Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету )
Кафедра кібербезпеки
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: "Методи шифрування даних в операційних системах. Windows та Linux"

Виконав: студент (ка)

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Чоп О.Г.
підпис (прізвище та ініціали)

Керівник Марценюк В.П.
підпис (прізвище та ініціали)

Нормоконтроль Лобур Т.Б.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2023

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Чопу Олексію Григоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи шифрування даних в операційних системах Windows та Linux

Керівник роботи Марценюк Василь Петрович, д.т.н., професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 03 » 04 2023 року № 4/7-349

2. Термін подання студентом завершеної роботи 12.06.2023

3. Вихідні дані до роботи Вимоги до операційних систем Windows та Linux

4. Зміст роботи (перелік питань, які потрібно розробити)
Огляд загальних принципів шифрування, включаючи базові поняття та класифікацію методів шифрування.
Розглянути загальні принципи шифрування даних операційних системах Windows та Linux
Розглянути методи шифрування даних в операційній системі Windows
Проаналізовати засіб шифрування дисків BitLocker та шифровану файлову систему EFS
Розглянути методи шифрування даних в операційній системі Linux
Проаналізовати програмне забезпечення LUKS та шифровану файлову систему EncFS
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, задачі. Методи шифрування даних. Симетричне шифрування. Асиметричне шифрування даних. Хеш функції. Шифрування даних в операційній системі Windows. Приклад шифрування даних за допомогою BitLocker. Шифрування даних за допомогою EFS в операційній системі Windows. Приклад шифрування у Windows даних за допомогою EFS. Шифрування даних в операційній системі Linux. Приклад шифрування диска за допомогою LUKS в RedHat Linux. Шифрування даних в операційній системі Linux за допомого EncFS Приклад шифрування даних в операційній системі Linux за допомогою EncFS. Переваги та недоліки BitLocker шифрування даних в Windows. Переваги та недоліки Encrypting File System шифрування даних в Windows. Переваги та недоліки LUKS шифрування даних в Linux. Переваги та недоліки EncFS шифрування даних в Linux. Крос-платформність методів шифрування. Приклад розшифрування тестового USB флеш диску з міткою диску "Encrypted_F" в RedHat Linux. Висновки

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Безпека життєдіяльності, основи хорони праці | Пилипець М.І., проф. кафедри МТ | | |

7. Дата видачі завдання 20.01.2023 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 20.01 – 23.01 | Виконано |
| 2. | Підбір джерел аналізу роботи і налаштування поштових серверів | 25.01 – 29.01 | Виконано |
| 3. | Опрацювання джерел в галузі дослідження | 30.02 – 10.02 | Виконано |
| 4. | Огляд методів шифрування даних | 20.02 – 05.03 | Виконано |
| 5. | Огляд можливостей шифрування даних в операційних системи Windows та Linux. | 07.03-20.03 | Виконано |
| 6. | Оформлення розділу «Теоретичні основи електронних поштових серверів» | 14.02 – 17.03 | Виконано |
| 7. | Оформлення розділу «Огляд методів шифрування даних» | 06.03 – 14.03 | Виконано |
| 8. | Оформлення розділу «Операційні системи windows та Linux. можливості шифрування даних» | 21.03 – 20.04 | Виконано |
| 9. | Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці» | 25.04 – 15.05 | Виконано |
| 10. | Оформлення кваліфікаційної роботи | 20.05 – 05.06 | Виконано |
| 11. | Нормоконтроль | 06.06 – 12.06 | Виконано |
| 12. | Перевірка на плагіат | 10.06 – 11.06 | Виконано |
| 13. | Попередній захист кваліфікаційної роботи | 14.06 – 15.06 | Виконано |
| 14. | Захист кваліфікаційної роботи | 23.06.2023 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Чоп О.Г.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценюк В.П.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Методи шифрування даних в операційних системах Windows та Linux// Кваліфікаційна робота ОР «Бакалавр» // Чоп Олексій Григорович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-41 // Тернопіль, 2023 // С. 83, рис. – 60, табл. – - , кресл. 27 , додат. – -.

Ключові слова: ШИФРУВАННЯ, ОПЕРАЦІЙНІ СИСТЕМИ, WINDOWS, LINUX, BITLOCKER, LUKS.

Дана кваліфікаційна робота присвячена аналізу методів шифрування даних в операційних системах Windows та Linux. Дослідження проведене в даній роботі допоможе зрозуміти основні принципи роботи з шифруванням даних в операційних системах, оцінити їх ефективність, переваги та недоліки.

Дану роботу можна використовувати в навчальному процесі для ілюстрації методів шифрування даних в операційних системах Windows та Linux.

В першому розділі проведено огляд загальних принципів шифрування, включаючи базові поняття та класифікацію методів шифрування.

В другому розділі розглянуто загальні принципи шифрування даних в операційній системі Windows. Показано методику шифрування файлів, шифрування дискового простору. Проаналізовано методи шифрування за допомогою BitLocker та Encrypting File System (EFS) в Windows. Розглянуто принципи роботи цих методів, їх застосування. Також в другому розділі розглянуто методи шифрування даних в операційній системі Linux. Проаналізовано методи шифрування за допомогою LUKS та EncFS. Розглянуто принципи роботи цих методів, їх застосування.

В третій частині проведено порівняльний аналіз різних методів шифрування даних в операційних системах Windows та Linux. Зазначено їх переваги, недоліки, рівень захисту, зручність використання та інші фактори, які можуть вплинути на вибір методу шифрування в певному контексті. Також проаналізовано їх кросплатформність.

# ANNOTATION

Data encryption methods in Windows and Linux operating systems// Thesis of educational level "Bachelor" // Oleksii Chop // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, СБс-41 group // Ternopil, 2023 // P. 83 , fig. 60, table. - __, chair. - 27 , added.__.

Keywords: ENCRYPTION, OPERATING SYSTEMS, WINDOWS, LINUX, BITLOCKER, LUKS.

This qualification work is dedicated to the analysis of data encryption methods in Windows and Linux operating systems. The research conducted in this work will help understand the basic principles of working with data encryption in operating systems, evaluate their effectiveness, advantages, and disadvantages.

This work can be used in the educational process to illustrate data encryption methods in Windows and Linux operating systems.

The first chapter provides an overview of general encryption principles, including basic concepts and classification of encryption methods.

The second chapter discusses the general principles of data encryption in the Windows operating system. It presents the methods of file encryption and disk space encryption. The encryption methods using BitLocker and Encrypting File System (EFS) in Windows are analyzed. The principles of operation and their applications are considered. The second chapter also examines the methods of data encryption in the Linux operating system. The encryption methods using LUKS and EncFS are analyzed. The principles of operation and their applications are discussed.

The third part presents a comparative analysis of different methods of data encryption in Windows and Linux operating systems. Their advantages, disadvantages, level of protection, ease of use, and other factors that can influence the choice of encryption method in a specific context are indicated. Their cross-platform compatibility is also analyzed.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| ECB | — | Electronic Codebook |
| CBC | — | Cipher Block Chaining |
| CTR | — | Counter |
| XTS | — | XEX-based Tweaked Codebook Mode with Cipher Text Stealing |
| AES | — | Advanced Encryption Standard |
| DES | — | Data Encryption Standard |
| 3DES | — | Triple Data Encryption Standard |
| ГПП | — | Генератор псевдовипадкової послідовності |
| WEP | — | Wired Equivalent Privacy |
| NaCl | — | Networking and Cryptography library |
| WCCP | — | Web Cache Communication Protocol |
| TLS | — | Transport Layer Security |
| PKI | — | Public Key Infrastructure |
| RSA | — | Rivest Shamir Adleman |
| ECC | — | Elliptic Curve Cryptography |
| SHA-256 | — | Secure Hash Algorithm 256-bit |
| SHA-3 | — | Secure Hash Algorithm 3 |
| MD5 | — | Message Digest Algorithm 5 |
| SHA-1 | — | Secure Hash Algorithm 1 |
| RIPEMD-160 | — | RACE Integrity Primitives Evaluation Message Digest 160-bit |
| TPM | — | Trusted Platform Module |
| UEFI | — | Unified Extensible Firmware Interface |
| EFS | — | Encrypting File System |
| LUKS | — | Linux Unified Key Setup |
| EncFS | — | Encrypted File System |

# ВСТУП

Захист конфіденційності та цілісності даних є однією з найважливіших задач в сучасному інформаційному суспільстві. З розвитком технологій та зростанням кількості цифрових даних, потреба у надійних методах шифрування для захисту інформації в операційних системах стає все більш актуальною. Операційні системи Windows та Linux є одними з найпопулярніших операційних систем, використовуваних в багатьох сферах життя, включаючи бізнес, науку, владу та особисте використання. Тому розуміння методів шифрування даних в цих операційних системах є важливим аспектом для забезпечення безпеки і конфіденційності даних.

Шифрування це процес перетворення інформації, яка є в відкритому вигляді, в зашифровану з використанням криптографічного алгоритму. Цей процес може забезпечити конфіденційність даних, тобто забезпечити захист від несанкціонованого доступу до інформації, таким чином дозволяючи лише авторизованим користувачам читати дані. Шифрування та хешуванням можуть забезпечити не тільки конфіденційність даних а і їх цілісність, тобто забезпечити захист від внесення несанкціонованих змін до даних під час передачі або збереження.

Дана кваліфікаційна робота присвячена аналізу методів шифрування даних в операційних системах Windows та Linux. Дослідження проведене в даній роботі допоможе зрозуміти основні принципи роботи з шифруванням даних в операційних системах, оцінити їх ефективність, переваги та недоліки.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Провести огляд загальних принципів шифрування, включаючи базові поняття та класифікацію методів шифрування.

2) Провести огляд операційних систем Windows та Linux. Розглянути загальні принципи шифрування даних в цих операційних системах, такі як шифрування файлів, шифрування дискового простору.

3) Розглянути методи шифрування даних в операційній системі Windows. Проаналізовати засіб шифрування дисків BitLocker та шифровану файлову

систему EFS. Розглянути принципи роботи цих методів, їх застосування, переваги та недоліки.

4) Розглянути методи шифрування даних в операційній системі Linux. Проаналізовати програмне забезпечення LUKS та шифровану файлову систему EncFS. Розглянути принципи роботи цих методів, їх застосування, переваги та недоліки.

5) Провести порівняльний аналіз методів шифрування даних в операційних системах Windows та Linux. Порівняти їх переваги, недоліки, рівень захисту, зручність використання та інші фактори, які можуть вплинути на вибір методу шифрування в певному контексті. Проаналізовати їх кросплатформність.

# 1 ОГЛЯД МЕТОДІВ ШИФРУВАННЯ ДАНИХ.

Шифрування даних — це обчислювальний процес, який кодує відкритий текст (незашифровані, зрозумілі людині дані) у зашифрований текст (зашифровані дані), доступ до якого мають лише авторизовані користувачі з правильним криптографічним ключем. Простіше кажучи, шифрування перетворює читабельні дані в іншу форму, яку можуть декодувати та переглядати лише люди з правильним ключем, і є життєво важливим компонентом цифрової трансформації .

Існує кілька різних методів шифрування, які можуть бути класифіковані за різними ознаками, такими як використання ключів, алгоритми, кількість ключів та інші [1].

## 1.1 Симетричне шифрування

Симетричне шифрування - це метод шифрування, при якому використовується один і той же ключ для шифрування та розшифрування повідомлення [2]. У симетричному шифруванні дані перетворюються за допомогою шифрувального алгоритму з використанням спільного секретного ключа, який повинен бути відомий як відправнику, так і отримувачу. Класичним прикладом симетричного шифрування є шифр Цезаря.

В симетричному шифруванні повідомлення перетворюється в шифротексту за допомогою шифрувального алгоритму і ключа [3]. Отримувач, щоб розшифрувати повідомлення, використовує той самий ключ і шифрувальний алгоритм для відновлення початкового повідомлення з шифротексту. Оскільки в обох випадках використовується один і той самий ключ, його обмін між відправником та отримувачем вимагає певних безпечних механізмів.

Симетричне шифрування має перевагу в швидкості обробки даних, оскільки використовує прості математичні операції. Однак, одна з головних проблем симетричного шифрування полягає у безпечному обміні секретним ключем між відправником і отримувачем. Крім того, в ситуаціях, коли необхідно забезпечити

конфіденційність даних між багатьма сторонами, симетричне шифрування може бути незручним, оскільки потребує використання окремого ключа для кожної пари взаємодіючих сторін.

### 1.1.1 Блочне шифрування

Блочне шифрування - це метод шифрування, при якому повідомлення розбивається на блоки фіксованої довжини, які потім шифруються незалежно один від одного за допомогою шифрувального алгоритму. Кожен блок зазвичай має розмір 64 або 128 біт. Блочне шифрування використовує ключ для виконання перетворення над кожним блоком даних [4].

Режими роботи в блочному шифрі визначають, як вхідні блоки даних обробляються і перетворюються в шифртекст. Основними режимами роботи в блочному шифру є:

Режим роботи ECB.

У цьому режимі кожний блок даних шифрується незалежно від інших блоків. Це означає, що один і той самий блок вхідних даних буде мати однаковий шифртекст. Режим ECB не забезпечує конфіденційності для однакових блоків даних і, в цілому, не рекомендується для застосування у криптографічних системах.

Режим роботи CBC.

У цьому режимі перед шифруванням кожен блок даних комбінується з попереднім блоком шифртексту шляхом використання операції XOR. Це забезпечує розмивання залежностей між блоками і забезпечує випадковість у вихідному шифртексті. Режим CBC є більш безпечним в порівнянні з режимом ECB, але потребує вектора ініціалізації (IV), який повинен бути унікальним для кожного повідомлення.

Режим роботи CTR.

У цьому режимі блоки даних шифруються за допомогою послідовності лічильників, яка залежить від ключа шифрування та номера блоку. Цей режим

дозволяє паралельне шифрування блоків даних і може бути використаний для ефективного шифрування великих обсягів даних.

Режим роботи XTS.

Є режимом роботи в блочному шифрі. Він є одним з режимів, призначених для шифрування данних, особливо в симетричних блочних шифрах, таких як AES.

XTS використовується для шифрування секторних або блочних пристроїв, таких як диски або файли. Він забезпечує конфіденційність і захист. Основна особливість XTS полягає в тому, що він використовує два незалежні ключі для кожного блоку даних - один ключ використовується для шифрування, а інший - для вирішення проблеми різних блоків, що виникає при шифруванні однакових блоків.

XTS використовується, зокрема, для шифрування даних на жорстких дисках, дисках SSD, розподілених файлових системах і т.д. Цей режим забезпечує ефективне і безпечне шифрування блоків даних з урахуванням особливостей таких пристроїв та систем зберігання.

Ці режими роботи в блочному шифру використовуються для забезпечення конфіденційності, цілісності та інших криптографічних властивостей при шифруванні даних. Вибір конкретного режиму залежить від вимог застосування, використовуваного шифру та контексту системи.

Один з головних викликів блочного шифрування - це керування розміром блоку та режимами роботи. Малий розмір блоку може обмежити максимальний розмір повідомлення, яке може бути оброблене, а також зробити шифрування вразливим до атак з відомим текстом. Режими роботи визначають, як оброблюються блоки даних, чи враховується попередній шифротекст, і як здійснюється ініціалізація вектора ініціалізації (IV) для кожного блоку.

Блочне шифрування використовується в багатьох криптографічних протоколах та системах, таких як веб-переглядачі, протоколи безпеки Wi-Fi (WPA2), віртуальні приватні мережі (VPN) та багато інших застосувань, де забезпечення конфіденційності та цілісності даних є важливими аспектами. Блочне шифрування широко використовується в сфері інформаційної безпеки,

включаючи захист даних в електронних комунікаціях, збереження даних та інші сценарії.

Популярні блочні шифри включають AES, DES, 3DES та інші. Ці шифри використовуються для захисту конфіденційності даних, забезпечення цілісності та інших криптографічних операцій.

Шифр AES.

AES - це симетричний блочний шифр, який використовується для захисту конфіденційності даних. Він є одним з найпоширеніших і надійних алгоритмів шифрування, які використовуються в сучасних системах інформаційної безпеки.

Ось деякі характеристики AES:

1) AES шифрує та розшифровує дані блоками фіксованої довжини 128 біт.
2) Розмір ключа 128, 192 або 256 біт. AES підтримує різні розміри ключа, що дозволяє вибрати відповідний рівень безпеки для конкретного застосування.
3) Кількість раундів залежить від розміру ключа. Для 128-бітного ключа використовується 10 раундів, для 192-бітного - 12 раундів, а для 256-бітного - 14 раундів. Кожен раунд включає послідовність операцій, які перетворюють блок даних.
4) AES використовує операції підстановки (Substitution), перестановки (Permutation), зсуву (Shift) та лінійного перетворення (MixColumns) для обробки даних в кожному раунді.

AES є стандартом, який використовується у багатьох криптографічних системах та протоколах, завдяки своїм високим показникам безпеки та ефективності.

Шифр 3DES.

3DES - це блочний шифр, який є розширенням оригінального DES. 3DES використовує повторне застосування DES алгоритму з трьома різними ключами для забезпечення більшої безпеки.

Ось деякі характеристики 3DES:

1) 3DES шифрує та розшифровує дані блоками фіксованої довжини 64 біта, так само як і оригінальний DES;

2) В 3DES використовуються три ключі DES розміром 56 біт кожен, що загалом дає 168 біт. Однак, фактичною ефективною довжиною ключа є 112 біт, оскільки третій ключ використовується для зворотного шифрування;
3) 3DES складається з 3 раундів, в кожному з яких застосовується стандартний DES алгоритм. У результаті 3DES виконується тричі для кожного блоку даних;
4) 3DES використовує операції підстановки (Substitution), перестановки (Permutation) та зсуву (Shift) для обробки даних в кожному раунді DES.

3DES, хоча є відносно повільним порівняно з більш сучасними шифрами, все ще використовується у багатьох системах, де підтримка сумісності з DES є важливою.

Шифр Blowfish.

Blowfish - це симетричний блочний шифр, розроблений Брюсом Шнайером у 1993 році. Він вважається одним з надійних алгоритмів шифрування і має наступні характеристики:

1) Blowfish шифрує та розшифровує дані блоками фіксованої довжини 64 біта.
2) Blowfish дозволяє використовувати ключі довільного розміру, але рекомендовані розміри ключа знаходяться в діапазоні від 32 до 448 біт.
3) За замовчуванням Blowfish виконує 16 раундів шифрування. Кожен раунд включає послідовність операцій, які перетворюють блок даних.
4) Blowfish використовує заміни (Substitution), перестановки (Permutation) та операції XOR для обробки даних у кожному раунді.

Blowfish досі застосовується в деяких системах, але його використання зменшується на користь більш сучасних шифрів, таких як AES, які вважаються більш безпечними та ефективними.

Шифр Serpent.

Serpent є симетричним блочним шифром, розробленим як один з кандидатів на позицію стандарту AES.

