

**CY2002**

**Digital Forensics**

**Project**

**EFS Key Parser**

**Submitted by:** Zaid Ahmed, Umar Ahmed, Samiullah Butt, Ashfaq Ahmed

**Roll number:** 22i-7444, 22i-1743, 22i-1663, 22i-1616

**Date:** 19-Nov-2024

Contents

[Introduction: 3](#_Toc182931759)

[Steps & Details: 3](#_Toc182931760)

[Conclusion: 10](#_Toc182931761)

[References: 10](#_Toc182931762)

# 

# Introduction:

In this project, we aim to analyze and decrypt encrypted files from a Windows system that uses the **Encrypting File System (EFS)** for file encryption. The objective is to demonstrate the extraction of **EFS keys**, decryption of encrypted files, and the use of various digital forensics tools to assist in the process. The project is divided into two parts:

* **Part A** focuses on identifying encrypted files on a Windows system, extracting the **File Encryption Key (FEK)** from the file's logged utility stream, and using it to decrypt the encrypted files. This process involves using tools such as **Sleuth Kit** and **AES decryption** to retrieve the original content of the encrypted files.
* **Part B** focuses on the extraction of the **private key** from a certificate, which is essential for decrypting the FEK. It involves advanced techniques using tools like **Mimikatz** and **OpenSSL** to extract and manage the key information, allowing us to finally access and decrypt the encrypted files.

By the end of this project, we will have demonstrated a step-by-step method for obtaining the private key from the certificate, decrypting the FEK, and using that FEK to decrypt the file's contents.

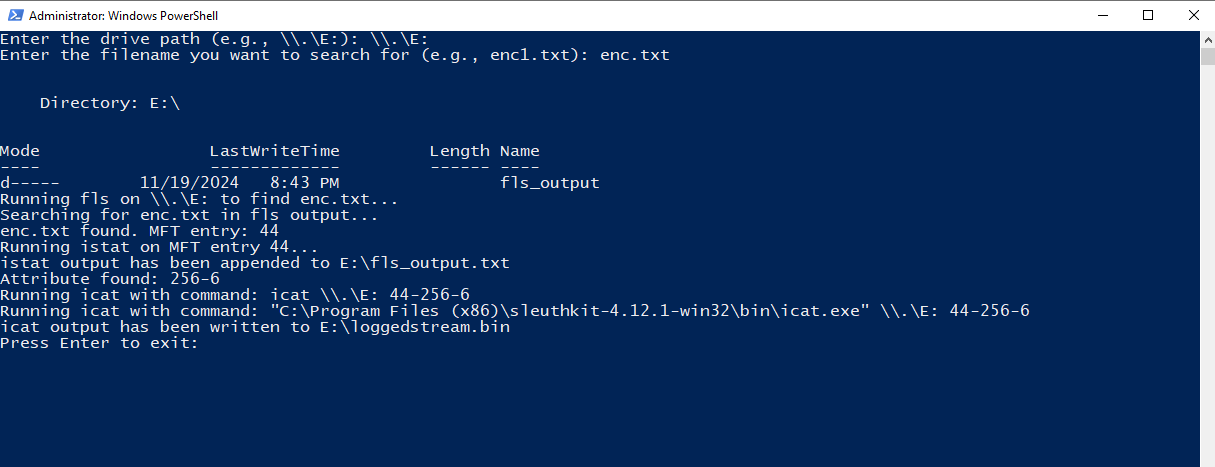
# Steps & Details:

**Part A:**

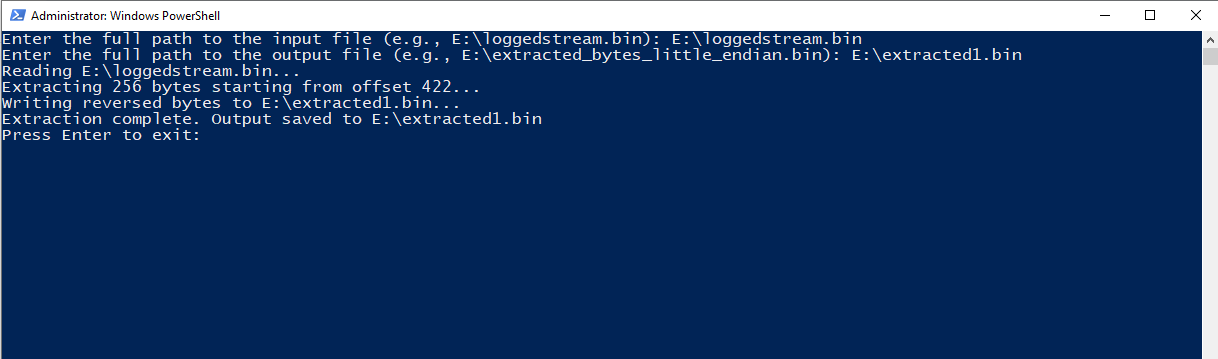
When we first run the code, it will search the entire system for the encrypted files and will print those encrypted files.



