Mechanism\*\*  
 - The key undergoes multiple layers of encryption through the algorithm.  
 - This recursive mechanism ensures unauthorized decryption is entirely impossible.

5. \*\*C# Implementation\*\*  
 - The algorithm is written in C#, making it suitable for a wide range of applications.  
 - It ensures a high-performance and secure system.

## How the Algorithm Works

1. \*\*Key Generation\*\*  
 - The system generates a random key, which is used for encrypting the data.

2. \*\*Initial Key Encryption\*\*  
 - The key is encrypted using the system's default key and the algorithm's internal mechanisms.  
 - This ensures a unique result for each key.

3. \*\*AES-128 Encryption of the Key\*\*  
 - The initially encrypted key is further secured using the AES-128 standard.  
 - This additional step prevents the encrypted key from being analyzed or brute-forced.

4. \*\*Data Encryption\*\*  
 - Data is encrypted using the randomly generated key with the algorithm's high-performance mechanisms.

5. \*\*Storage and Decryption\*\*  
 - Both the encrypted data and the AES-128 protected key are stored or transmitted.  
 - Decryption requires access to both the encrypted key and the proprietary algorithm.

## Advantages

1. \*\*Multi-Level Security\*\*  
 - Encrypting the key with both internal mechanisms and AES-128 raises security to maximum levels.

2. \*\*Resistance to Brute-Force Attacks\*\*  
 - The recursive mechanism and AES-128 standard ensure complete protection against brute-force attacks.

3. \*\*High Compatibility\*\*  
 - The algorithm can be easily integrated into desktop and web applications.

4. \*\*Flexible and Scalable Design\*\*  
 - Suitable for various use cases, such as data encryption, secure data transmission, and database protection.

5. \*\*Ease of Implementation in C#\*\*  
 - The algorithm provides simple and fast integration into C# projects.

## Conclusion

This cryptographic algorithm offers a robust solution to modern security challenges. By combining unique internal mechanisms with the AES-128 standard, it ensures complete protection of both data and keys. Users can rely on this algorithm to safeguard their information with confidence, knowing that unauthorized decryption is rendered impossible.

This algorithm is a state-of-the-art example of security technology.

## Command Analysis and Usage

This section provides an in-depth explanation of the commands available in the Cryptography-App. Each command is analyzed in detail along with practical usage examples.

### --Win32

Ensures that file and folder names are encrypted or decrypted using Windows-compatible naming conventions. If omitted, the app uses Linux-compatible naming conventions.  
  
Example Usage:  
.\Cryptography-app-original.exe --txt Salam --Win32  
Encrypts the word 'Salam' in Windows-compatible format.  
.\Cryptography-app-original.exe -K <Key> -d --D <Directory> --Win32  
Decrypts the directory contents using Windows-compatible format.

### --help or -h

Displays a list of all available commands and their descriptions.  
  
Example Usage:  
.\Cryptography-app-original.exe --help  
.\Cryptography-app-original.exe -h

### --aes

Applies an additional AES encryption layer during encryption or removes it during decryption.  
  
Example Usage:  
.\Cryptography-app-original.exe --txt Salam --Win32 --aes  
Encrypts the word 'Salam' in Windows-compatible format with AES encryption.  
.\Cryptography-app-original.exe -K <Key> -d --txt <ciphertext> --aes  
Decrypts the AES-encrypted ciphertext with the provided key.

### --txt

Specifies that the encryption or decryption process is for a single word or text input.  
  
Example Usage:  
.\Cryptography-app-original.exe --txt salam  
Encrypts the word 'salam' with default settings.  
.\Cryptography-app-original.exe -K <Key> --txt <plaintext>  
Encrypts the plaintext with the provided key.

### -D (Directory Mode)

Encrypts or decrypts the files and folder names within a specified directory.  
  
Example Usage:  
.\Cryptography-app-original.exe -K <Key> --D <Directory>  
Encrypts all files and folder names in the specified directory.  
.\Cryptography-app-original.exe -K <Key> -d --D <Directory> --Win32  
Decrypts all files and folder names in Windows-compatible format.

### -f (File Mode)

Encrypts or decrypts the contents of a specified file.  
  