Ось деякі характеристики Serpent:

1) Serpent шифрує та розшифровує дані блоками фіксованої довжини 128 біт.
2) Serpent підтримує ключі різного розміру, але найпоширенішими є ключі розміром 128, 192 та 256 біт.
3) Serpent використовує 32 раунди шифрування. Кожен раунд включає послідовність операцій, які перетворюють блок даних.
4) Serpent використовує операції підстановки (Substitution), перестановки (Permutation) та операції XOR для обробки даних у кожному раунді.

Хоча Serpent не отримав позицію AES, він все ще використовується в деяких системах, як альтернатива іншим шифрам і для дослідницьких цілей. Він відомий своєю високою безпекою та стійкістю до атак, але може бути менш ефективним за швидкістю порівняно з іншими шифрами, такими як AES.

Шифр Twofish.

Twofish - це симетричний блочний шифр, розроблений як один з кандидатів на позицію стандарту AES. Ось деякі характеристики Twofish:

1) Twofish шифрує та розшифровує дані блоками фіксованої довжини 128 біт.
2) Twofish підтримує ключі різного розміру, включаючи 128, 192 та 256 біт.
3) Twofish використовує до 16 раундів шифрування, залежно від розміру ключа. Кожен раунд включає послідовність операцій, таких як підстаовка, перестановка та зсуви;
4) Twofish використовує операції підстановки (Substitution), перестановки (Permutation) та операції XOR для обробки даних у кожному раунді.

Хоча Twofish не став стандартом AES, він все ще використовується в деяких системах, особливо тих, де підтримка сумісності з AES не є обов'язковою, або як альтернатива іншим шифрам для забезпечення безпеки даних.

### 1.1.2 Потокове шифрування.

Потокове шифрування - це метод шифрування, при якому дані шифруються (або розшифровуються) поодинокими бітами або байтами з використанням

ключа та генератора псевдовипадкової послідовності [4]. Основна ідея полягає у тому, що кожен біт або байт повідомлення комбінується з відповідним бітом або байтом згенерованої псевдовипадкової послідовності за допомогою операції XOR.

Особливості потокового шифрування:

1) Потокове шифрування зазвичай працює досить швидко, оскільки кожен біт або байт можна шифрувати окремо.
2) ГПП використовується для генерації послідовності псевдовипадкових бітів або байтів, які будуть комбіновані з повідомленням за допомогою XOR.
3) Ключ використовується для ініціалізації генератора псевдовипадкової послідовності. Однак, важливо мати достатньо довгий та криптографічно міцний ключ для забезпечення безпеки.
4) Кожен біт або байт повідомлення шифрується незалежно від інших, що дозволяє забезпечити потокову передачу даних.

Прикладів потокових шифрів:

1) RC4. Є одним з найбільш відомих потокових шифрів. Він використовується, наприклад, у протоколі безпеки бездротових мереж WEP та інших застосуваннях.
2) Salsa20 - це сучасний потоковий шифр, який використовується, наприклад, в криптографічній бібліотеці NaCl та у протоколі шифрування диска VeraCrypt.
3) Grain - це потоковий шифр, розроблений для стандарту шифрування мобільних GSM-з'єднань, він використовується для захисту мережевого трафіку інтернету речей (IoT) та інших застосунків.
4) A5/1 - це потоковий шифр, який використовується у GSM-телефонії для шифрування голосового трафіку.
5) ISAAC - це потоковий шифр, який відрізняється високою безпекою і використовується у різних криптографічних застосуваннях, таких як безпека мереж та генерація випадкових чисел.

6) ChaCha20 є сучасним потоковим шифром, який широко використовується в різних протоколах та програмах. Він використовується, наприклад, у TLS 1.3, Google's QUIC протоколі, і в багатьох інших застосуваннях.
7) HC-128 є потоковим шифром, який отримав третє місце в експертному конкурсі eSTREAM, спрямованому на вибір надійних потокових шифрів. Він використовується, наприклад, в протоколі безпеки WPA3.
8) Grain-128 є одним з переможців конкурсу eSTREAM. Цей потоковий шифр використовується в різних застосуваннях, включаючи безпеку мобільних мереж та стандарт WCCP.
9) Fortuna - це потоковий шифр, розроблений для генерації псевдовипадкових чисел. Він використовується, наприклад, у криптографічних бібліотеках та системах для забезпечення випадкових чисел для криптографічних операцій.
10) Grain-128a - це покращена версія шифру Grain-128 з виправленням певних слабкостей. Він використовується, наприклад, в протоколі безпеки Bluetooth.

Ці приклади дають уявлення про різноманітність потокових шифрів і їх застосування в різних областях криптографії та безпеки.

## 1.2 Асиметричне шифрування

Асиметричне шифрування - це криптографічний метод, який використовує пару ключів для шифрування та розшифрування даних [1]. Пара ключів складається з приватного ключа і публічного ключа. Публічний ключ використовується для шифрування даних, тоді як приватний ключ використовується для розшифрування даних.

У асиметричному шифруванні, як правило, публічний ключ розповсюджується відкрито, тоді як приватний ключ залишається секретним і відомим тільки власнику. Коли користувач бажає надіслати зашифровані дані, він використовує публічний ключ одержувача для шифрування даних. Лише

власник приватного ключа може розшифрувати ці дані за допомогою свого приватного ключа.

Асиметричне шифрування є важливим засобом забезпечення безпеки в Інтернеті і використовується для захисту конфіденційності даних, аутентифікації та цифрового підпису.

Асиметричне шифрування забезпечує більш високий рівень безпеки порівняно з симетричним шифруванням, де використовується один ключ для шифрування та розшифрування [2]. Оскільки приватний ключ залишається секретним, навіть якщо публічний ключ потрапляє в руки зловмисника, він не може розшифрувати дані без приватного ключа.

Основні переваги асиметричного шифрування:

1) Конфіденційність. Дані залишаються конфіденційними, оскільки тільки власник приватного ключа може їх розшифрувати.
2) Аутентифікація. Використання приватного ключа дозволяє перевірити автентичність відправника даних.

Однак асиметричне шифрування має певні недоліки:

1) Вища обчислювальна складність. Операції шифрування та розшифрування за допомогою асиметричних ключів зазвичай вимагають більше обчислювальних ресурсів порівняно з симетричним шифруванням.
2) Обмеження на розмір даних. Асиметричне шифрування має обмеження на розмір даних, які можуть бути зашифровані безпосередньо за допомогою публічного ключа.

Асиметричне шифрування також дозволяє забезпечити безпеку при обміні ключами. Замість передачі симетричного ключа через незахищений канал, можна використовувати асиметричне шифрування для захищеного обміну симетричним ключем. Наприклад, при встановленні безпечного з'єднання TLS між веб-браузером і сервером, асиметричне шифрування використовується для обміну симетричним ключем, який потім використовується для шифрування і розшифрування даних протягом сесії.

Крім того, асиметричне шифрування дозволяє реалізувати системи публічного ключа (PKI). PKI використовує публічні ключі для створення цифрових сертифікатів, які підтверджують автентичність та ідентифікацію власника ключа. Це дозволяє впевнитися в безпеці комунікації з використанням публічних ключів, які можуть бути перевірені сторонами, що спілкуються.

Загалом, асиметричне шифрування є важливим інструментом для безпеки і конфіденційності даних, забезпечуючи надійний метод шифрування, аутентифікації та цифрового підпису. Воно застосовується в різних сферах, включаючи електронну комунікацію, фінансові транзакції, електронну комерцію та інші області, де безпека даних має велике значення.

### 1.2.1 Алгоритм шифрування RSA.

RSA є одним з найпоширеніших алгоритмів асиметричного шифрування. Основні характеристики:

1) RSA використовує математичні операції, пов'язані з факторизацією великих простих чисел, для генерації приватного і публічного ключів. Приватний ключ залишається секретним, а публічний ключ розповсюджується.
2) Публічний ключ використовується для шифрування повідомлень, тоді як приватний ключ використовується для їх розшифрування. RSA базується на математичних принципах, що забезпечують обернену обчислювальну складність.
3) RSA може бути використаний для створення цифрових підписів, які підтверджують автентичність та цілісність даних. Приватний ключ використовується для створення підпису, а публічний ключ використовується для його перевірки.
4) RSA є криптографічною системою з високим рівнем безпеки, якщо використовуються достатньо великі ключі. Зламати RSA шифрування шляхом факторизації числа на прості множники вважається обчислювально складною задачею.

Довжина ключів RSA вимірюється в бітах і визначається кількістю бітів в числових параметрах ключа. Зазвичай використовуються ключі з довжиною 1024, 2048, 3072 або 4096 біт. Довжина ключа впливає на безпеку алгоритму: чим довший ключ, тим важче його зламати шляхом перебору всіх можливих комбінацій. Однак, довші ключі також вимагають більше обчислювальних ресурсів для шифрування та розшифрування даних. Загальноприйнятий мінімальний розмір ключа для безпеки RSA сьогодні становить 2048 бітів.

### 1.2.2 Алгоритм шифрування ECC

ECC - це криптографічна система, що базується на використанні еліптичних кривих.

Система надає високу ступінь криптографічної безпеки з використанням ключів меньшої довжини порівняно з іншими асиметричними шифрами, такими як RSA. Це робить ECC більш ефективним з точки зору обчислювальних ресурсів та пропускної здатності. Також ECC забезпечує високий рівень безпеки шляхом використання складних математичних завдань, пов'язаних з дискретними логарифмами на еліптичних кривих. Зламати ECC шифрування шляхом перебору всіх можливих комбінацій ключів вважається обчислювально складною задачею.

Довжина ключів ECC також вимірюється в бітах і зазвичай виражається як число, наприклад, 256 бітів або 384 біти. Довжина ключа ECC визначається параметрами еліптичної кривої, яка використовується в алгоритмі. Зазвичай використовуються ключі з довжиною від 128 до 521 бітів.

Одна з переваг ECC полягає в тому, що в порівнянні з іншими асиметричними криптографічними системами, такими як RSA, для досягнення того ж рівня безпеки можна використовувати значно меншу довжину ключа ECC. Наприклад, ECC ключ довжиною 256 бітів відповідає безпековому рівню, який вимагає більшого ключа RSA, наприклад, 2048 бітів.

Це робить ECC особливо ефективним для застосувань з обмеженими обчислювальними ресурсами, такими як мобільні пристрої та вбудовані системи, де швидкодія та енергоефективність є важливими чинниками.

Проте, вибір оптимальної довжини ключа ECC повинен бути здійснений з урахуванням конкретних вимог до безпеки та рекомендацій криптографічних стандартів. Загальноприйняті рекомендації включають використання ключів ECC довжиною 256 бітів для достатньої безпеки.

### 1.2.3 Цифрові сертифікати

Цифрові сертифікати використовують асиметричне шифрування. Основна ідея полягає в тому, що сертифікати використовують пари криптографічних ключів - приватний ключ і відповідний йому публічний ключ.

Приватний ключ залишається в секреті у власника сертифіката, тоді як публічний ключ розповсюджується в цифровому сертифікаті. Коли сторона хоче перевірити автентичність сертифіката, вона використовує публічний ключ з сертифіката для розшифрування цифрового підпису або шифрування даних.

Асиметричне шифрування використовує два ключі - публічний і приватний, де публічний ключ використовується для шифрування даних, а приватний ключ - для розшифрування або підпису даних. У випадку цифрових сертифікатів, публічний ключ використовується для перевірки автентичності сертифіката та шифрування симетричного ключа для безпечного обміну даними. Приватний ключ залишається в секреті власника сертифіката і використовується для створення цифрових підписів та розшифрування отриманих даних.

Використання асиметричного шифрування в цифрових сертифікатах забезпечує безпеку та автентичність у електронних комунікаціях, дозволяючи перевірити автентичність сертифіката та обмінятися зашифрованими даними.

Етапи створення сертифікату:

1) Генерація ключів. Спочатку генерується пара криптографічних ключів - приватний ключ та відповідний йому публічний ключ. Приватний ключ

зберігається в секреті, а публічний ключ стає доступним для розповсюдження.

2) Заява та підпис. Власник ключа подає заяву на видачу сертифіката до довіреної сторони, яка називається центром сертифікації (Certificate Authority, CA). Заява містить інформацію про власника, публічний ключ та інші реквізити. Заява підписується приватним ключем власника для підтвердження її автентичності.
3) Видача сертифіката. Після перевірки заяви та підпису CA видає цифровий сертифікат, що містить відомості про власника, публічний ключ, період дії сертифіката та підпис CA. Сертифікат зберігається у цифровій формі та може бути доступний для перевірки автентичності.

## 1.3 Хеш-функції

Хеш-функція є криптографічною функцією, яка приймає вхідні дані будь-якого розміру і обчислює хеш-код фіксованого розміру.

Кожному унікальному вхідному повідомленню відповідає єдиний вихідний хеш-код. Хеш-функції мають велику розсіювальну здатність, що означає, що навіть невеликі зміни вхідних даних призводять до великих змін в хеш-коді. Це робить хеш-функції чутливими до навіть мінімальних змін у вхідних даних.

Обчислення хеш-коду для будь-якого вхідного повідомлення має бути обчислювально ефективним процесом. Величина хеш-коду повинна бути незначною порівняно з розміром вхідних даних. З вихідного хеш-коду неможливо відновити вихідні вхідні дані. Це надає хеш-функціям властивість незворотності.

Хеш-функції використовуються для перевірки цілісності даних, тобто впевненості, що дані не були змінені після обчислення хеш-коду. Також хеш-функції використовуються для зберігання хеш-значень паролів замість зберігання самого пароля. Це підвищує безпеку, оскільки хеш-значення непередбачувані і не дозволяють отримати початковий пароль з хеш-коду.

Хеш-функції використовуються для обчислення хеш-кодів повідомлень, які потім підписуються приватним ключем для створення цифрових підписів. Це дозволяє перевірити автентичність повідомлення за допомогою відкритого ключа. Також хеш-функції використовуються для реалізації хеш-таблиць, що дозволяють швидкий доступ до даних за ключем. Хеш-функція перетворює ключ в індекс хеш-таблиці, де зберігається відповідна інформація.

Хеш-функції використовуються для контролю цілісності файлів. Шляхом обчислення хеш-коду файлу та порівняння його зі збереженим хеш-кодом можна перевірити, чи файл був змінений. Також вони є ключовим елементом в блокчейн технології. Кожен блок даних у блокчейні має свій хеш-код, який залежить від вмісту блоку. Це дозволяє забезпечити недоторканність та незмінність даних у блокчейні. Хеш-функції використовуються для захисту від колізій, коли два різних вхідних повідомлення мають однаковий хеш-код. Сучасні хеш-функції, такі як SHA-256, мають велику довжину хеш-коду, що знижує ймовірність колізій.

Хеш-функції використовуються для пошуку дублікатів великих масивів даних. Обчислення хеш-кодів для даних дозволяє швидко виявити однакові елементи.

Декілька відомих криптографічних хеш-функцій:

1) Алгоритм хешування SHA-256. Це одна з найпоширеніших хеш-функцій. Вона використовується для створення 256-бітових хеш-кодів і має широке застосування у безпеці даних, блокчейні, цифрових підписах та інших областях.
2) Алгоритм хешування SHA-3. Це нова версія хеш-функції SHA, яка була прийнята як стандарт NIST у 2015 році. Вона надає різні розміри хеш-кодів, такі як SHA-3-256, SHA-3-384, SHA-3-512, і має покращену стійкість до атак.
3) Алгоритм хешування MD5. Це старіший хеш-алгоритм, який генерує 128-бітові хеш-коди. Проте, він вважається застарілим у зв'язку зі своєю вразливістю до колізій, тому не рекомендується для криптографічних застосувань.

4) Алгоритм хешування SHA-1. Це інший старіший хеш-алгоритм, який генерує 160-бітові хеш-коди. Він також вважається застарілим і вразливим до колізій, і його використання не рекомендується для криптографічних цілей.
5) Алгоритм хешування RIPEMD-160. Це хеш-функція, яка генерує 160-бітові хеш-коди. Вона використовується, наприклад, у криптовалюті Bitcoin для створення адрес гаманців.

Важливо використовувати належно випробувані та стандартизовані хеш-функції залежно від конкретного застосування та вимог безпеки.

# 2 ОПЕРАЦІЙНІ СИСТЕМИ WINDOWS ТА LINUX. МОЖЛИВОСТІ ШИФРУВАННЯ ДАНИХ

## 2.1 Шифрування даних в операційній системі Windows

Windows є комерційною операційною системою, розробленою компанією Microsoft. Вона широко використовується на персональних комп'ютерах та серверах по всьому світу. Операційна система Windows має вбудовані можливості шифрування, які можуть допомогти захистити дані від несанкціонованого доступу.

### 2.1.1 Шифрування даних за допомогою BitLocker

Огляд можливостей шифрування.

BitLocker є вбудованим методом шифрування в операційній системі Windows, який дозволяє шифрувати весь диск (Full Disk Encryption), забезпечуючи високий рівень захисту для даних, збережених диску [5].

Він шифрує весь диск, включаючи операційну систему, програми та дані. Це забезпечує захист від несанкціонованого доступу до даних, якщо комп'ютер потрапить в руки зловмисників. Також BitLocker використовує апаратні можливості сучасних комп'ютерів, такі як TPM або UEFI, для забезпечення високого рівня захисту ключів шифрування.

BitLocker керує ключами шифрування, включаючи генерацію, зберігання та керування ними. Ключі можуть бути збережені на захищеному обладнанні, такому як TPM, або можуть бути збережені в резервному файлі або на розділі USB-накопичувача. Він може бути легко керований та налаштований з використанням служби каталогу Active Directory, що дозволяє забезпечити централізоване керування шифруванням на рівні організації. Також BitLocker надає можливості відновлення даних в разі втрати ключів шифрування, таких як використання резервних копій ключів, резервного файлу або відновлення ключів з Active Directory. Він може бути інтегрований з іншими заходами безпеки в Windows, такими як Windows Defender Antivirus, що дозволяє виявляти та

блокувати шкідливі програми перед розшифруванням даних. Легкість у використанні та вбудований інтерфейс користувача дозволяє налаштовувати шифрування на різних рівнях - від всього диска до окремих розділів диска.

BitLocker використовує симетричний шифр для шифрування даних на системному диску. Зокрема, він використовує алгоритм шифрування AES для захисту даних. AES є одним з найбільш поширених симетричних алгоритмів шифрування, використовується для шифрування даних у багатьох застосунках та пристроях, включаючи шифрування даних на системних дисках в операційній системі Windows. AES забезпечує високий рівень безпеки та ефективності, що робить його популярним вибором для захисту даних в BitLocker. Зокрема, BitLocker в операційній системі Windows використовує режим шифрування XTS на базі AES, який забезпечує ще більший рівень безпеки шляхом використання твірної операції для поділу даних на блоки та використання унікального допоміжного ключа (tweak) для кожного блоку. Це дозволяє забезпечити високий рівень конфіденційності та захисту даних, які зашифровані з використанням BitLocker в операційній системі Windows.

Крім симетричного шифру AES, BitLocker також використовує асиметричний шифр для захисту ключів шифрування. Зокрема, він використовує асиметричний алгоритм шифрування RSA для захисту головного ключа шифрування, який використовується для розшифрування ключів симетричного шифру AES, які використовуються для розшифрування даних на зашифрованому диску. Цей асиметричний алгоритм дозволяє забезпечити захист ключів шифрування від несанкціонованого доступу та забезпечити можливість відновлення даних в разі втрати або пошкодження головного ключа шифрування.