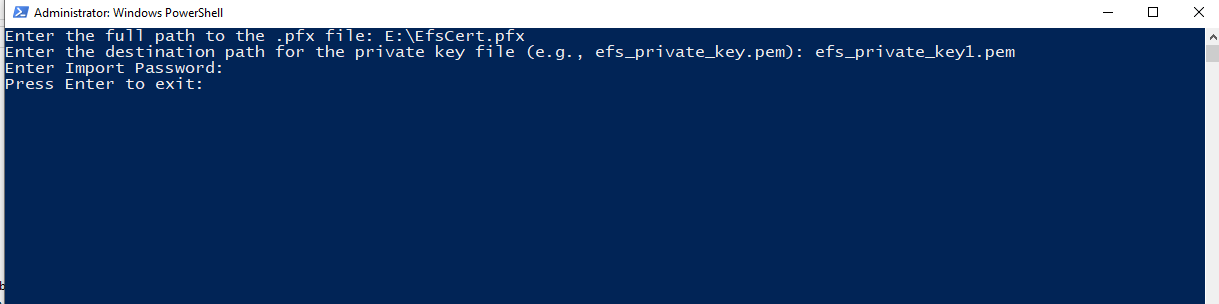
The second script then input for the drive path as an example we provide it with [\\E](file:///\\E): drive. Based on the input we provide, it exports the logged utility stream 0x100 attribute using sleuth kit. **The Sleuth Kit (TSK)** is a popular collection of forensic tools used to investigate and analyze digital evidence. It is mainly used by forensic investigators, security experts, and law enforcement agencies to perform **digital forensics** on various types of devices, such as computers, hard drives, and other digital storage media.



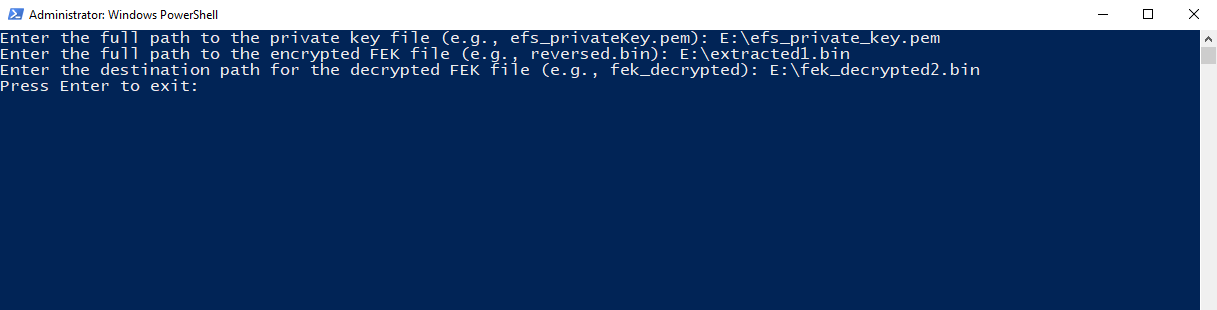
The next script exports the encrypted FEK from the logged utility stream file. It stores it in reversed order while keeping little endian in mind. FEK is itself encrypted using the Public key.



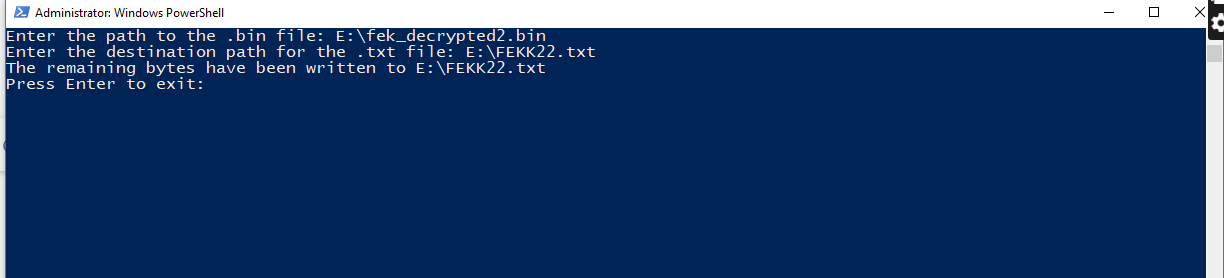
As we already have the certificate, the tool extracts the private key from the EFS certificate and stores it.