Example Usage:  
.\Cryptography-app-original.exe -K <Key> -f <filePath>  
Encrypts the specified file using the provided key.  
.\Cryptography-app-original.exe -K <Key> -d -f <filePath>  
Decrypts the specified file using the provided key.

### -K (Key Specification)

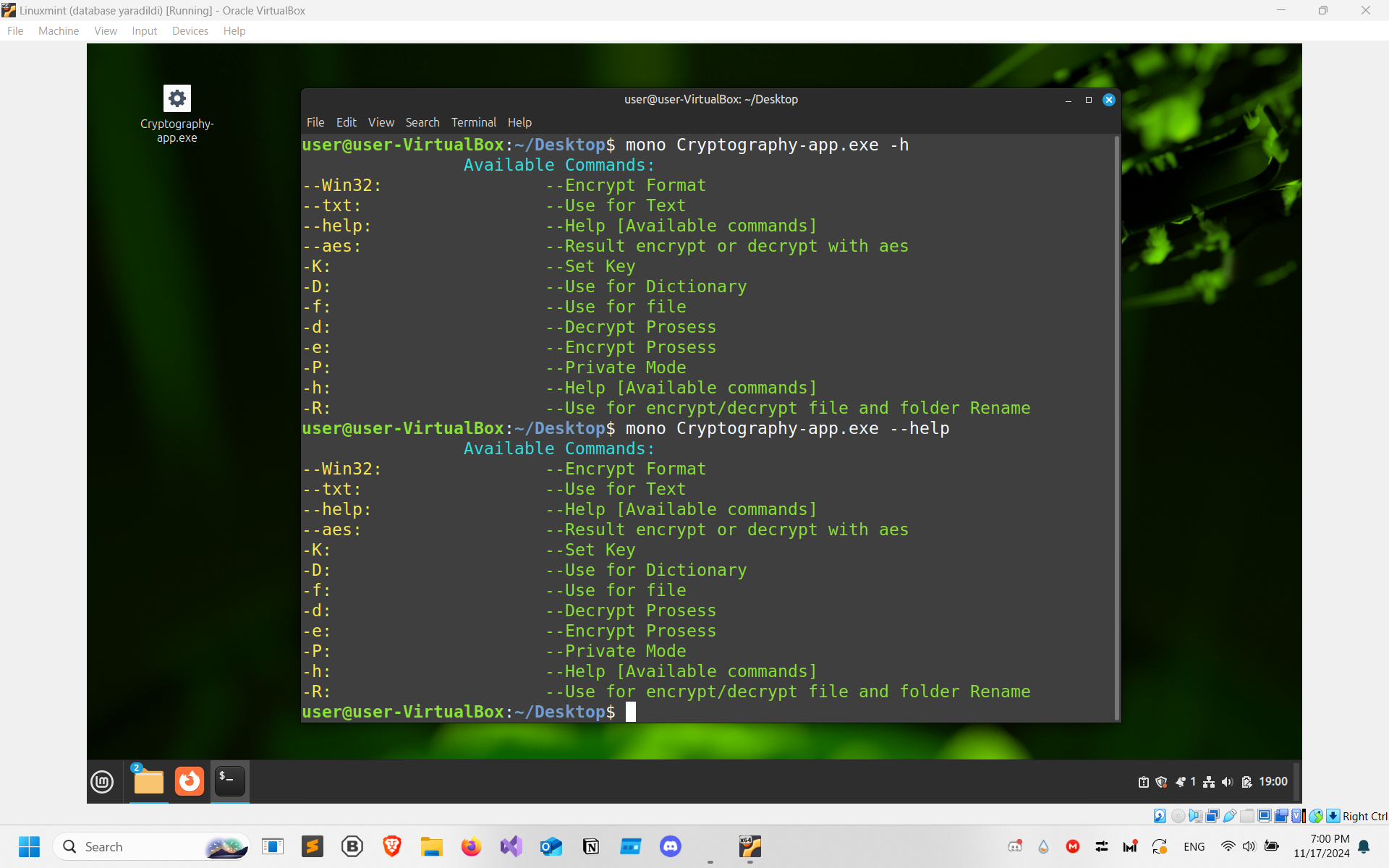
Allows specifying a key for encryption or decryption.  
  