Крім того, BitLocker також використовує TPM, який є апаратним компонентом комп'ютера, для забезпечення додаткового рівня захисту ключів шифрування. TPM забезпечує зберігання ключів шифрування на апаратному рівні, що унеможливлює їх витягнення або використання ззовні без відповідної авторизації. Це забезпечує захист від атак на апаратному рівні та забезпечує додатковий рівень безпеки для зашифрованих даних.

AES-XTS забезпечує високий рівень безпеки та надійності для захисту даних на дисках, оскільки використовує потужний алгоритм шифрування AES з довжиною ключа 128 або 256 біт, який вважається стійким до різних атак, включаючи брутфорс та криптографічний аналіз.

Застосування AES-XTS в BitLocker дозволяє надійно захищати дані на зашифрованих дисках в операційній системі Windows, забезпечуючи високий рівень безпеки та конфіденційності.

Приклад процедури шифрування.

Проведемо шифрування тестового диску e: з міткою диску “Encrypted_Disk” в Windows 10 Education. Для виконання процесу шифрування та керування ним буде використано графічні програми управління BitLocker та консольну утиліту manage-bde [6].

Утиліта manage-bde - це інструмент командного рядка, який надає доступ до різноманітних функцій, пов'язаних з управлінням зашифрованими томами BitLocker в ОС Windows. Цей інструмент може бути використаний для встановлення налаштувань, зміни паролю або ключа відновлення, розблокування або блокування диску, а також для виконання інших дій, пов'язаних з управлінням томами BitLocker.

Нижче наведені деякі з команд manage-bde, які можуть бути корисними при роботі з зашифрованими томами BitLocker:

1) manage-bde -status - виводить статус зашифрованого диску BitLocker, включаючи ступінь шифрування, наявність ключів, дискретний ідентифікатор та іншу інформацію.
2) manage-bde -lock <DriveLetter> - блокує зашифрований том BitLocker і перериває доступ до даних на ньому.
3) manage-bde -unlock <DriveLetter> -password - розблоковує зашифрований том BitLocker за допомогою пароля.
4) manage-bde -unlock <DriveLetter> -RecoveryKey - розблоковує зашифрований том BitLocker за допомогою ключа відновлення.

5) manage-bde -changepassword <DriveLetter> - змінює поточний пароль на новий.
6) manage-bde -protectors -add <DriveLetter> -RecoveryPassword - додає ключ відновлення для захисту зашифрованого диску BitLocker.

Ці команди можуть бути використані для управління зашифрованими томами BitLocker в ОС Windows і дозволяють змінювати налаштування шифрування, аутентифікації та інші параметри тому для підвищення безпеки зашифрованих даних.

На початковому етапі диск e: не зашифровано. Що можна побачити з виводу команди manage-bde –status (Рис. 2.1).

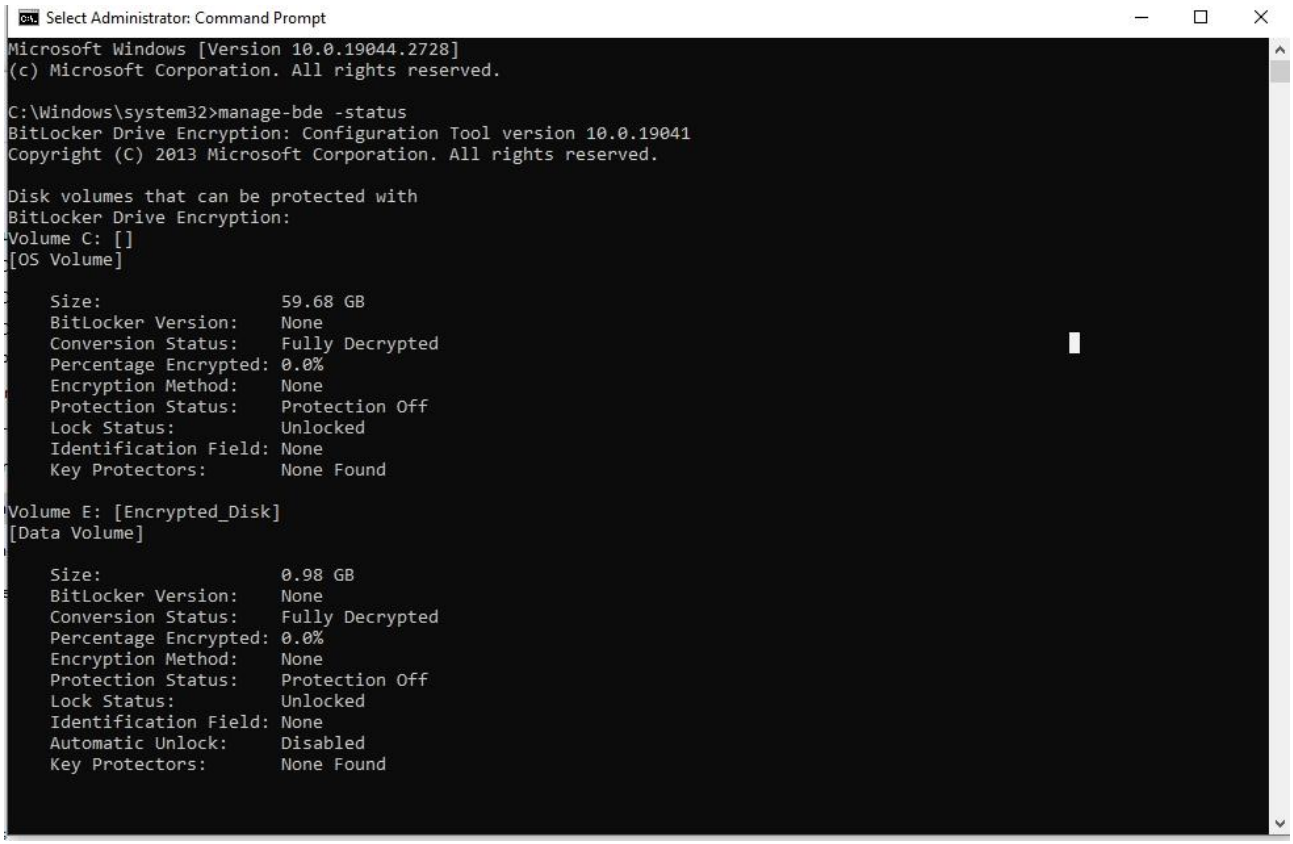


Рисунок 2.1 – Вивід команди manage-bde –status

Після виконання процедури шифрування (Рис. 2.2 – Рис. 2.8) диск e: буде зашифровано за допомогою алгоритму AES-XTS 128.