At this point we have the private key and encrypted **FEK**, we will use the user’s private key that we extracted earlier, to decrypt the **FEK** and stores it to the **.bin** file.

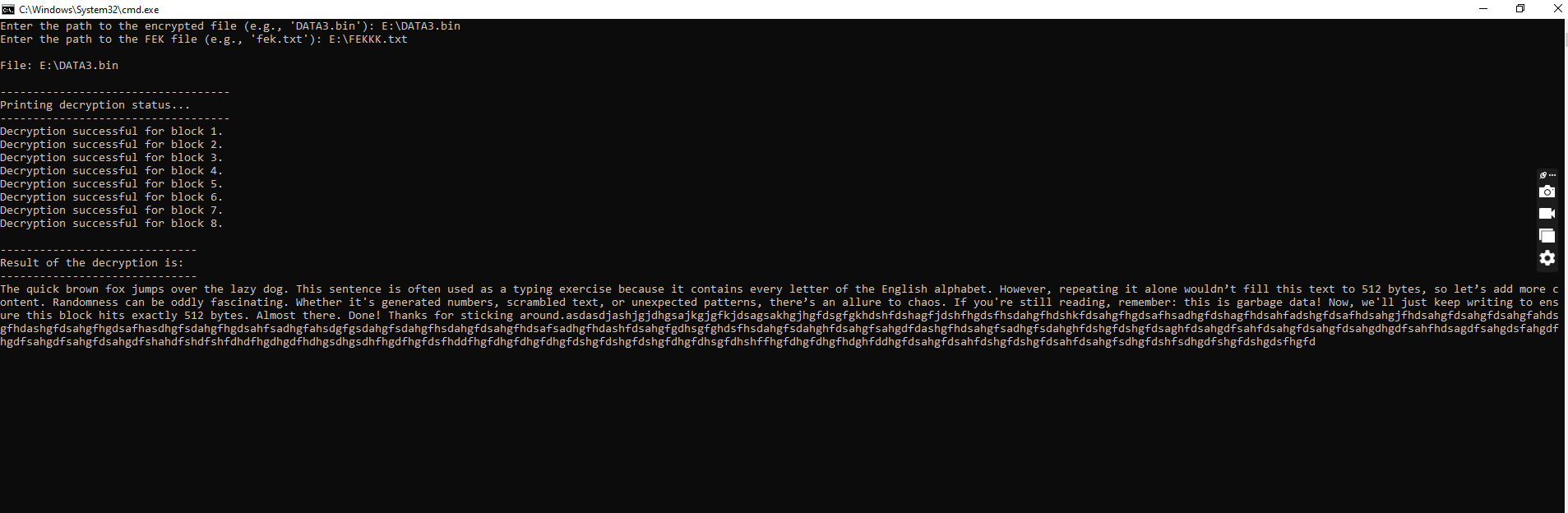


The first 16 bytes of FEK are header bytes and are not needed so they are discarded, the FEK starts after the first 16 bytes. We store the remaining bytes in a Text File to use it for decryption.

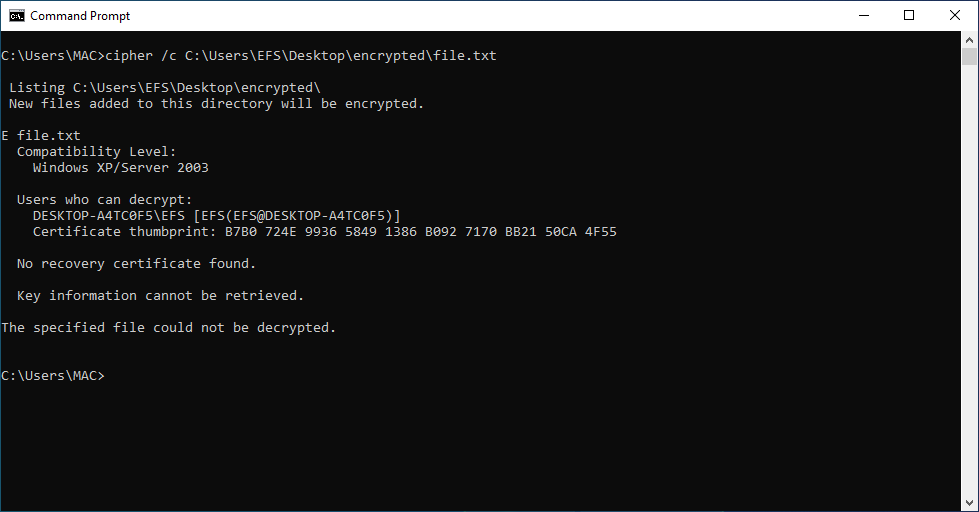


Finally, we give the AES decryption code two inputs, the FEK that we decrypted in previous step, and the encrypted file, it gives us the decrypted text as the output. Make sure to export the entire cluster assigned to the encrypted file to avoid any skips of the data.