Example Usage:  
.\Cryptography-app-original.exe -K <Key> --txt <plaintext>  
Encrypts the plaintext with the provided key.  
.\Cryptography-app-original.exe -K <Key> -d --txt <ciphertext>  
Decrypts the ciphertext using the provided key.

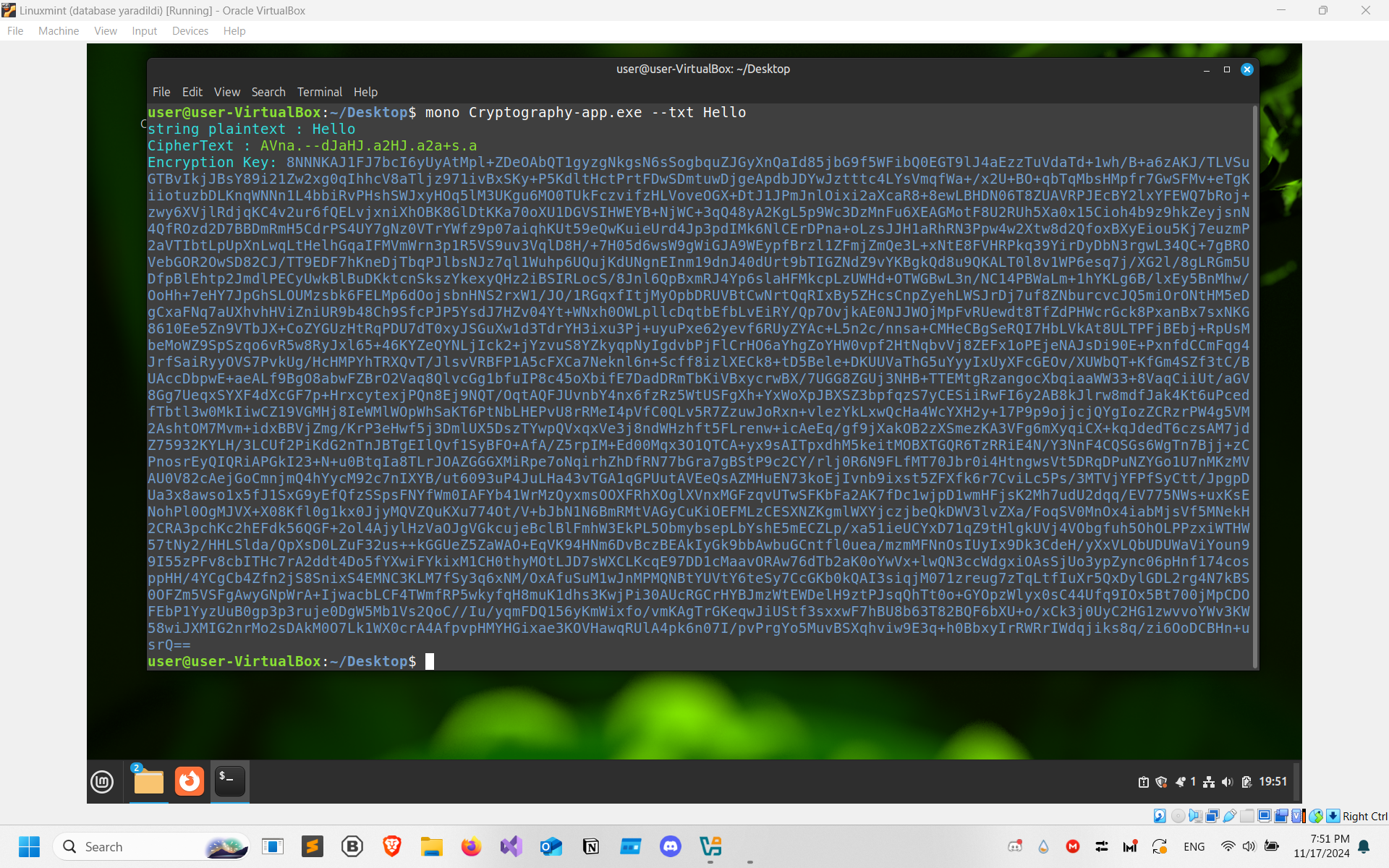
### -P (Private Mode)