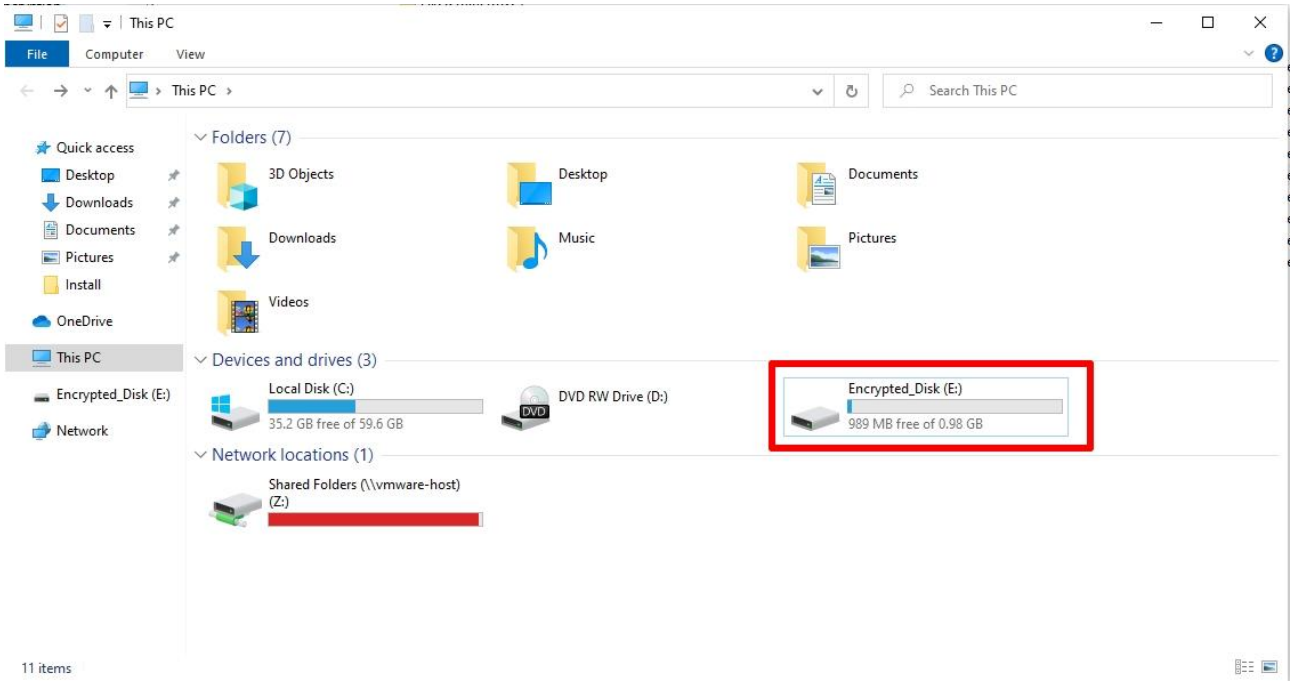


Рисунок 2.2 – Вибір диску для шифрування

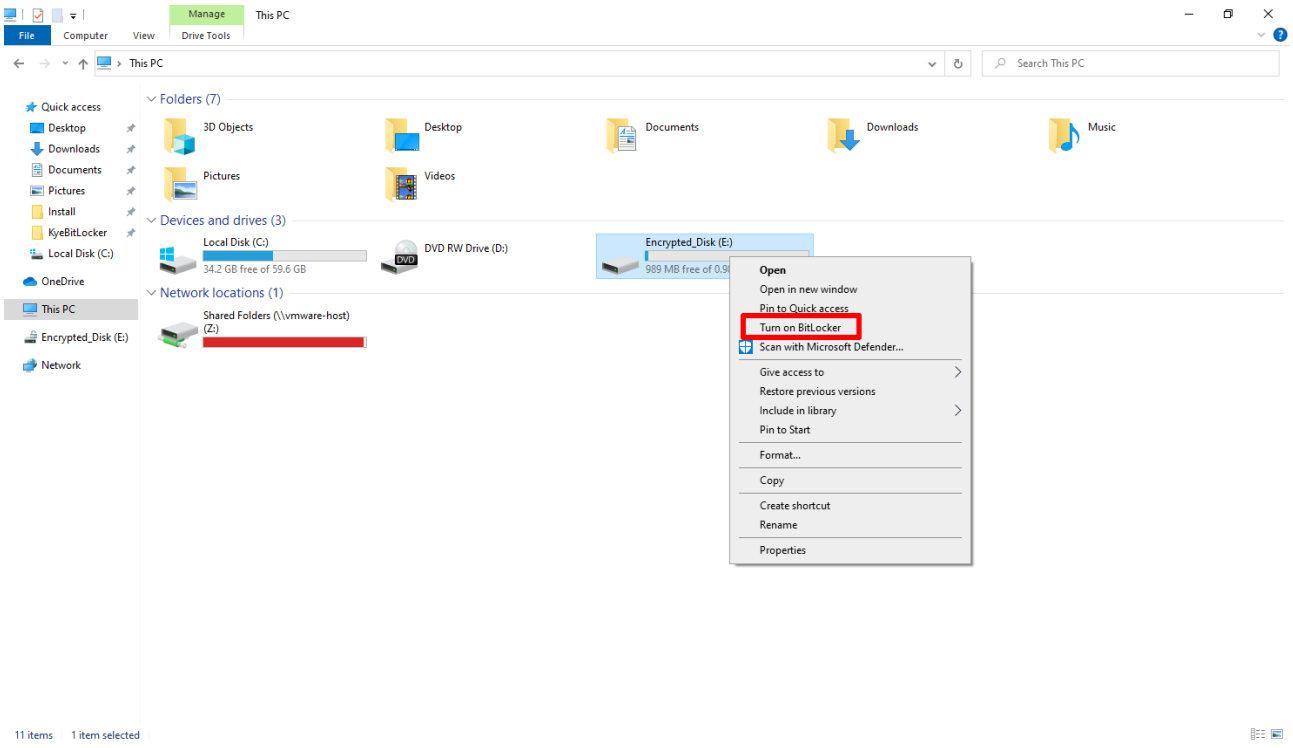


Рисунок 2.3 – Запуск шифрування

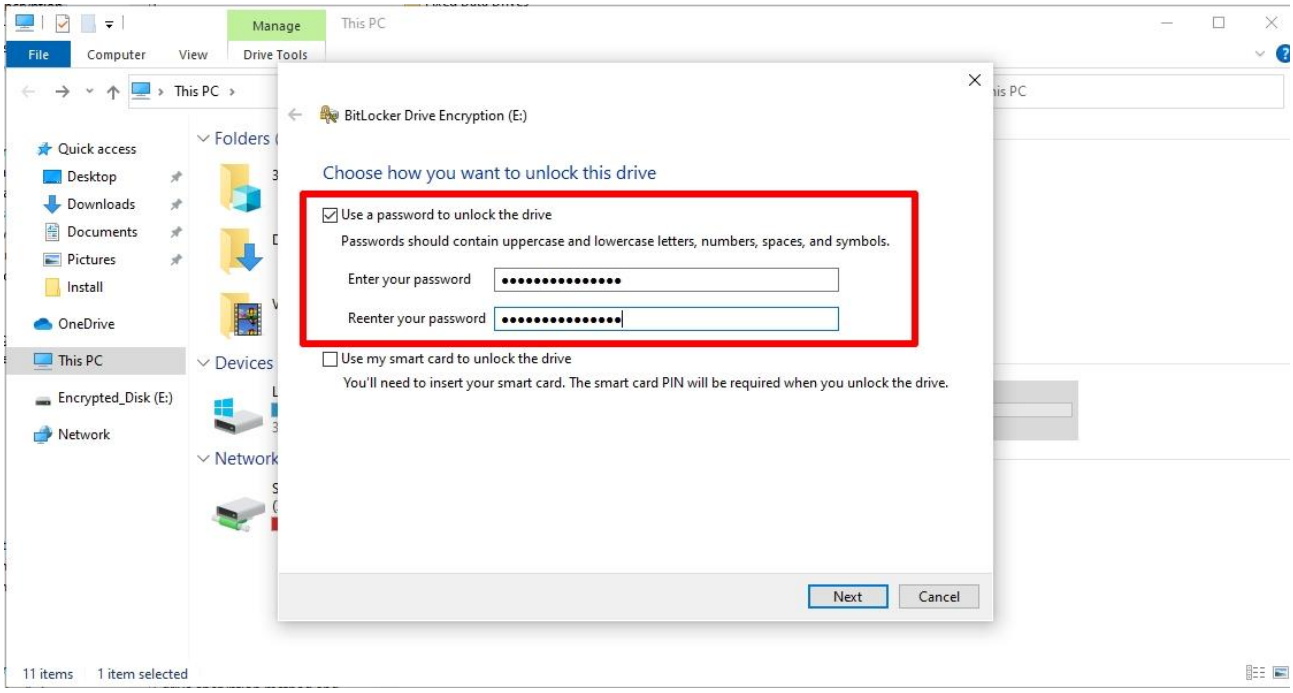


Рисунок 2.4 – Встановлення паролю для розблокування диску

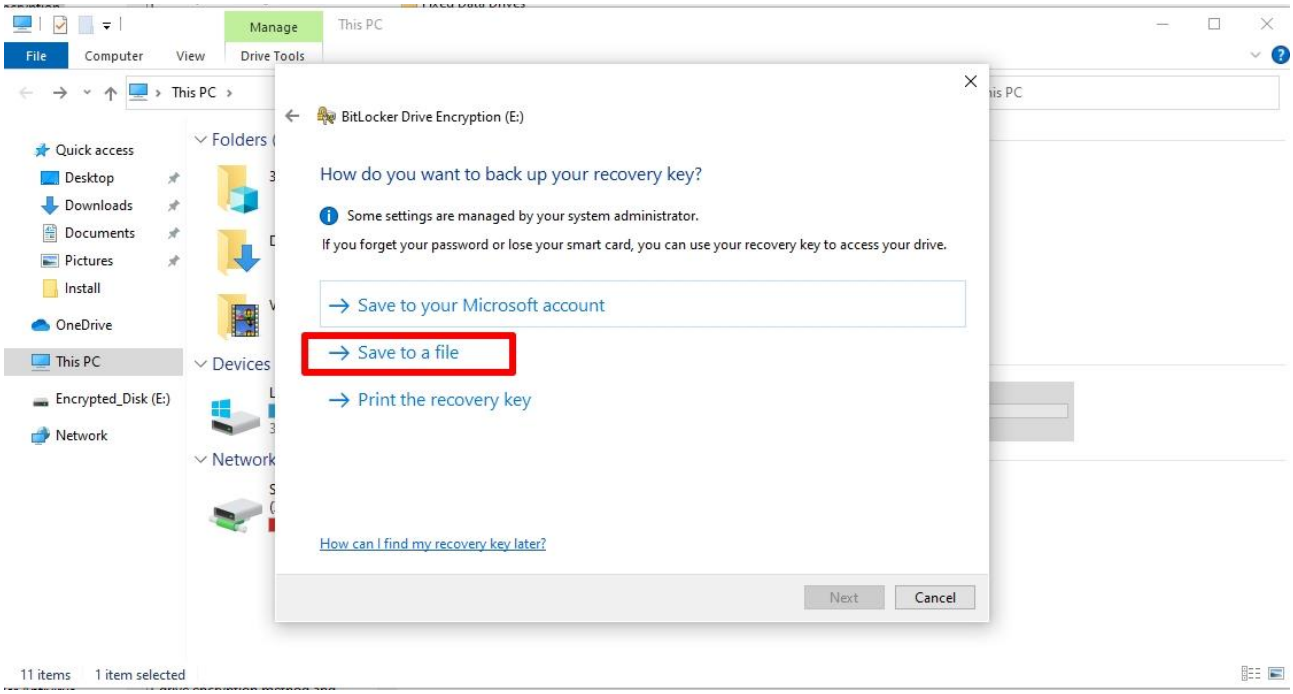


Рисунок 2.5 – Вибір варіанту збереження ключа відновлення

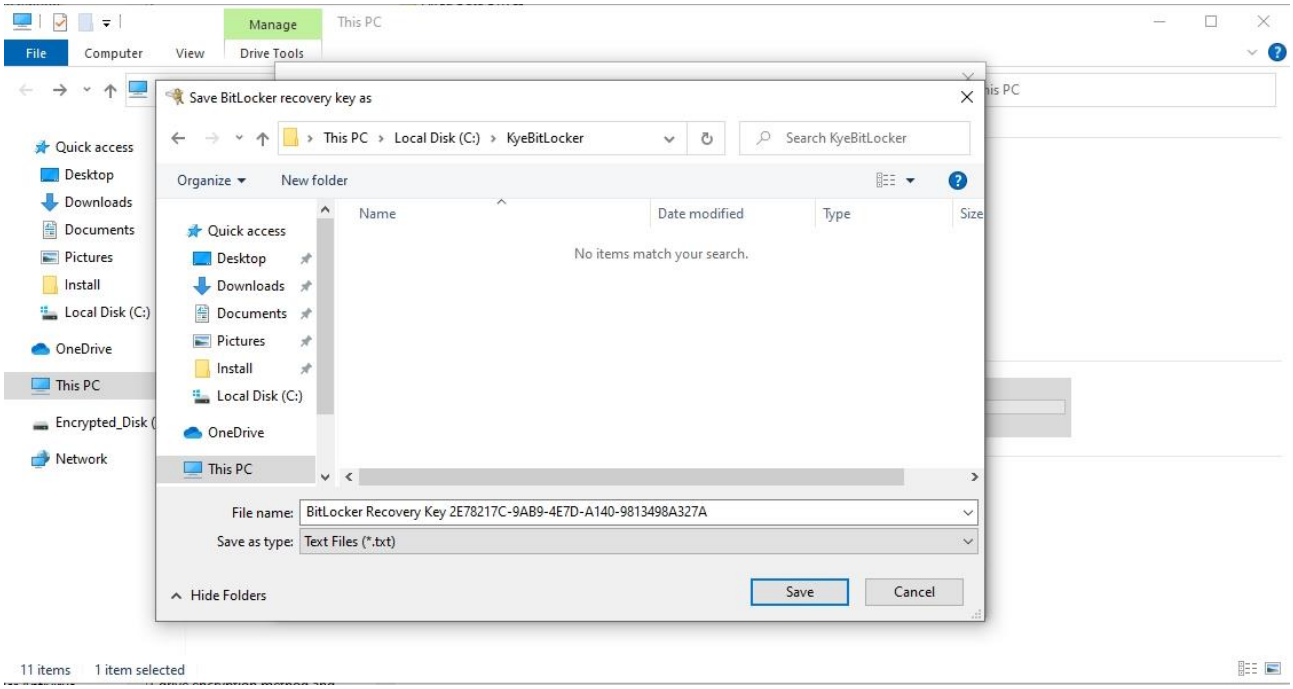


Рисунок 2.6 – Збереження ключа відновлення

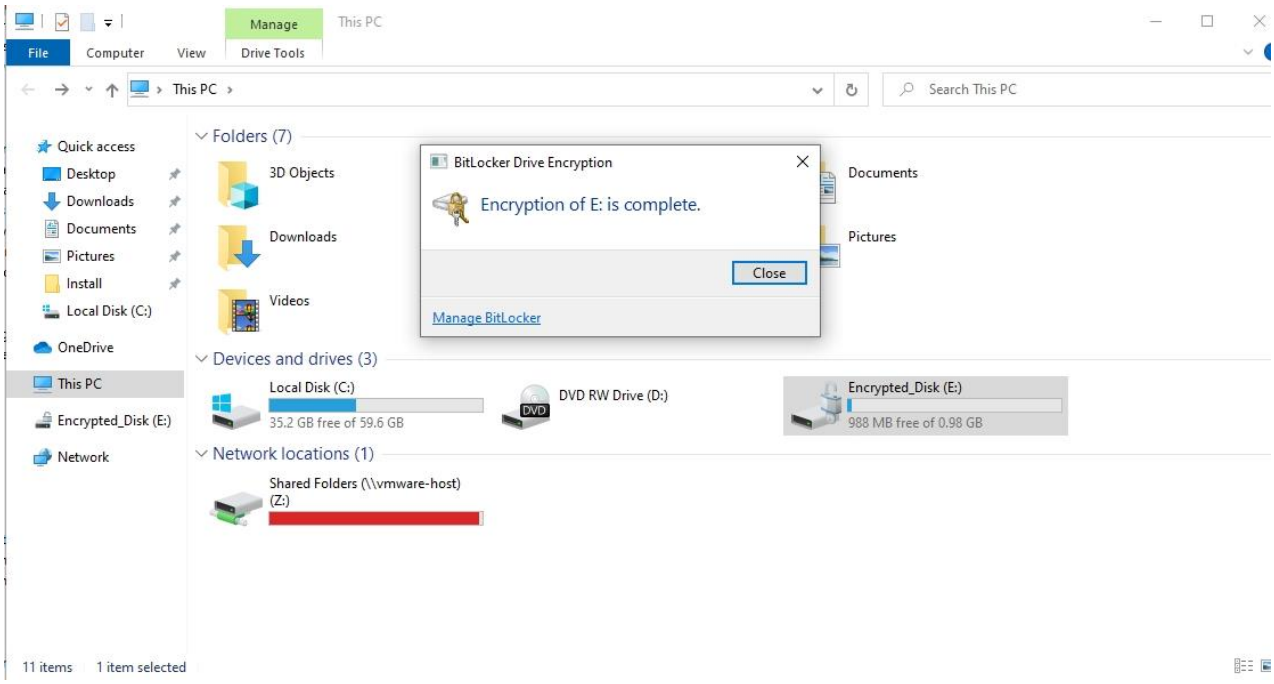


Рисунок 2.7 – Завершення процедури шифрування

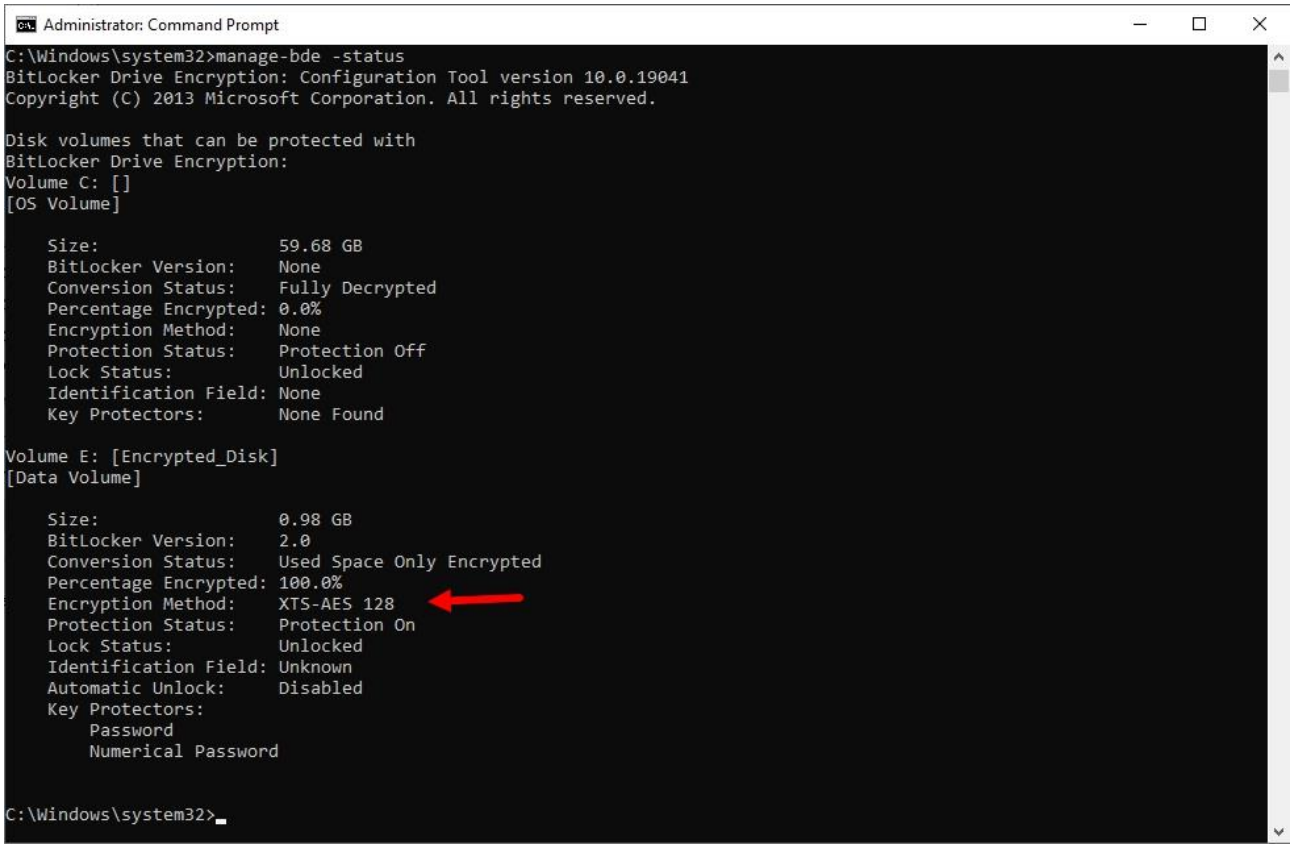


Рисунок 2.8 – Вивід команди manage-bde –status після шифрування

Змінимо алгоритм шифрування BitLocker за допомогою групової політики (gpedit) (Рис 2.9). Встановимо окремо алгоритм шифрування для системного диску, фіксованих дисків та знімних носіїв інформаціях.

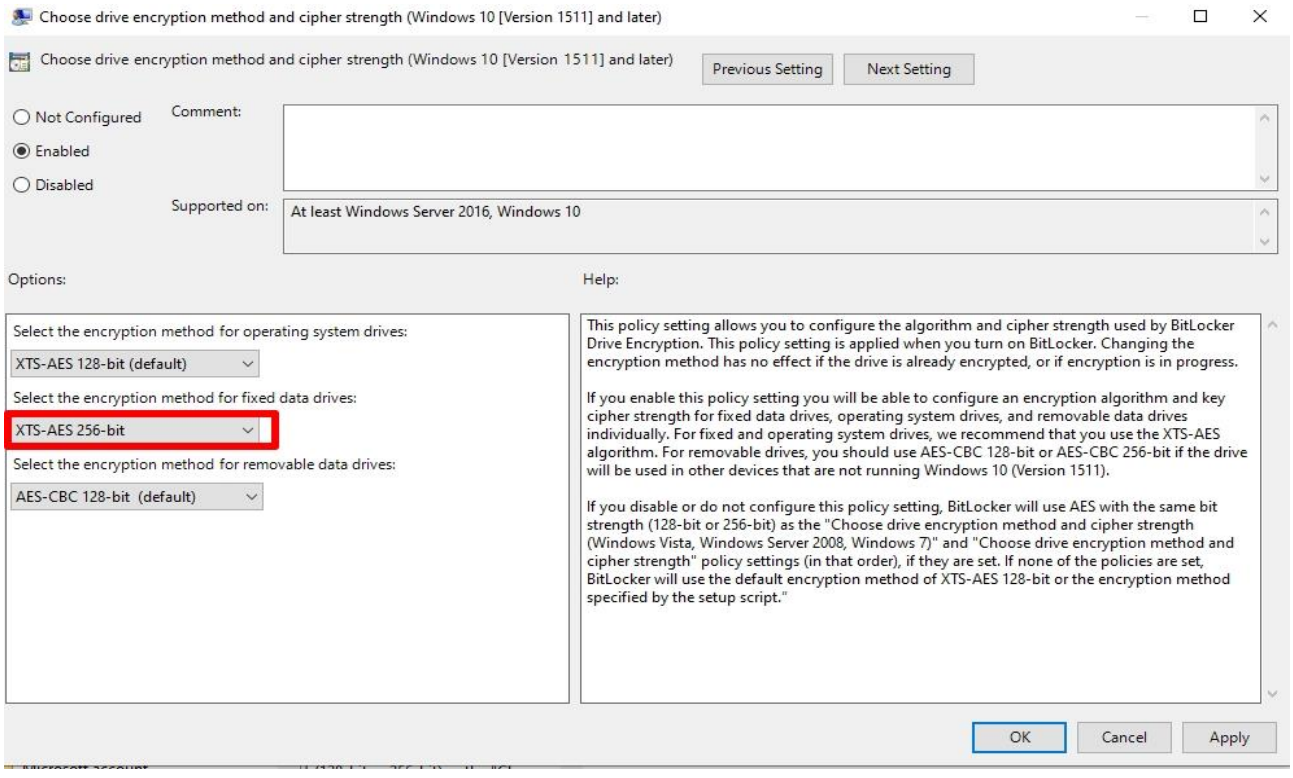


Рисунок 2.9 – Зміна алгоритму шифрування

Для фіксованих дисків і дисків з операційною системою рекомендовано використовувати алгоритм XTS-AES 128-біт або XTS-AES 256-біт. Для знімних дисків слід використовувати AES-CBC 128-біт або AES-CBC 256-біт, якщо диск використовуватиметься в інших пристроях, які не працюють під керуванням Windows 10 (версія 1511) або новіших. Після повторення шифрування з використанням нових налаштувань диск e: буде зашифровано з використанням XTS-AES 256. Що можна побачити з виводу команди manage-bde –status (Рис. 2.10).

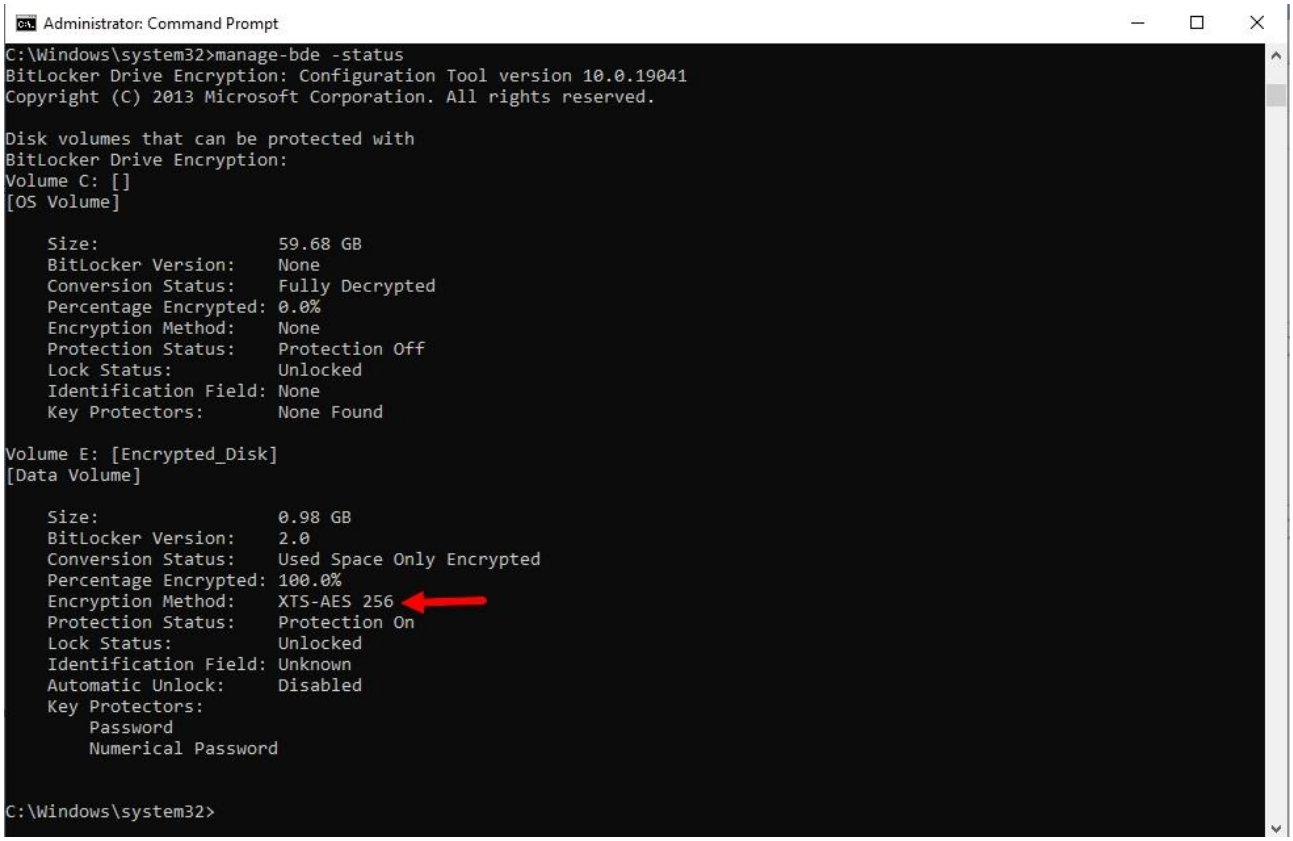


Рисунок 2.10 – Вивід команди manage-bde –status після зміни алгоритму шифрування

Зміна алгоритму шифрування BitLocker може зайняти деякий час, проте вона допоможе підвищити безпеку інформації. Перед зміною алгоритму шифрування слід зробити резервну копію важливих даних, щоб в разі непередбачуваних проблем ви могли відновити свою інформацію. Крім того, перед зміною алгоритму шифрування потрібно впевнитись, що комп'ютер та його операційна система відповідає вимогам нового методу шифрування.

Зміна алгоритму шифрування може бути необхідна, якщо старий метод шифрування став застарілим або більше не є безпечним. У такому випадку зміна алгоритму шифрування може захистити інформацію від зловмисників та хакерів.

Одним з основних моментів шифрування за допомогою BitLocker є ключ відновлення. Ключ відновлення (Recovery Key) - це 48-значний код, що використовується для розблокування зашифрованого диску BitLocker в Windows, якщо втрачено пароль або доступ до іншого методу аутентифікації.

Приклад ключа відновлення BitLocker показано на Рис. 2.11.