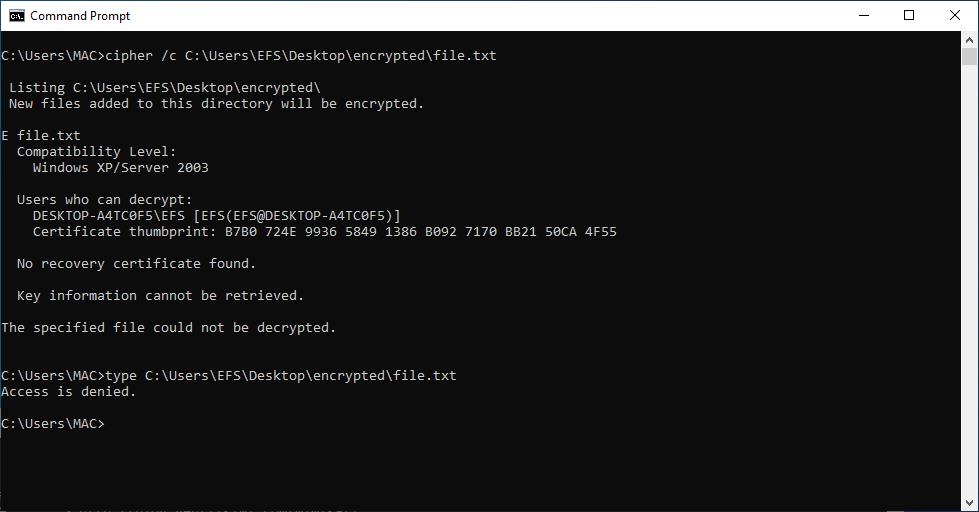
And perfect, it gives us the decrypted output.