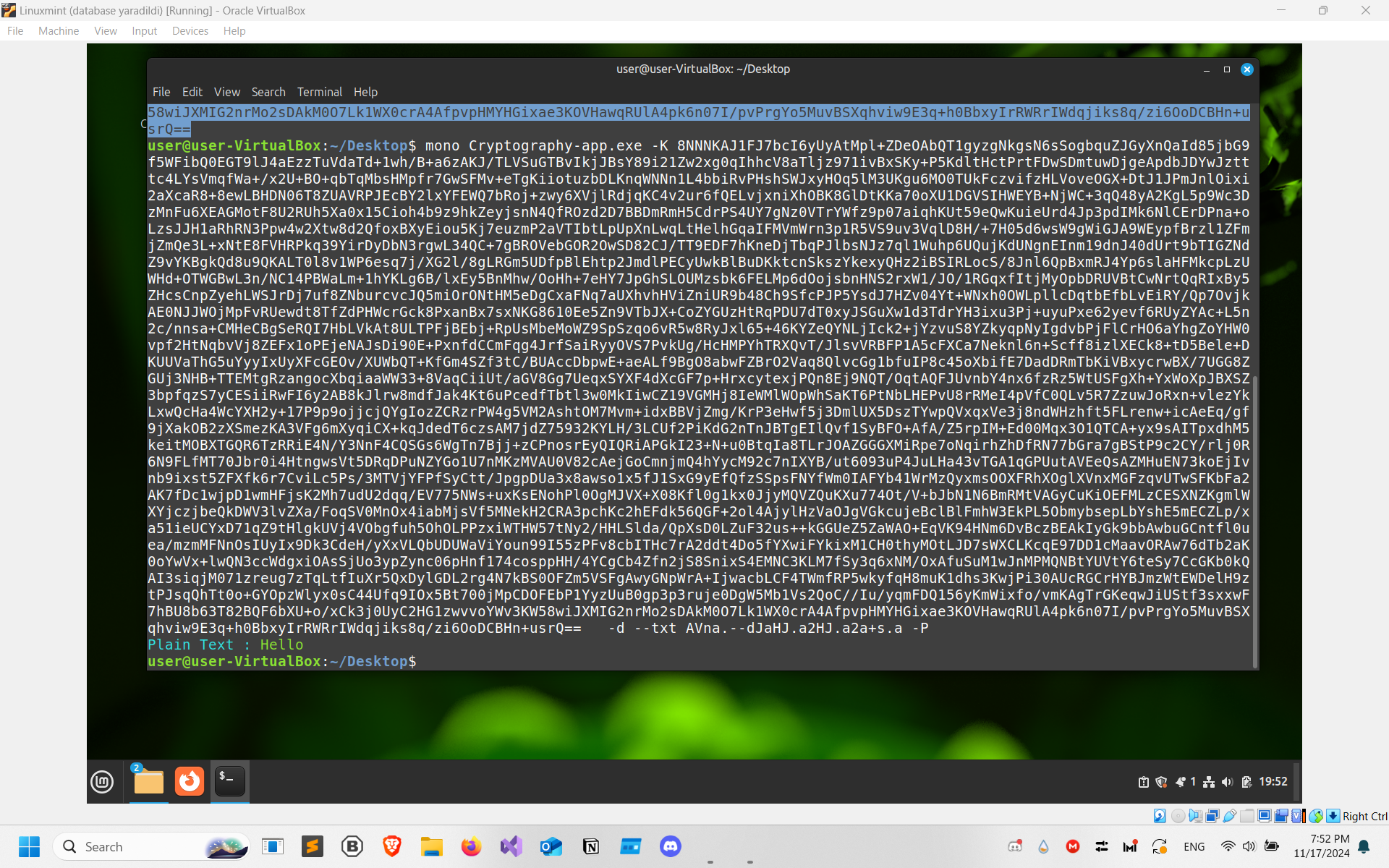
Activates private mode, suppressing process logs and output. Only results (e.g., plaintext or ciphertext) are displayed.  
  