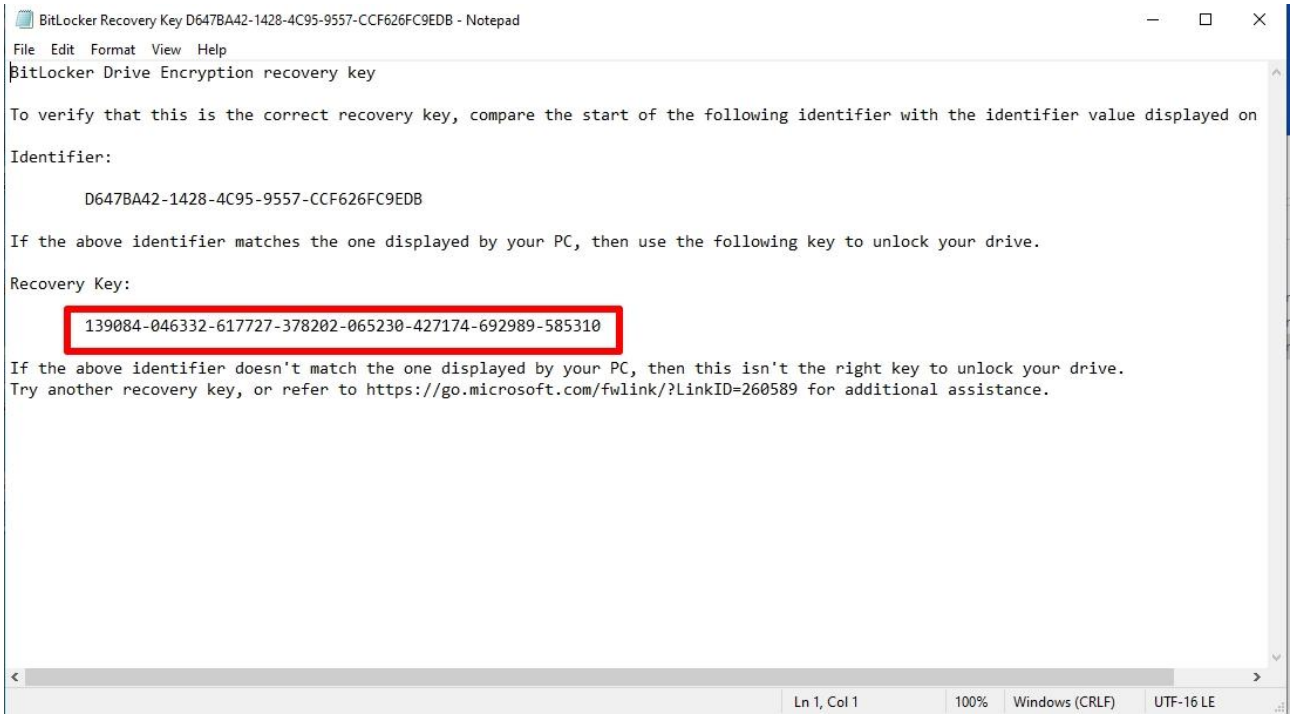


Рисунок 2.11 – Приклад ключа відновлення BitLocker

Ключ відновлення має бути збережений в безпечному місці для використання в майбутньому, якщо виникнуть проблеми з доступом до зашифрованих даних. Також його можна використовувати для розблокування диску на іншому комп'ютері або віртуальній машині, якщо ви перенесли його з одного пристрою на інший.

Важливо зберігати ключ відновлення в безпечному місці, оскільки він дає повний доступ диску зашифрованого BitLocker, а тому може бути використаний для незаконного доступу до вашої інформації. Також, якщо буде втрачено ключ відновлення, може бути важко або неможливо отримати доступ до зашифрованих даних, тому потрібно зберігати його в надійному місці.

На Рисунку 2.12 показано процедуру розблокування диску з використання ключа відновлення.

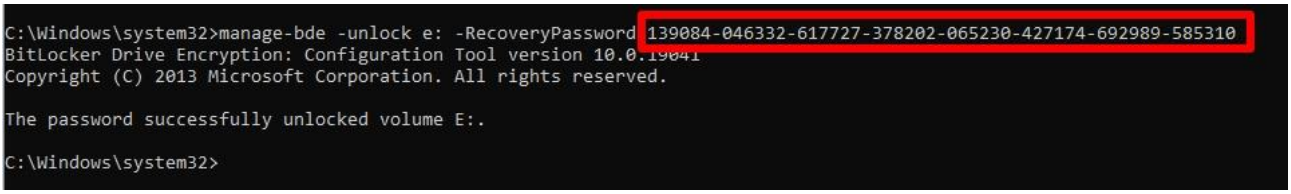


Рисунок 2.12 – Розблокування диску за допомогою manage-bde та ключа відновлення

Шифрування диска за допомогою BitLocker є одним з найбільш надійних методів захисту інформації на ПК з операційною системою Windows. Цей процес є не складним і займає не багато часу.

### 2.1.2 Шифрування даних за допомогою EFS

Огляд можливостей шифрування.

EFS є вбудованим методом шифрування в операційній системі Windows, який дозволяє захищати файли та теки на рівні файлової системи [7].

Цей метод забезпечує шифрування на рівні файлової системи, що означає, що файли та теки, які зашифровані за допомогою EFS, залишаються зашифрованими, навіть якщо вони переміщуються або копіюються на інші розташування на тому ж диску або на іншому диску. EFS використовує публічні та приватні ключі для шифрування та розшифрування файлів. Файл, зашифрований за допомогою EFS, має свій власний унікальний ключ шифрування, який зашифровується за допомогою публічного ключа власника файлу, а розшифровується за допомогою його приватного ключа.

EFS інтегрується з системою управління ключами (Key Management) в операційній системі Windows, такою як служба Active Directory Certificate Services, що дозволяє централізовано керувати ключами шифрування EFS.

EFS дозволяє здійснювати резервне копіювання ключів шифрування, що забезпечує можливість відновлення доступу до зашифрованих файлів в разі втрати або пошкодження ключів. Він інтегрований з інтерфейсом користувача

Windows, що дозволяє користувачам легко шифровувати та розшифровувати файли та теки, використовуючи контекстне меню провідника Windows або властивості файлу. За замовчуванням EFS підтримує алгоритми шифрування AES 256.

EFS може бути інтегрований з процедурою резервного копіювання в операційній системі Windows, що дозволяє забезпечити резервне копіювання зашифрованих файлів, включаючи ключі шифрування, для забезпечення можливості відновлення в разі втрати даних. Він може бути налаштований для використання на рівні домену в мережах Windows, дозволяючи більш централізоване керування правами доступу та ключами шифрування для зашифрованих файлів на рівні домену.

Ці функції роблять EFS потужним інтегрованим методом шифрування в операційній системі Windows, який дозволяє захищати файли та теки на рівні файлової системи. Використання EFS може забезпечити захист від несанкціонованого доступу до конфіденційної інформації, а також від втрати чи крадіжки даних.

Приклад процедури шифрування.

При створенні зашифрованого файла за допомогою EFS, система створює ключ шифрування файлу, який використовується для захисту даних в файлі. Однак, для того, щоб забезпечити безпеку ключа шифрування, він також шифрується за допомогою асиметричного шифрування. EFS використовує алгоритм RSA для захисту ключа шифрування файлів.

Зашифруємо теку TestFolder в якій міститься файл TestFile.txt. Для виконання процесу шифрування та керування ним буде використано графічний інтерфейс управління шифруванням EFS та консольну утиліту cipher.exe.

Утиліта cipher.exe - це інструмент командного рядка в операційній системі Windows, яка надає можливість шифрування та розшифрування файлів та папок, а також виконання інших операцій з безпекою файлів та папок.

Ця утиліта за замовчуванням використовує алгоритм шифрування AES 256. Використовуючи cipher.exe, можна зашифрувати файли та папки в операційній системі Windows зі збереженням їх оригінального розміру та метаданих.

Ця утиліта може бути корисною для захисту конфіденційних даних на комп'ютері, особливо якщо комп'ютер використовується багатьма користувачами або для зберігання даних на зовнішньому носії, наприклад, на флеш-накопичувачі.

Крім того, cipher.exe також дозволяє встановлювати атрибути захисту на файли та папки. Наприклад, можна встановити право доступу до файлу чи папки тільки для певних користувачів або груп користувачів, тим самим забезпечуючи додаткову безпеку файлів та папок.

Основні параметри, які можна використовувати з cipher.exe, включають наступні:

1) /e - шифрування файлу або папки.

2) /d - розшифрування файлу або папки.

3) /s - застосування параметру до всіх файлів та папок у вказаному каталозі та його підкаталогах.

4) /w - очищення вільного простору на диску.

Цей параметр використовується для очищення вільного простору на диску від конфіденційної інформації, що була раніше збережена на ньому.

Вивід команди cipher до початку шифрування (Рис.2.13) показує що тека та її вміст не зашифровані.

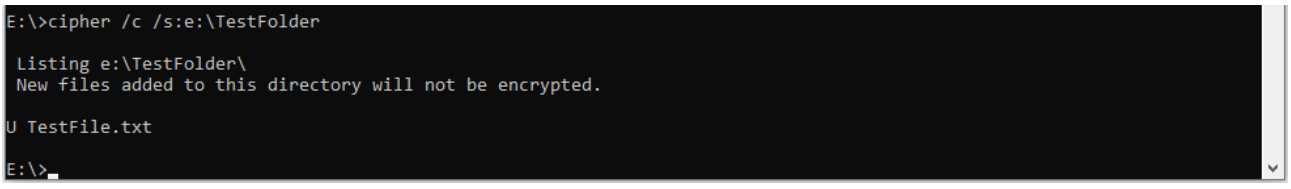


Рисунок 2.13 – Вивід команди cipher до початку шифрування

Після виконання процедури шифрування (Рис. 2.14 – Рис. 2.16) тека TestFolder та її вміст буде зашифровано за допомогою алгоритму AES 256.

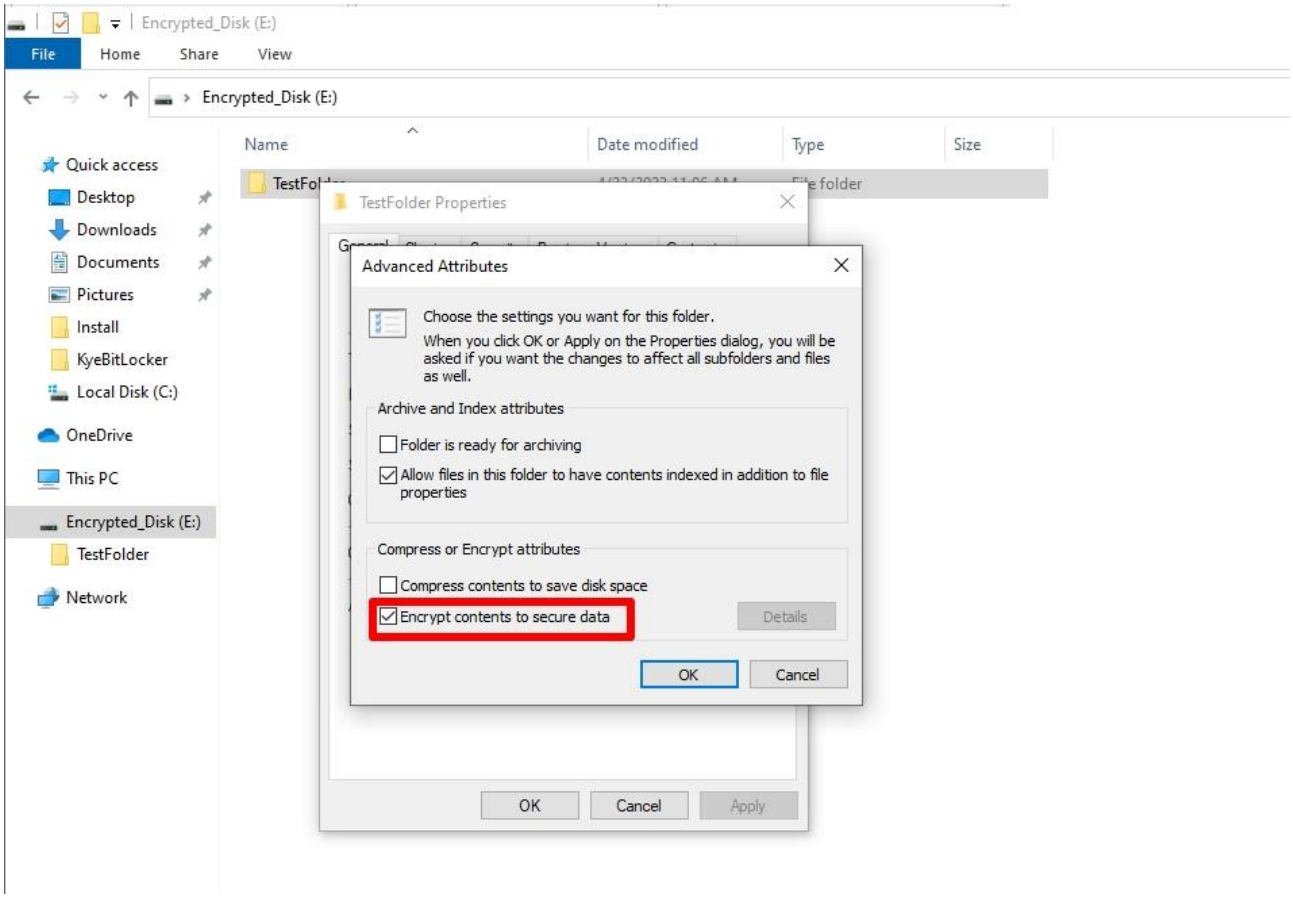


Рисунок 2.14 – Активація шифрування для теки TestFolder

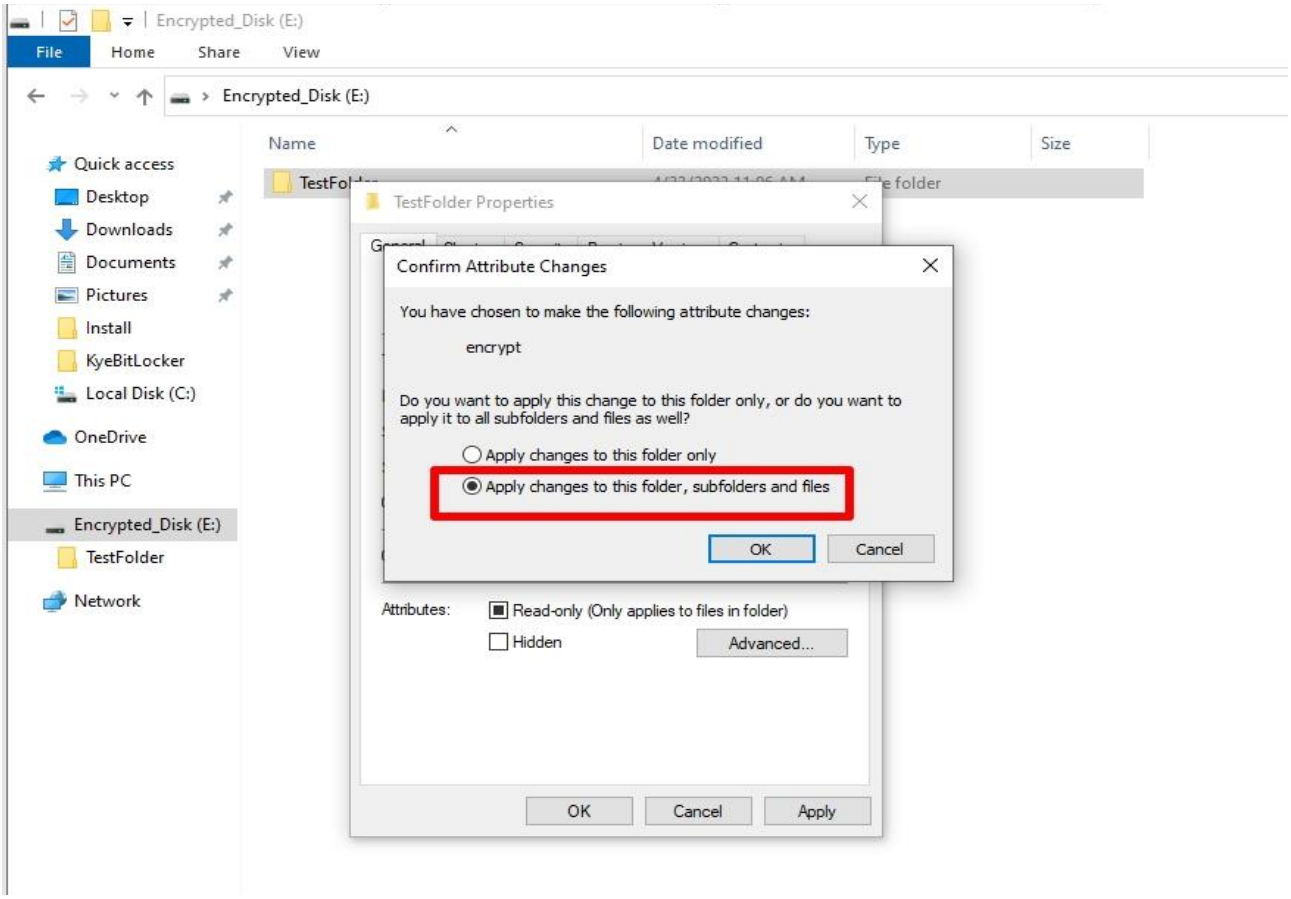


Рисунок 2.15 – Застосування шифрування до теки TestFolder та її вмісту

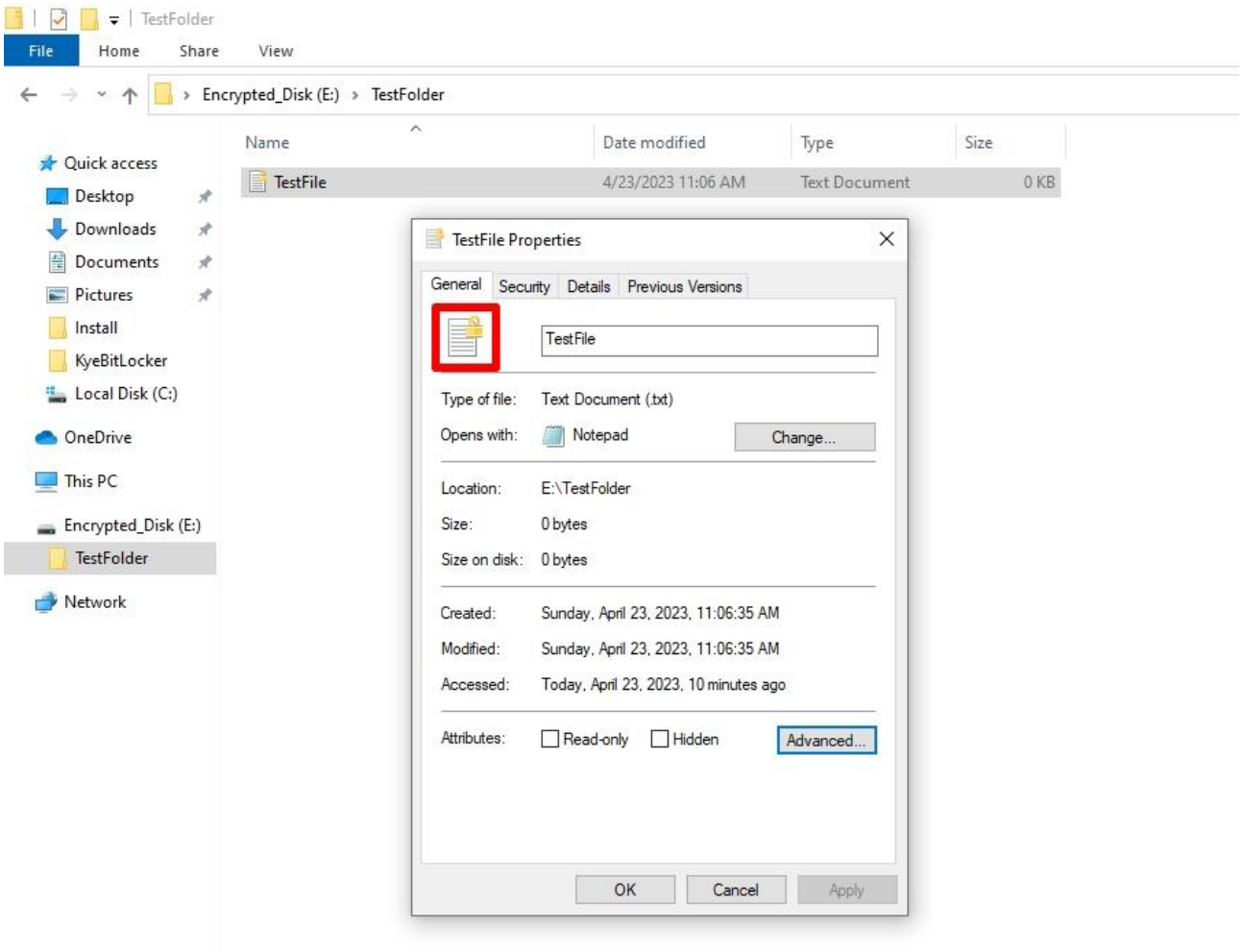


Рисунок 2.16 – Додаткові атрибути файлу після шифрування

EFS використовує симетричне шифрування з ключем, що виводиться з пароля користувача за допомогою алгоритму Microsoft Enhanced Cryptographic Provider. За замовчуванням EFS використовує алгоритм шифрування AES з довжиною ключа 256 біт. Однак, в залежності від версії операційної системи та налаштувань EFS, можуть використовуватись інші алгоритми шифрування, такі як 3DES.