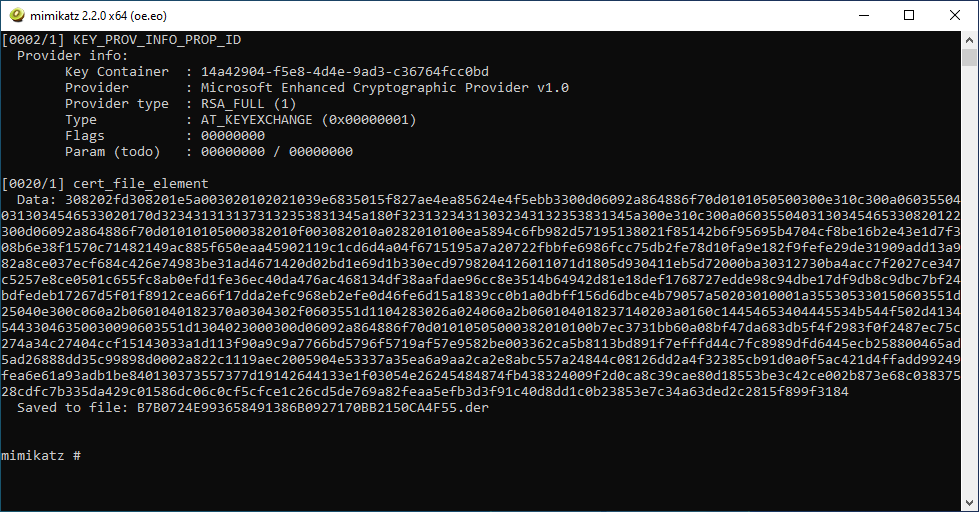
**Part B:**



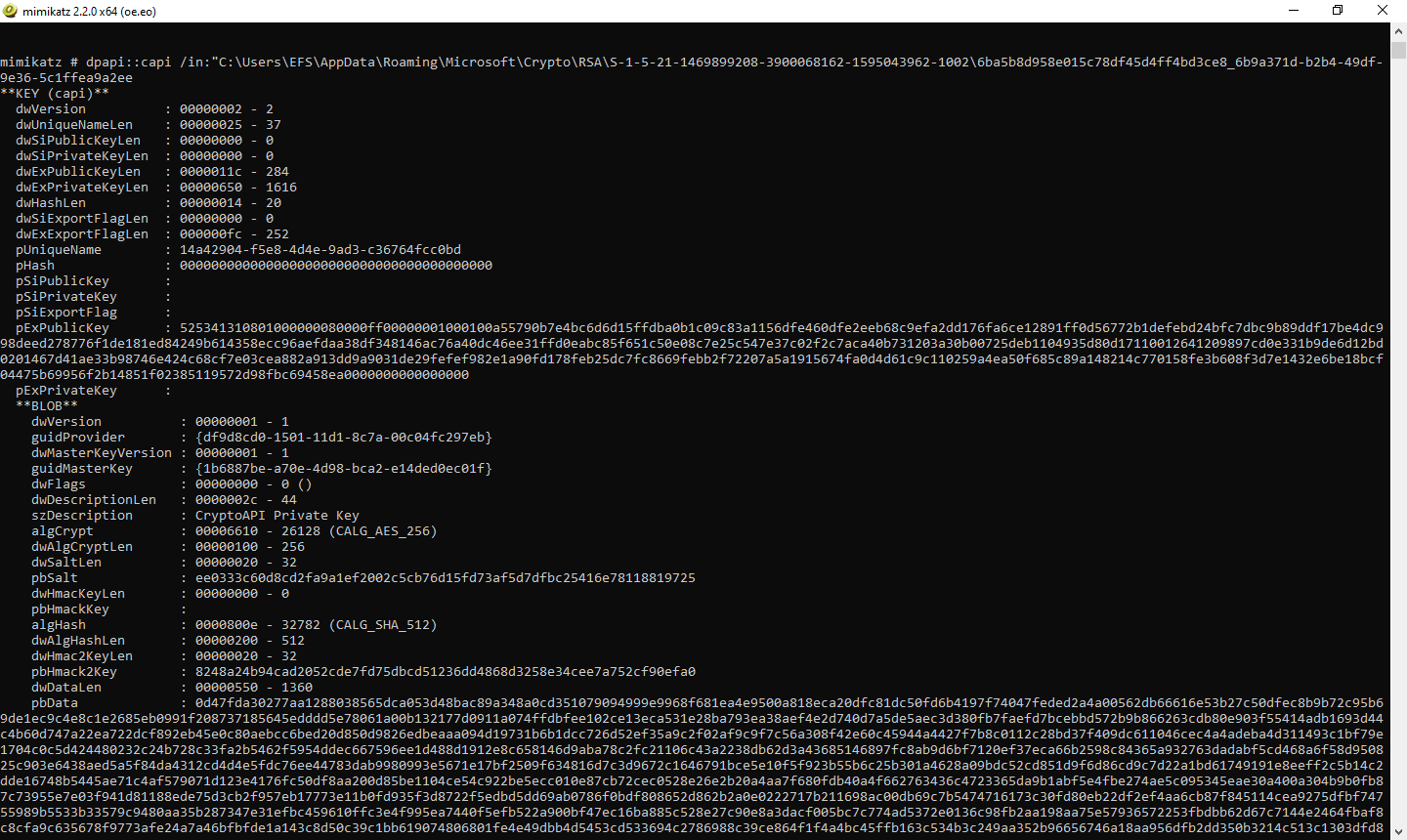
First of all, certificate thumbprint of encrypted file is generated using the **cipher /c** command, which will be used in next steps.