Example Usage:  
.\Cryptography-app-original.exe --txt Salam --Win32 -P  
Encrypts the word 'Salam' with no logs or key display.  
.\Cryptography-app-original.exe -K <Key> -d --txt <ciphertext> -P --aes  
Decrypts AES-encrypted ciphertext in private mode.

### -R (Rename Mode)

Encrypts or decrypts the names of files and folders without affecting their contents.  
  
Example Usage:  
.\Cryptography-app-original.exe -R --D <Directory>  
Encrypts or decrypts all file and folder names in the specified directory.  
  
**EXAMPLES:**

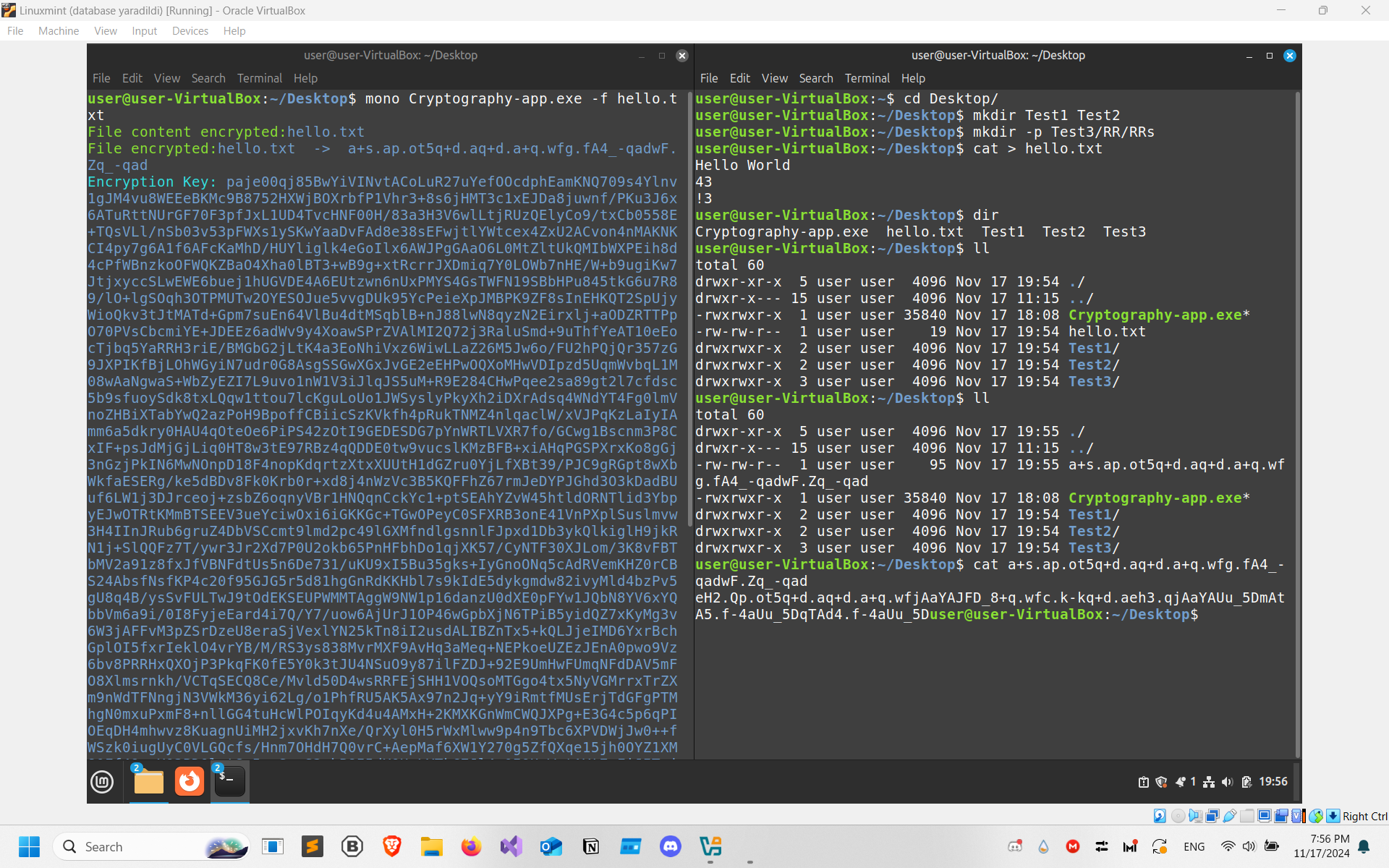