Вивід команди cipher після завершення шифрування (Рис.2.17) показує що тека та її вміст зашифровані з використанням AES 256.

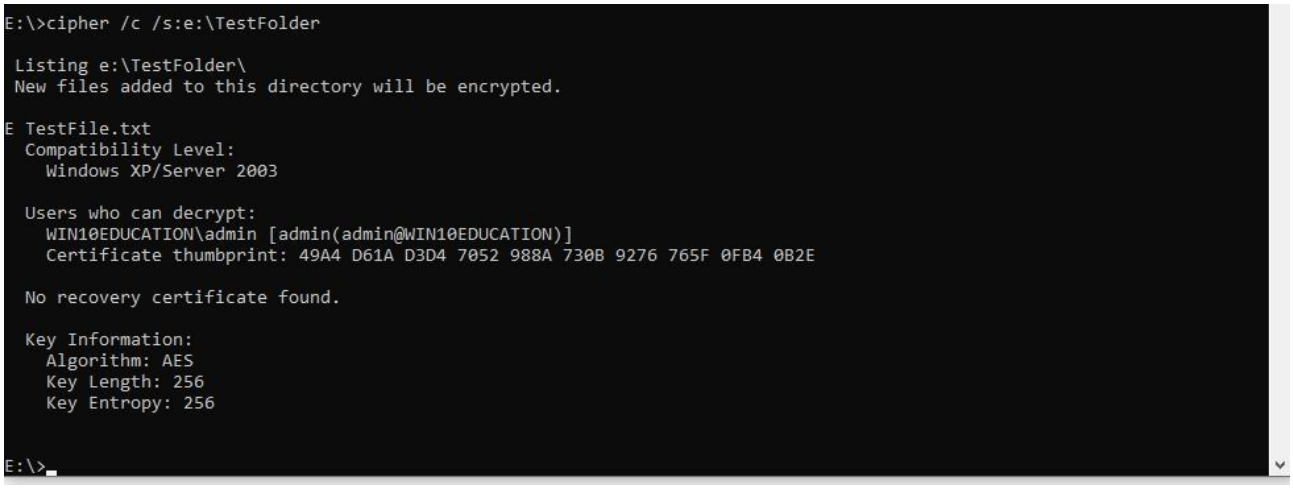


Рисунок 2.17 – Вивід команди cipher після шифрування

Оскільки неможливо відкрити зашифрований файл або папку на іншому комп'ютері, якщо ви не експортуєте свій сертифікат шифрування та не імпортуєте його на іншому комп'ютері, процедура експорту сертифікату є важливим етапом шифрування. Також процедуру експорту потрібно виконувати з міркувань безпеки та запобіганню втраті доступу до зашифрованих файлів.

Після виконання процедури експорту (Рис. 2.18 – Рис. 2.20) тека KyeEFS буде містити сертифікат захищений паролем та зашифрований AES 256.

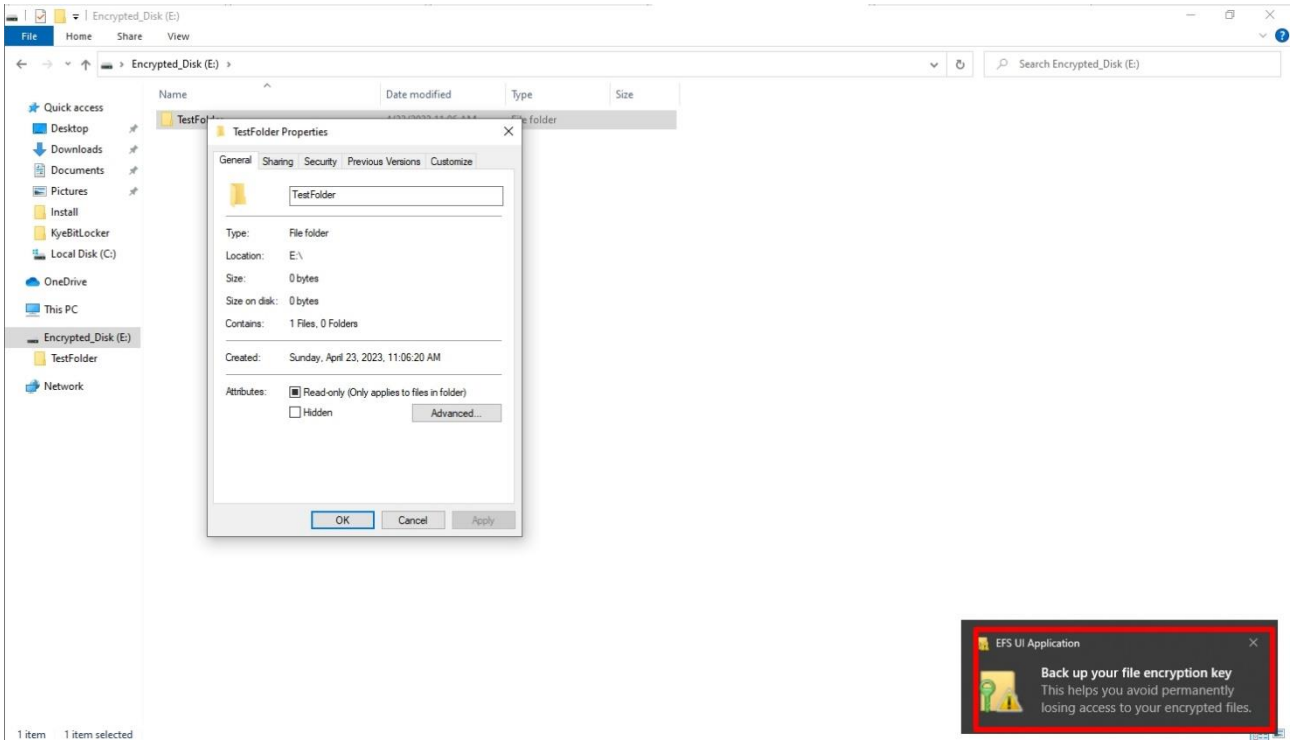


Рисунок 2.18 – Початок процедури експорту сертифікату

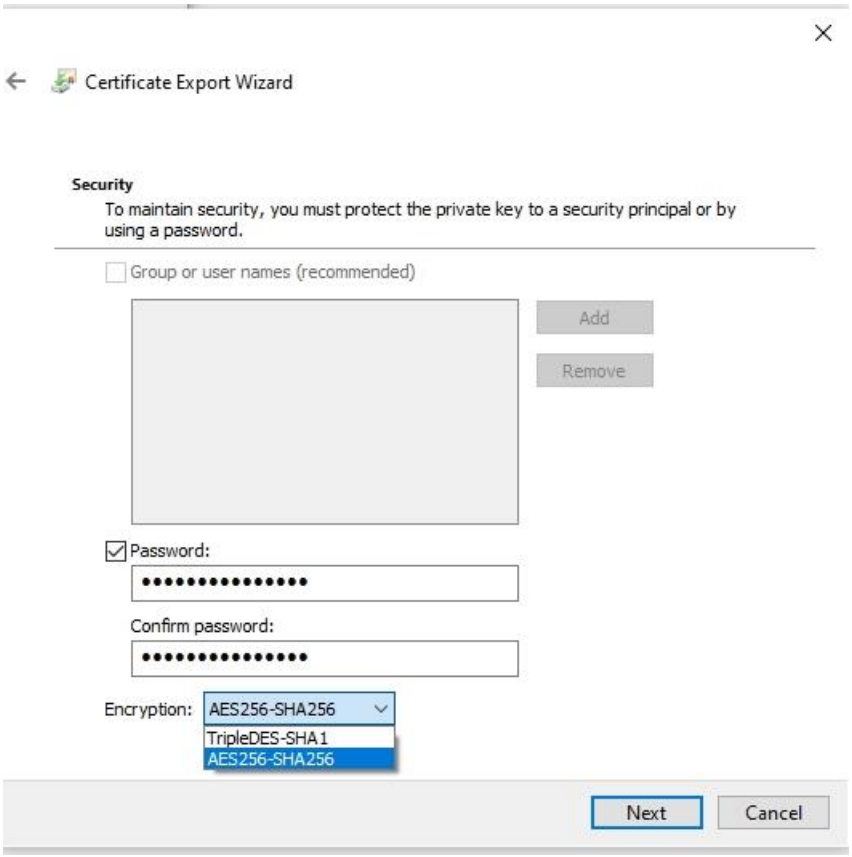


Рисунок 2.19 – Встановлення пароля на сертифікат та вибір методу шифрування сертифікату


Рисунок 2.20 – Вміст файлу сертифікату після шифрування та експорту

Для перевірки того, як працює шифрування наберемо довільний текст в зашифрованому файлі TestFile.txt (Рис 1.21).

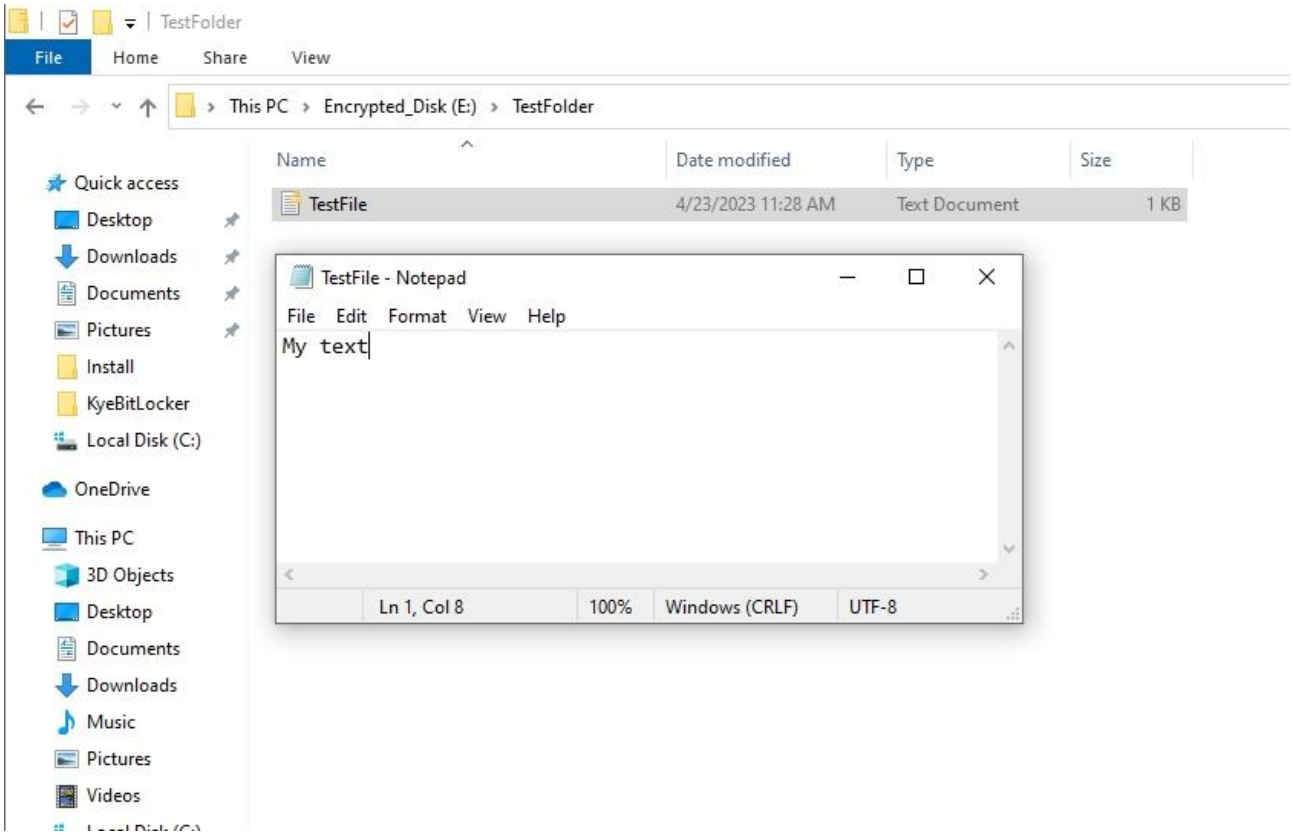


Рисунок 2.21 – Вміст зашифрованого файла

Здійснимо вхід в операційну стемну  під іншим користувачем, який також має права адміністратора. Як видно з рисунку 2.22 до файлу TestFile.txt адміністративна група має повний доступ.

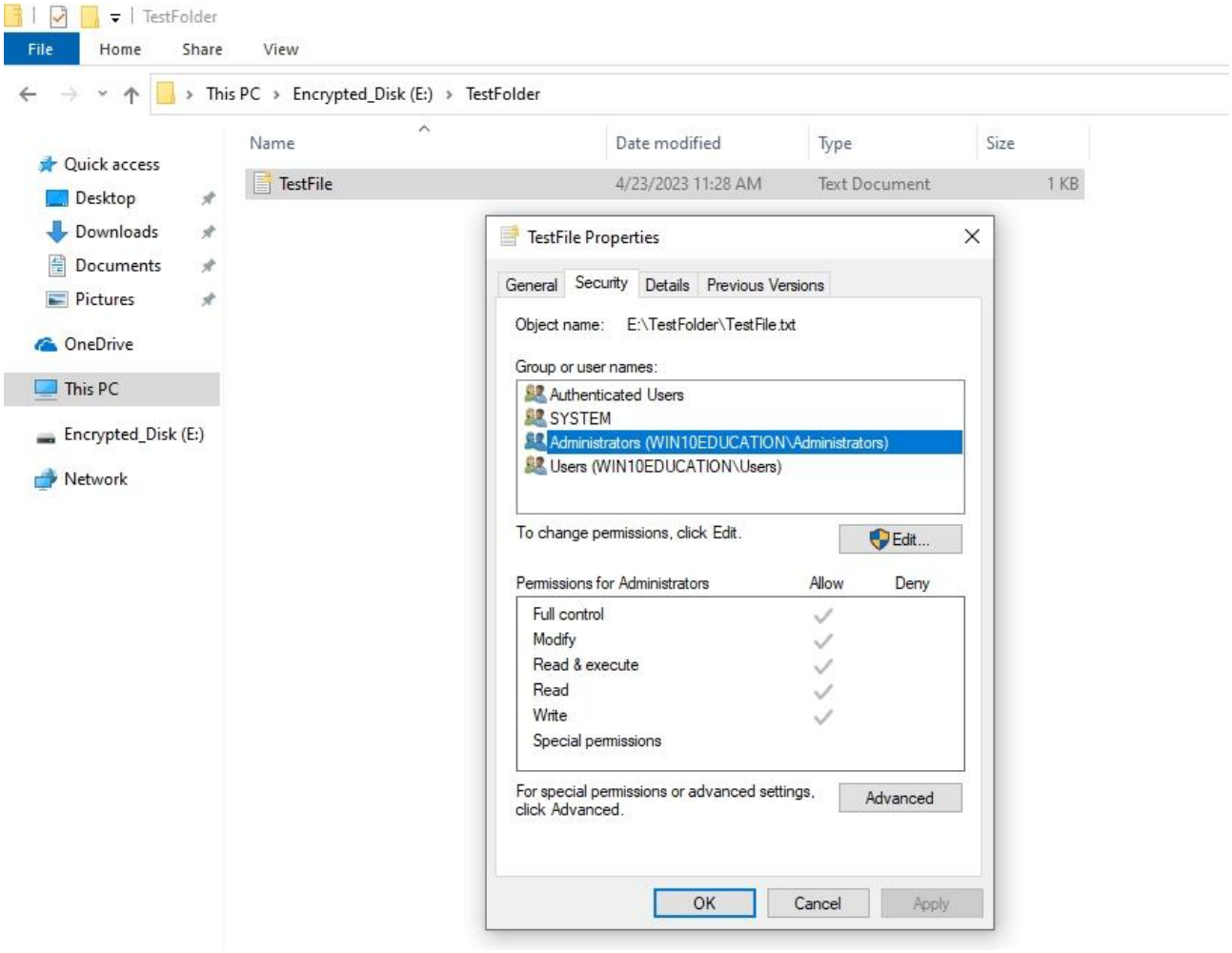


Рисунок 2.22 – Права доступу до зашифрованого файлу

Без імпорту сертифікату неможливо подивитись вміст файлу (Рис 2.23). Права доступу дозволяють виконати будь-які дії з файлом, але файл зашифровано і прочитати його вміст не можливо.