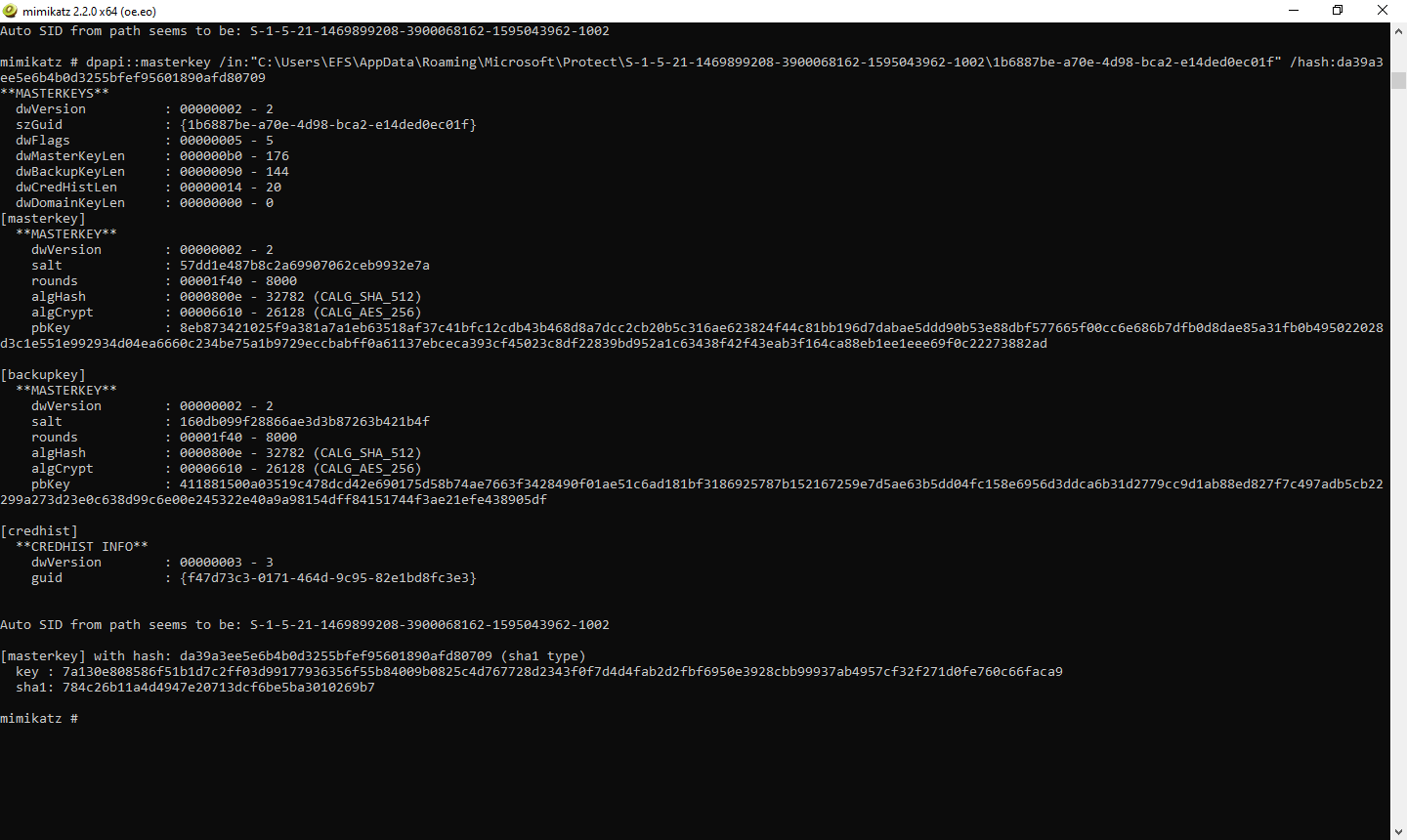
When it is being accessed using the type command, it shows the Access Denied which means that file is encrypted and we can’t access it without key.