A screen shot of a computer

Description automatically generated

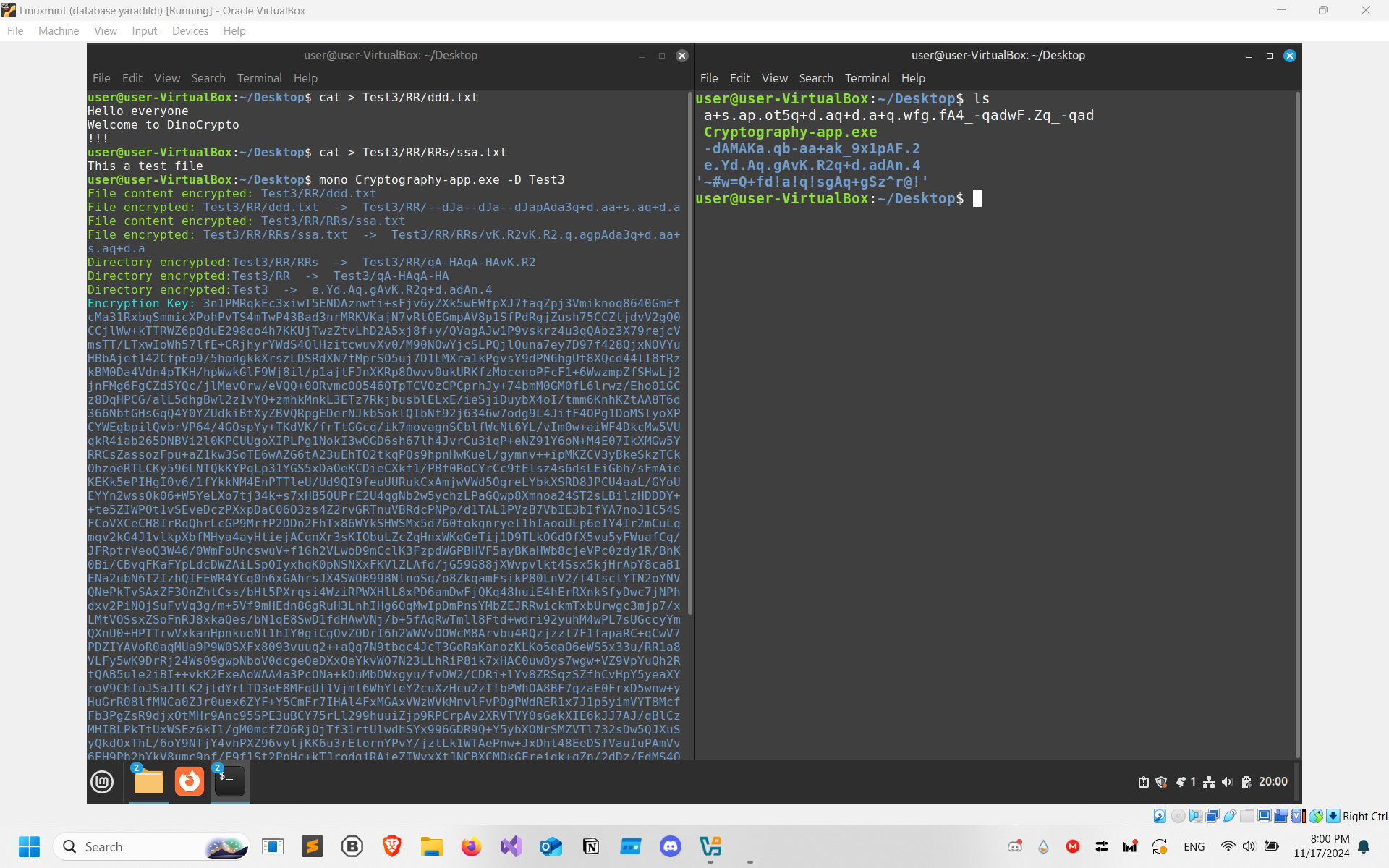


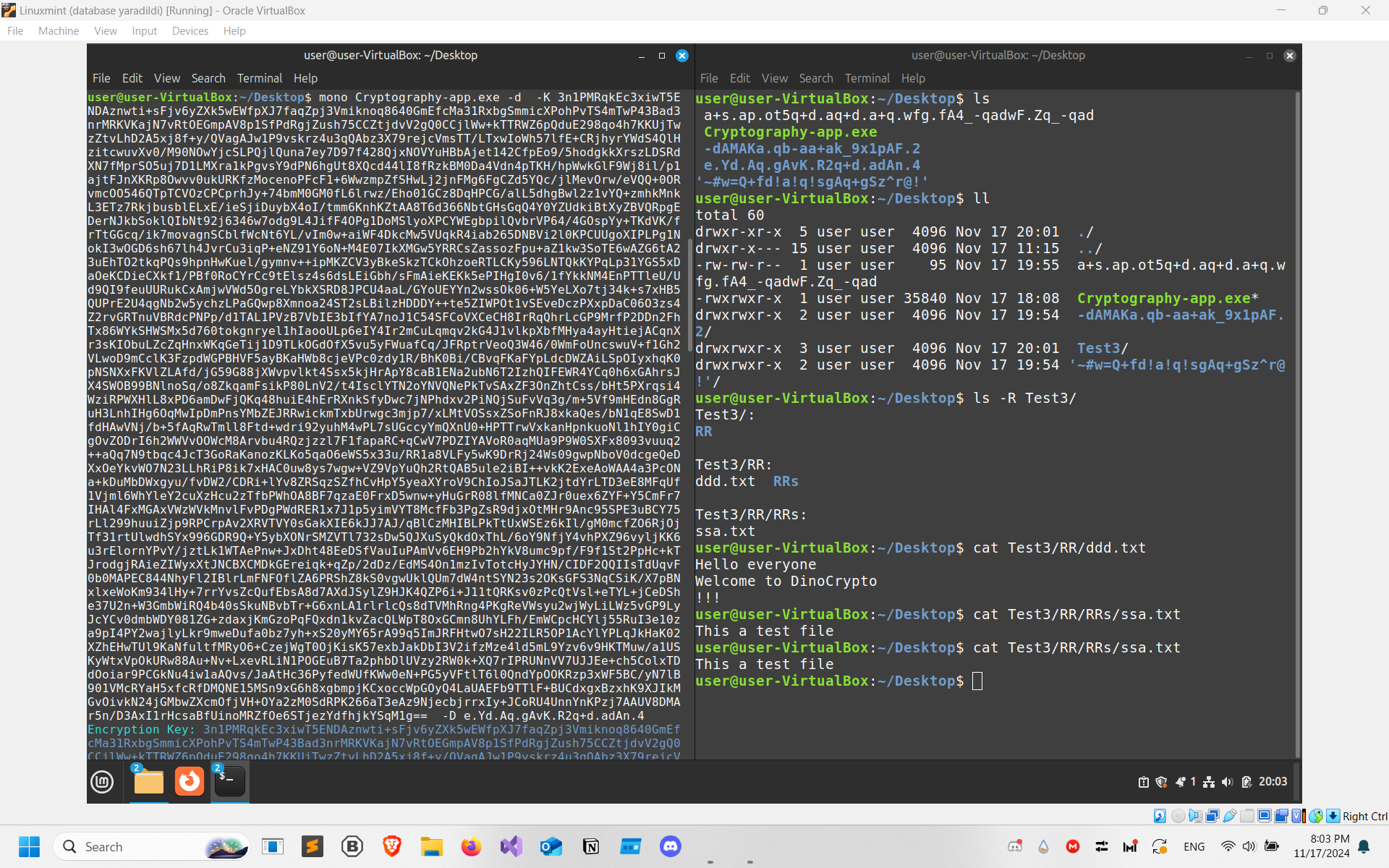
A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated





A screen shot of a computer

Description automatically generated