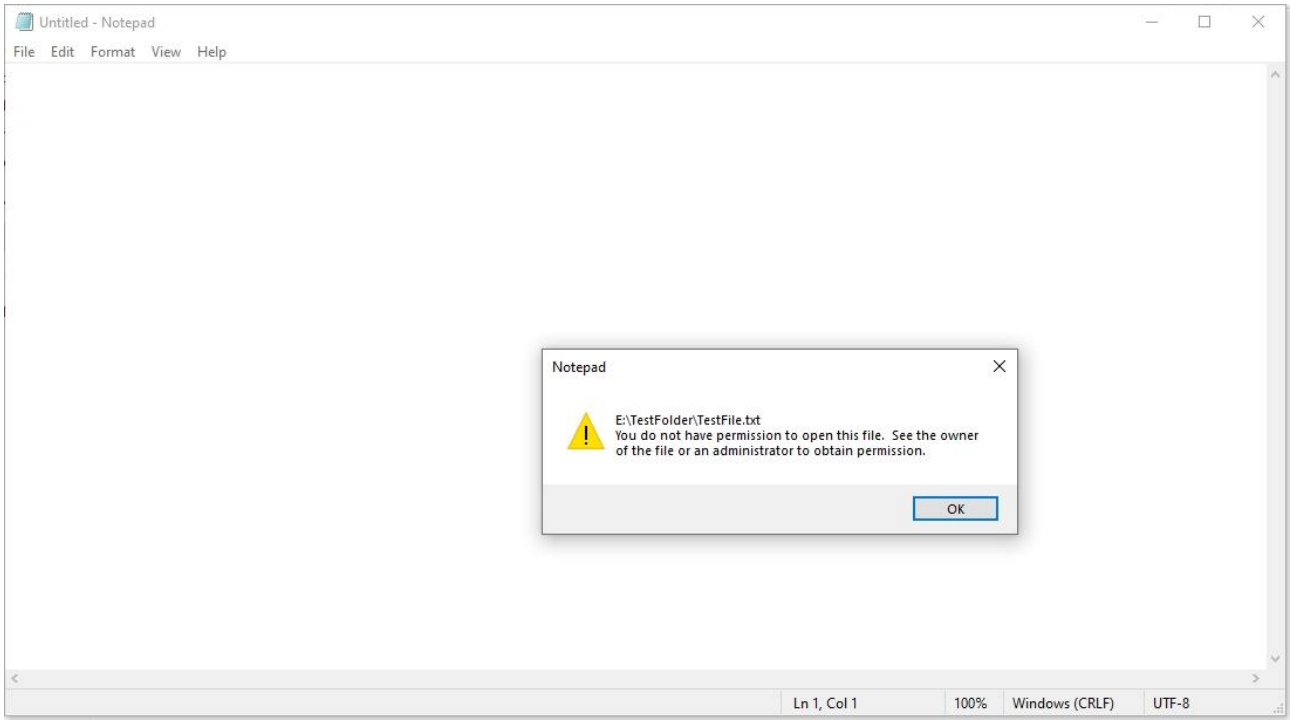


Рисунок 2.23 – Повідомлення при спробі перегляду вмісту шифрованого файлу

Імпорт сертифікату myefckye.pfx показано на рисунках 2.24-2.26

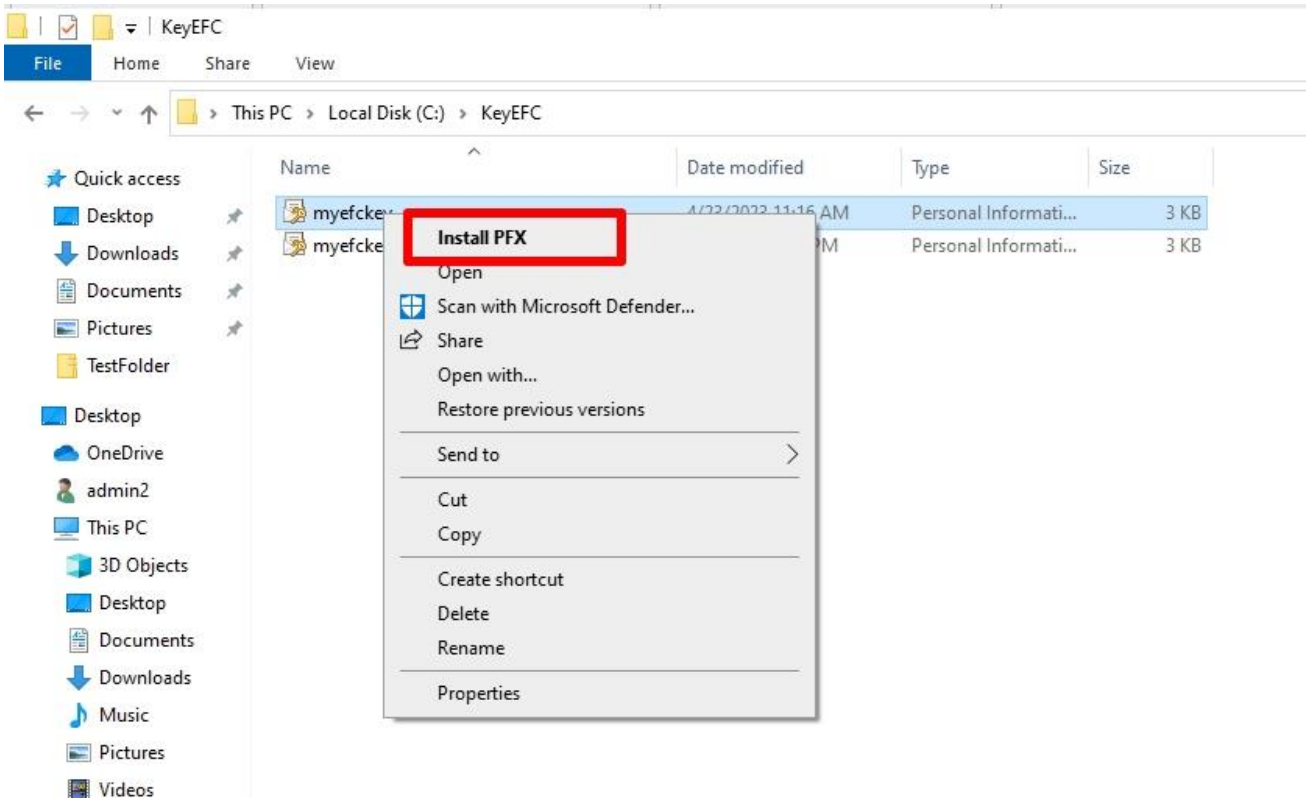


Рисунок 2.24 – Запуск процедури встановлення сертифікату

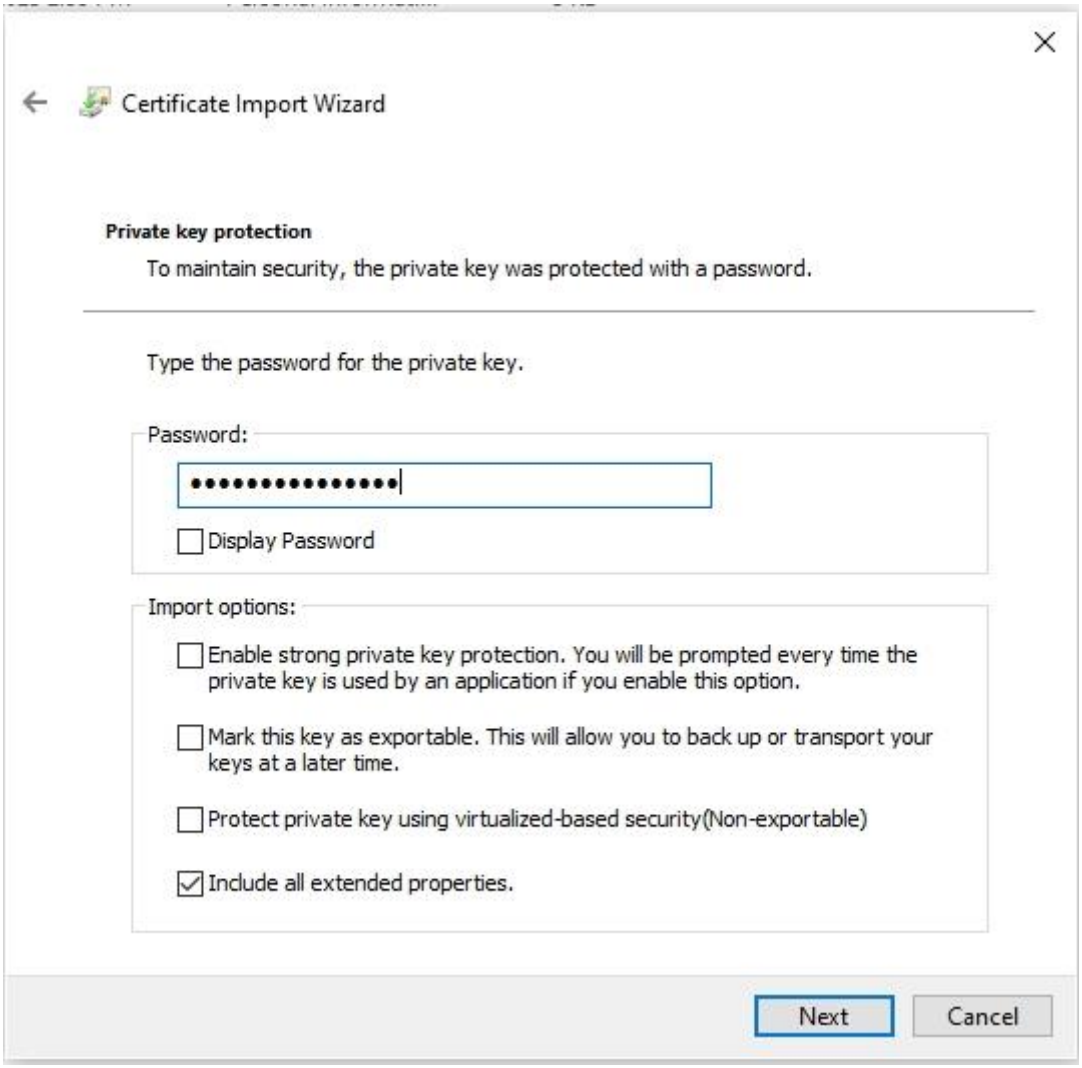


Рисунок 2.25 – Введення паролю для розблокування приватного ключа сертифікату

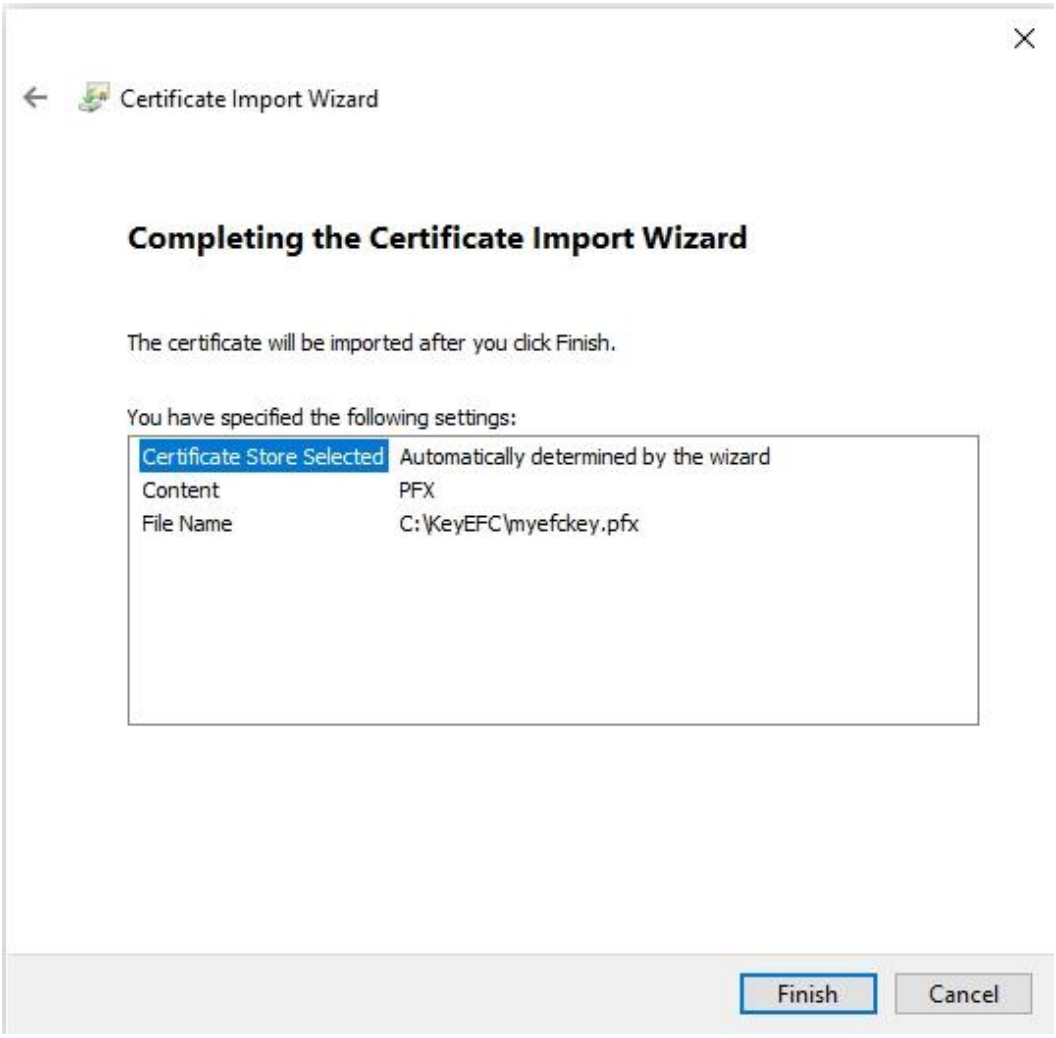


Рисунок 2.26 – Завершення процедури встановлення сертифікату

Після імпорту сертифікату можна подивитись докладну інформацію про сертифікат (Рис. 2.27-2.28).

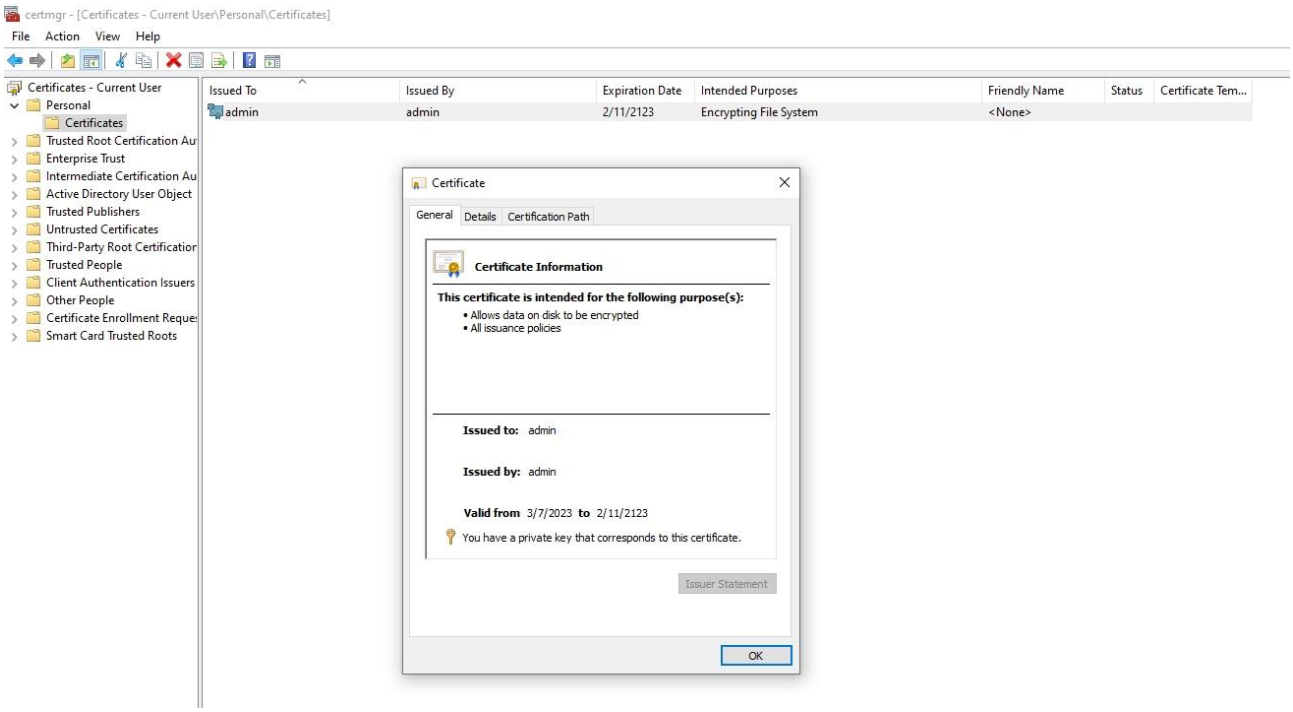


Рисунок 2.27 – Призначення та термін дії сертифікату

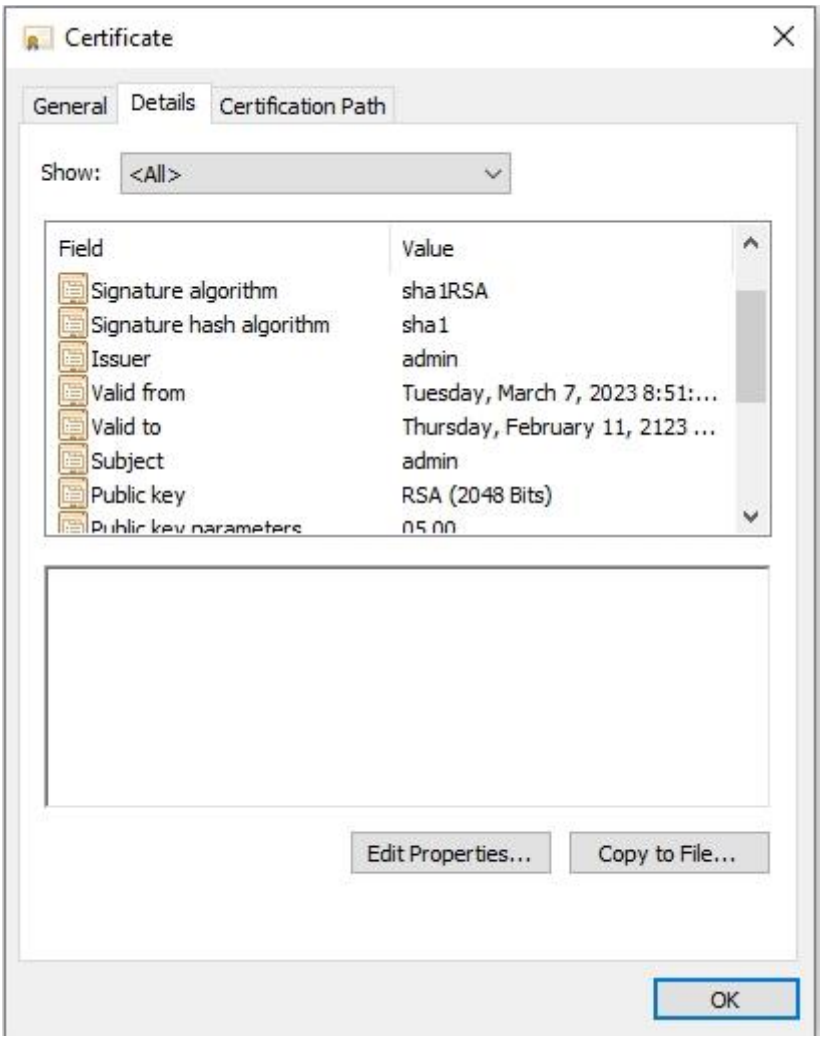


Рисунок 2.28 – Докладна інформація про сертифікат

Після імпорту сертифікату ми можемо побачити вміст зашифрованого TestFile.txt (Рис.2.29).