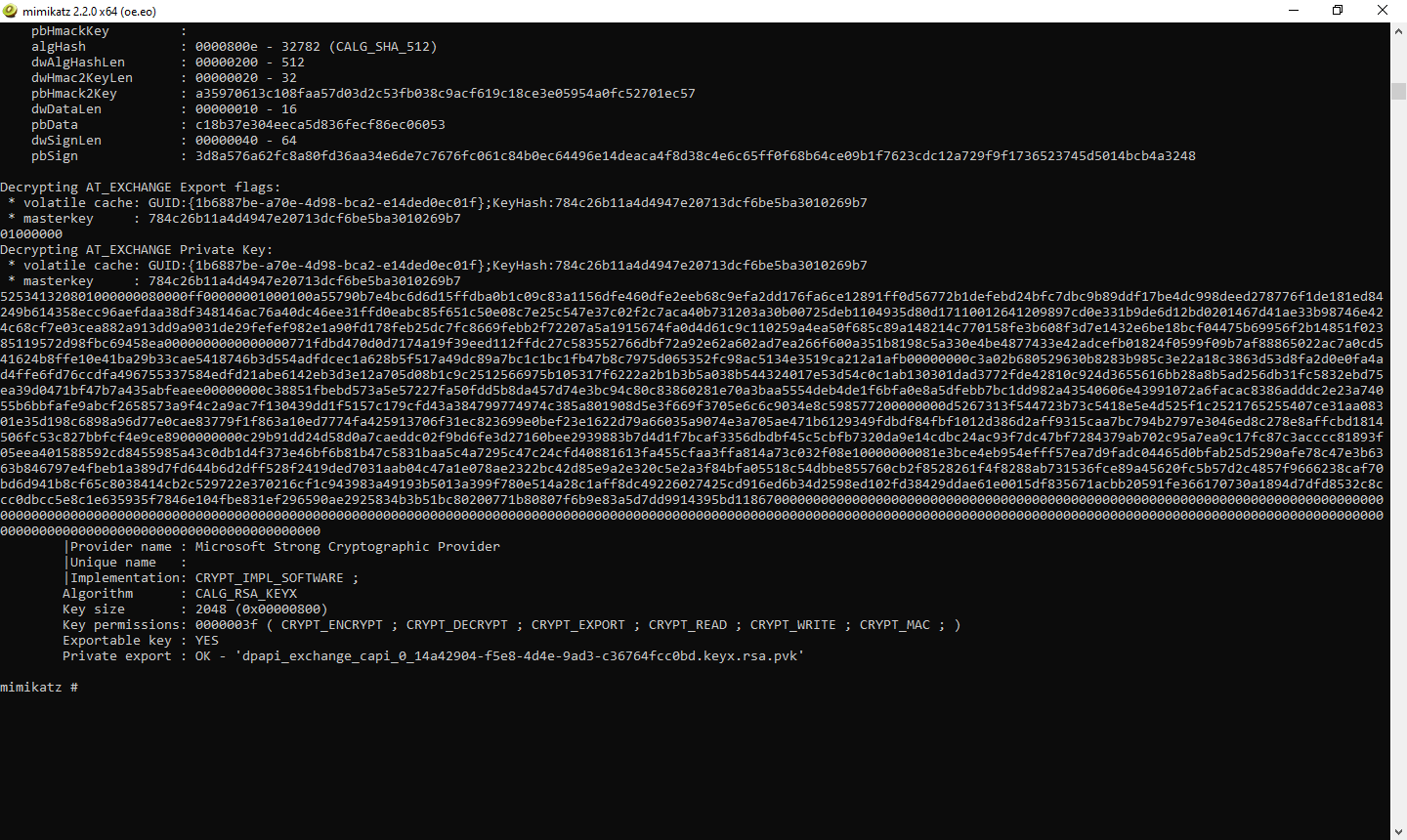
Certificate is being generated using the **crypto::system** command of mimikatz. It stored the certificate and its **public** key in the **B53C6DE283C00203587A03DD3D0BF66E16969A55.der** file, and the private is visible inside the **Key Container**.



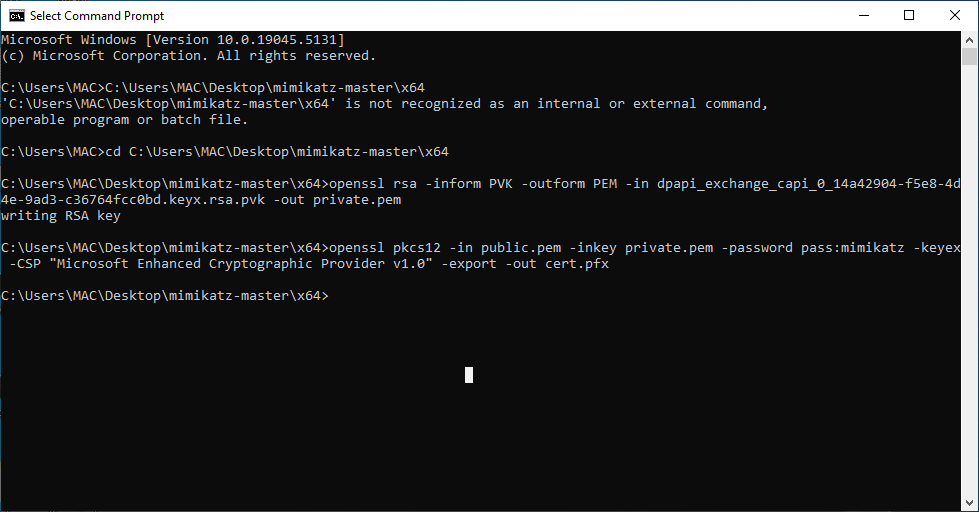
Now using the **dpapi::capi** command we have known that private key is being encrypted with the **guidMasterKey** {**1b6887be-a70e-4d98-bca2-e14ded0ec01f**}, and this master key is itself encrypted. We have to decrypt it first to use it further.



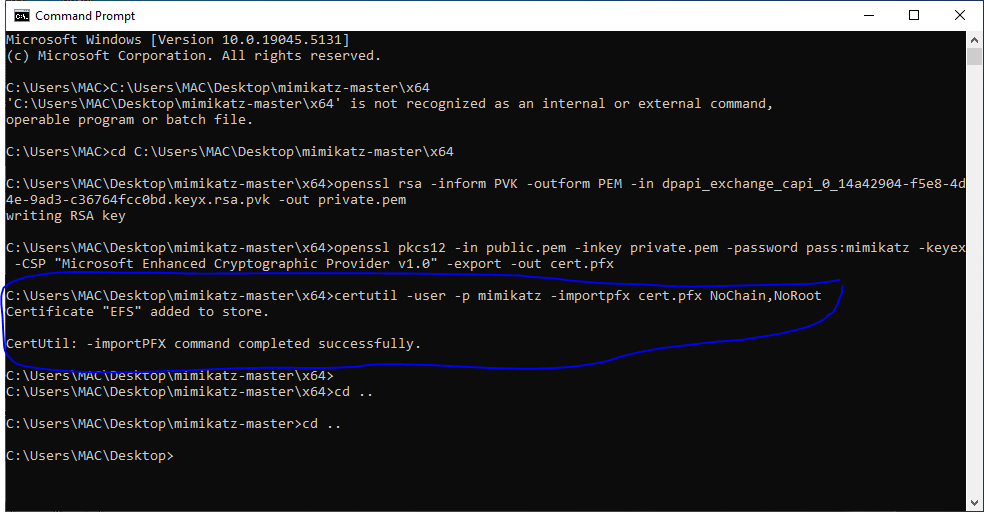
Now we have to decrypt the master key. We can do it by using **dpapi::masterkey** command and giving it the password of the current user. As there is no password on my current profile so we will be using the hash of empty string {**da39a3ee5e6b4b0d3255bfef95601890afd80709**}. It will give us the decrypted master key and its hash.



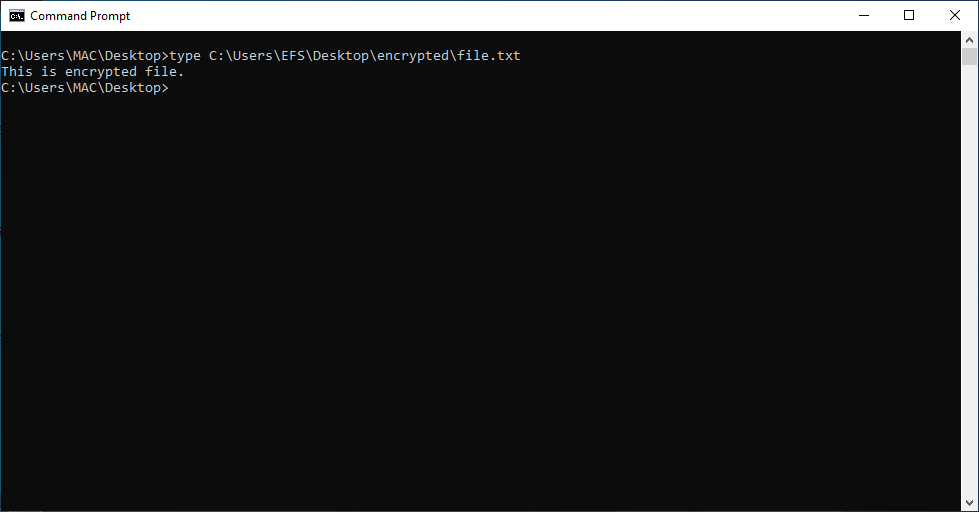
As we have master key and already have the private key, we can decrypt it using the decrypted master key. We have done it using the **dpapi::capi** command. It will export the private key (**pvk** format).



Now that we have the decrypted private key and certificate, we can use **openssl** to extract the key information from it. It will write RSA key to the file which will be used for the decryption of the file. The **pkcs12** command will store the information in the **cert.pfx** file which will be used to decrypt the file.



**CertUtil** imports the certificate and private key from the **cert.pfx** file into the current user's certificate store using the password "mimikatz" to decrypt the **.pfx** file.



Now that we have the decrypted private key and the import is successful, we can access the decrypted data as shown.

# Conclusion:

This project successfully demonstrates the process of extracting and using the **File Encryption Key (FEK)** from encrypted files on a Windows system protected by **Encrypting File System (EFS)**. We explored two main parts: first, identifying encrypted files and extracting the FEK using digital forensic tools like **Sleuth Kit** and **AES decryption**, and secondly, extracting and decrypting the private key from an EFS certificate using **Mimikatz**, **OpenSSL**, and **CertUtil**.

By following this approach, we were able to decrypt the file's content, showcasing the practical application of cryptographic key extraction and decryption processes in the context of digital forensics.

# References:

🡺 <https://tinyapps.org/docs/decrypt-efs-without-cert-backup.html>

🡺 <https://github.com/gentilkiwi/mimikatz/wiki/howto-%7E-decrypt-EFS-files>

🡺Chatgpt