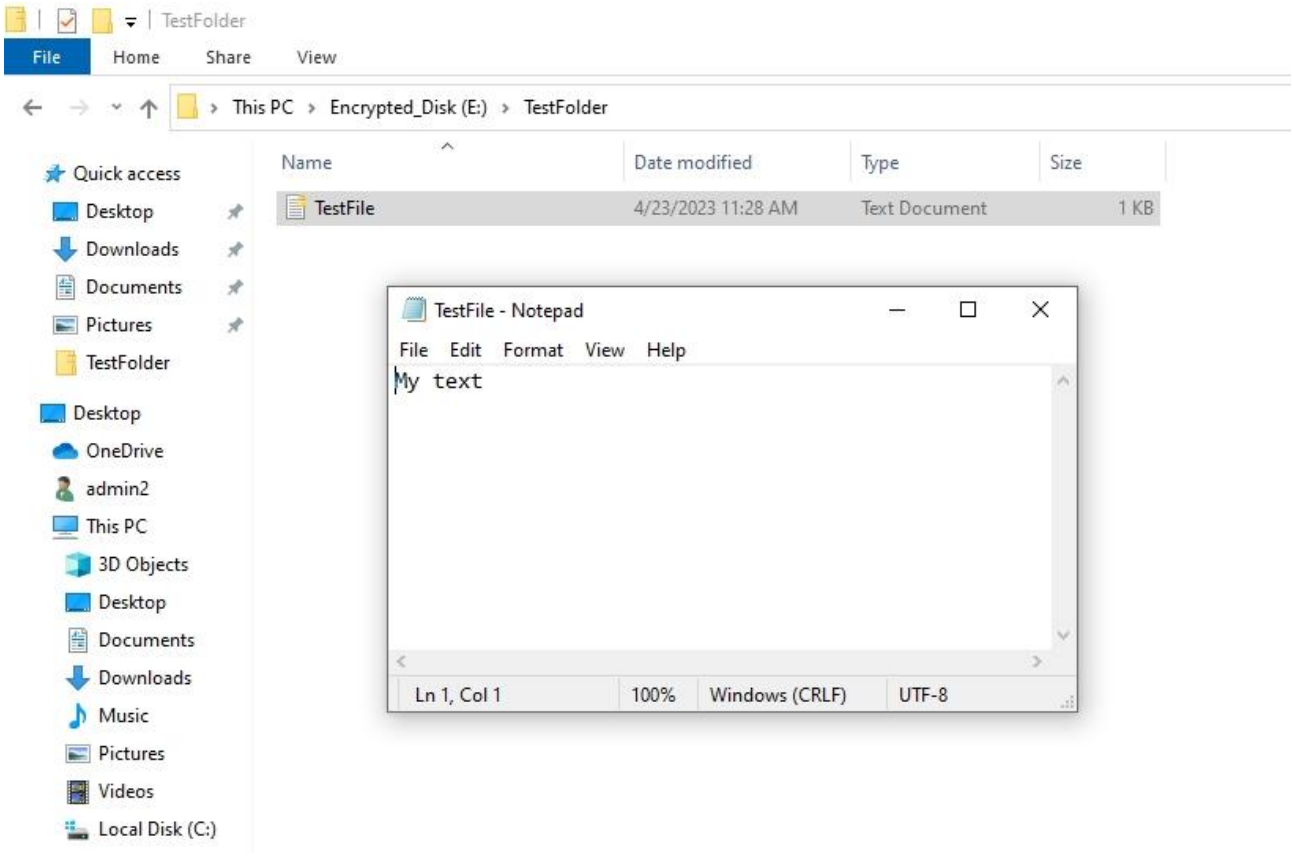


Рисунок 2.29 – Вміст зашифрованого файлу TestFile.txt

Важливо пам'ятати, що шифрування файлів за допомогою EFS не забезпечує повного захисту від зловмисників або хакерів, тому рекомендується використовувати його разом з іншими методами захисту даних. Крім того, необхідно регулярно оновлювати свій пароль та забезпечувати вчасну зміну сертифіката шифрування для збереження безпеки своїх даних.

## 2.2 Шифрування даних в операційній системі Linux

Linux є відкритою операційною системою з відкритим вихідним кодом. Існують різноманітні дистрибутиви Linux, такі як Ubuntu, Fedora, Debian, RedHat тощо. Вона широко використовується на серверах, вбудованих системах тощо.

Операційні системи типу Linux мають широкі можливості що до шифрування як всього дискового простору так і окремих тек чи файлів.

### 2.2.1 Шифрування даних за допомогою LUKS

Огляд можливостей шифрування.

У Linux шифрування дискового простору може бути досягнуте за допомогою LUKS, який дозволяє шифрувати весь диск або окремі розділи на рівні блокових пристроїв [9]. LUKS використовує пароль або ключ для розшифрування даних при зверненні до них.

Linux також надає можливість шифрування swap-простору, який використовується для обміну даними між оперативною пам'яттю та диском, коли пам'ять системи переповнена. Шифрування swap-простору дозволяє захистити дані, які можуть бути записані на диск з оперативної пам'яті, в тому числі чутливі дані, такі як паролі або ключі шифрування.

LUKS є стандартом шифрування дискових томів в Linux. LUKS забезпечує повний захист від несанкціонованого доступу до даних на диску. LUKS може бути використаний для шифрування внутрішніх дисків та зовнішніх пристроїв з підключенням через USB або інші варіанти.

LUKS використовує симетричне шифрування для шифрування кожного блоку диска за допомогою випадкового ключа, який згенерований під час створення шифрованого тома. Цей ключ шифрується за допомогою відкритого ключа RSA або за допомогою пароля користувача, який також використовується для розшифрування ключа під час розблокування диска.

Ключі можуть бути змінені або додані до системи в будь-який момент, і ці зміни можуть бути відображені на жорсткому диску без втрати даних. Ключі можуть також бути збережені на зовнішньому носії та імпортовані до системи в разі потреби.

Для використання LUKS необхідна програма cryptsetup, яка входить до складу більшості дистрибутивів Linux. Ця програма дозволяє створити новий зашифрований диск, відкрити його для роботи, змінити ключі та паролі, а також змінити рівень шифрування [10].

LUKS є потужним інструментом шифрування, який забезпечує високий рівень захисту даних на диску. Він є стандартом в більшості дистрибутивів

Linux. Він підтримує різні алгоритми шифрування, зокрема AES, Serpent, Twofish, Triple DES, Blowfish.

Приклад процедури шифрування.

Основна ідея LUKS полягає в тому, щоб використовувати ключову фразу або пароль для шифрування всього дискового простору. Після введення пароля, ключ для розшифрування диска автоматично генерується і зберігається в пам'яті, і використовується для доступу до даних на диску. Крім того, LUKS також забезпечує можливість створення резервних копій ключів та їх збереження на зовнішніх носіях.

Алгоритм шифрування за замовчуванням Argon2i для ключів шифрування та AES-XTS для шифрування даних.

При створенні LUKS-контейнера, який використовує Argon2i, пароль користувача перетворюється на ключ шифрування, за допомогою генератора випадкових чисел. Потім ключ шифрування шифрується самим алгоритмом Argon2i.

Таким чином, використання Argon2i в LUKS дозволяє забезпечити більш високий рівень безпеки шифрування дискового простору на Linux-системах, зменшуючи вразливість до парольних атак та інших видів криптоаналізу.

Нижче наведений приклад шифрування диска за допомогою LUKS в RedHat Linux [11].

Здійснимо шифрування тестового диску /dev/sda (Рис. 2.30) та зіставимо його в device mapped name з назвою “encrypted_disk”.

```
Disk /dev/sda: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Рисунок 2.30 – Параметри тестового диску для здійснення шифрування

Для виконання процесу шифрування та керування ним буде використано консольну утиліту cryptsetup.

Утиліта cryptsetup працює з командного рядка та дозволяє створювати та налаштовувати шифровані томи в Linux. Вона підтримує різні алгоритми шифрування та розміри ключів. Кілька параметрів cryptsetup, які допомагають налаштувати шифровані томи:

1) luksFormat - цей параметр використовується для форматування диска або розділу як LUKS-тома.

2) luksOpen - цей параметр використовується для відкриття LUKS-тома та створення віртуального пристрою для зчитування та запису даних на нього.

3) luksClose - цей параметр використовується для закриття LUKS-тома та звільнення зайнятих ним ресурсів.

4) luksDump - цей параметр використовується для виведення інформації про LUKS-том, таку як інформацію про ключі та алгоритми шифрування.

5) status - цей параметр дозволяє перевірити статус відкритого LUKS-тома.

6) --cipher - вказує шифрувальний алгоритм, що буде використовуватися для шифрування даних.

7) --key-size - встановлює розмір ключа шифрування в бітах.

8) --iter-time - встановлює час, що витрачається на кожну ітерацію алгоритму виведення ключа з пароля користувач.

9) --pbkdf - вказує алгоритм для виведення ключа з пароля користувач.

10) --hash - вказує хеш-функцію, яка використовується для виведення ключа з пароля користувача.

11) --type - встановлює тип контейнера шифрування, який буде створено.

12) --key-file - вказує шлях до файлу ключа, який буде використовуватися замість пароля користувача.

13) --keyfile-offset - вказує зміщення в байтах з початку файлу ключа, з якого починається ключ.

14) --keyfile-size - встановлює розмір ключа.

На початковому етапі диск /dev/sda не зашифровано. Що можна побачити з виводу команди lsblk -f /dev/sda (Рис. 2.31).


Рисунок 2.31 – Вивід команди перевірки відсутності шифрування диску

Після виконання процедури шифрування (Рис. 2.32 – Рис. 2.35) диск /dev/sda буде зашифровано за допомогою алгоритму aes-xts-plain64.

AES-XTS є одним з найбільш популярних алгоритмів шифрування, що використовуються в LUKS. XTS режим розроблений спеціально для шифрування даних на накопичувачах з використанням блочного шифрування з двома шифрувальними ключами.

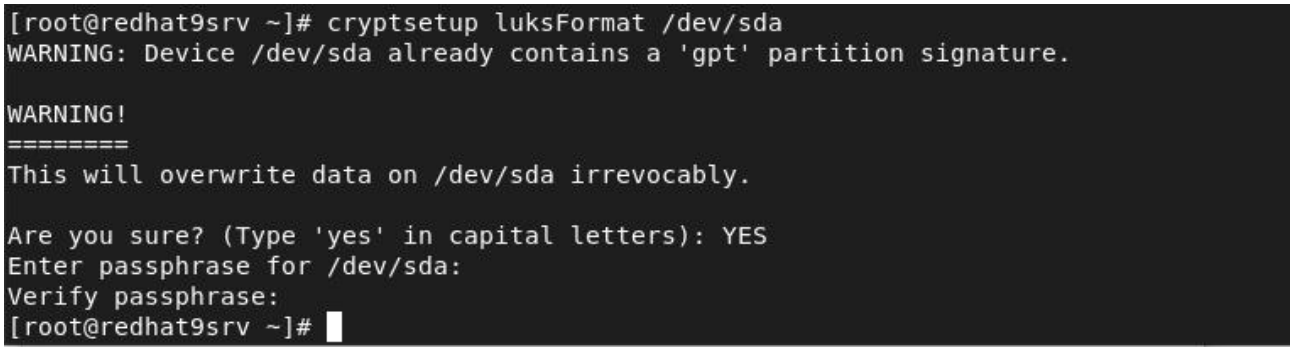


Рисунок 2.32 – Налаштування розділу як зашифрованого розділу LUKS та встановлення паролю розшифрування


Рисунок 2.33 – Розблокування розділу і відображення його на новому пристрої за допомогою device mapper


Рисунок 2.34 – Створення файлової системи

```
[root@redhat9srv ~]# mount /dev/mapper/encrypted_disk /mnt/my_encrypted_disk
[root@redhat9srv ~]# df -h
Filesystem                     Size  Used Avail Use% Mounted on
devtmpfs                       5.7G     0  5.7G   0% /dev
tmpfs                          5.8G     0  5.8G   0% /dev/shm
tmpfs                          2.3G   18M  2.3G   1% /run
/dev/mapper/rhel-root           36G  7.4G   28G  21% /
/dev/nvme0n1p2                1014M  455M  560M  45% /boot
/dev/nvme0n1p1                 599M  7.0M  592M   2% /boot/efi
/dev/mapper/rhel-home           18G  4.7G   13G  28% /home
tmpfs                          1.2G  140K  1.2G   1% /run/user/1000
/dev/mapper/encrypted_disk    1007M  4.0K 1007M   1% /mnt/my_encrypted_disk
[root@redhat9srv ~]#
```

Рисунок 2.35 – Монтування файлової системи

Переконаємось за допомогою команди lsblk -f /dev/sda що /dev/sda зашифровано. (Рис. 2.36).

```
[root@redhat9srv ~]# lsblk -f /dev/sda
NAME             FSTYPE   FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda              crypto_L 2           37143164-a97c-4841-9558-9e7b1664aa6c
└─encrypted_disk vfat     FAT32       0752-8CB3                             1006M     0% /mnt/my_encrypted_disk
[root@redhat9srv ~]# blkid /dev/sda
/dev/sda: UUID="37143164-a97c-4841-9558-9e7b1664aa6c" TYPE="crypto_LUKS"
[root@redhat9srv ~]# file -s /dev/sda
/dev/sda: LUKS encrypted file, ver 2 [, , sha256] UUID: 37143164-a97c-4841-9558-9e7b1664aa6c
[root@redhat9srv ~]#
```

Рисунок 2.36 – Вивід команд перевірки підтвердження шифрування диску

Виведемо докладнішу інформацію про зашифрований диск, таку як інформацію про ключі, алгоритми шифрування та інше (Рис 2.37).

```
[root@redhat9srv ~]# cryptsetup luksDump /dev/sda
LUKS header information
Version:        2
Epoch:          3
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           37143164-a97c-4841-9558-9e7b1664aa6c
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
        offset: 16777216 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 512 [bytes]

Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2id
        Time cost:  4
        Memory:     1048576
        Threads:    4
        Salt:       8c f3 5a a2 e2 8b 45 0b 8c 45 79 23 11 6c 45 f5
                    6c 33 74 22 2c cd 90 20 a7 e6 55 46 c1 d4 a3 9f
        AF stripes: 4000
        AF hash:    sha256
        Area offset:32768 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
Tokens:
Digests:
  0: pbkdf2
        Hash:       sha256
        Iterations: 42833
        Salt:       a7 d3 42 6c dc 73 89 0e 25 7c 32 53 c1 76 5f 93
                    40 dc 32 b9 3f 23 f5 ef 58 14 fa af 04 fb 6b fa
        Digest:     fd 9b 1e 98 80 5a 42 65 91 14 e7 be 2d af 7d cd
                    0b cf 42 b7 27 1c c2 0b 56 5f d0 55 fc 1d 78 70
[root@redhat9srv ~]#
```

Рисунок 2.37 – Вивід докладної інформації про зашифрований диск

Змінимо алгоритм шифрування LUKS за допомогою параметра –cipher утиліти cryptsetup (Рис 2.38) на serpent-xts-plain64.

Serpent-XTS-plain64 є алгоритмом шифрування, який використовується для захисту даних в LUKS. Він є комбінацією двох алгоритмів - Serpent та XTS.

Алгоритм Serpent є блочним шифром з 128-бітним блоком та ключем довжиною до 256 бітів. Serpent є сильним шифром з хорошою стійкістю до атак.

```
[root@redhat9srv ~]# cryptsetup luksFormat /dev/sda -c serpent-xts-plain64
WARNING: Device /dev/sda already contains a 'crypto_LUKS' superblock signature.

WARNING!
========
This will overwrite data on /dev/sda irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sda:
Verify passphrase:
```

Рисунок 2.38 – Зміна алгоритму шифрування

З виводу команди cryptsetup luksDump /dev/sda можна побачити що алгоритм шифрування змінився (Рис.2.39) на serpent-xts-plain64.

```
Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     serpent-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2id
        Time cost:  4
        Memory:     1048576
        Threads:    4
        Salt:       d0 f2 c2 e7 54 da 2a c5 65 7d c0 9a 3e 80 52 bd
                    4c 7a 89 85 75 4c 05 35 55 a0 8a 98 92 f7 54 4f
        AF stripes: 4000
        AF hash:    sha256
        Area offset:32768 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
```

Рисунок 2.39 – Підтвердження зміни алгоритму шифрування на serpent-xts-plain64

LUKS є стандартом шифрування дискових томів в Linux, що забезпечує високий рівень захисту даних та зручність управління зашифрованими даними.

### 2.2.2 Шифрування даних за допомогою EncFS

Огляд можливостей шифрування.

В Linux є різні інструменти щодо шифрування файлів та папок, які дозволяють шифрувати дані на рівні файлу або папки, створюючи віртуальні шифровані файлові системи.

EncFS - це найпоширеніша програма для шифрування файлів в Linux, яка створює зашифрований контейнер файлів, в якому зберігаються всі зашифровані дані. Для доступу до цих даних потрібно ввести пароль або ключ шифрування. EncFS підтримує створення зашифрованих томів файлів в режимі реального часу, що дозволяє автоматично шифрувати всі файли, що записуються в цей том. EncFS використовує асиметричне шифрування на основі пари ключів RSA для захисту ключів симетричного шифрування. EncFS підтримує шифрування з використанням різних алгоритмів семеричного шифрування, таких як AES, Blowfish, та інші, а також може використовувати ключі шифрування різної довжини. При використанні EncFS кожен файл шифрується і розшифровується окремо за допомогою симетричного шифрування, а ключі шифрування файлів зберігаються зашифрованими за допомогою пари ключів RSA, яка використовується для захисту цих ключів [12].

EncFS працює на рівні файлової системи, тому зашифровані файли можуть бути відкриті, редаговані та збережені так само, як і звичайні файли. EncFS має відкритий вихідний код і є доступним для багатьох дистрибутивів Linux. Його можна встановити через менеджер пакетів дистрибутиву. Також є можливість використовувати EncFS на інших операційних системах, таких як Unix і Windows.

Приклад процедури шифрування.

Щоб почати використовувати EncFS, спочатку потрібно створити шифрований том даних. Це можна зробити за допомогою командного рядка або

за допомогою графічного інтерфейсу користувача. При створенні шифрованого тому EncFS використовує пароль для шифрування та розшифрування даних.

Після того, як шифрований том даних створено, він може бути змонтований у системі в якості звичайної файлової системи. Коли шифрований том монтується, він стає доступним для запису та читання, як будь-який інший файл на комп'ютері. Якщо файл збережено у шифрованому томі, EncFS автоматично зашифровує його, щоб зберегти конфіденційність даних. Аналогічно, якщо файл зчитується з шифрованого диску, EncFS автоматично розшифровує його.

Одним з найважливіших аспектів використання EncFS є вибір правильного пароля. Пароль повинен бути достатньо складним та унікальним, щоб уникнути можливості його підбору або зламу. Крім того, важливо пам'ятати про потребу в регулярній зміні пароля.

Створимо теку, в якій будуть зберігатися розшифровані файли encrypted_folder та теку в яку будуть зберігатися зашифровані файли unencrypted_folder (Рис 2.40).

```
[root@redhat9srv ~]# mkdir /root/unencrypted_folder
[root@redhat9srv ~]# mkdir /root/encrypted_folder
```

Рисунок 2.40 – Створення тек encrypted_folder та unencrypted_folder

Для виконання процесу шифрування та керування ним буде використано консольну утиліту encfs [12].

Після виконання процедури шифрування (Рис. 2.41 – Рис. 2.43) тека unencrypted_folder буде містити фали або папки зашифровані за допомогою алгоритму AES 256.

```
[root@redhat9srv ~]# encfs /root/unencrypted_folder /root/encrypted_folder
Creating new encrypted volume.
Please choose from one of the following options:
 enter "x" for expert configuration mode,
 enter "p" for pre-configured paranoia mode,
 anything else, or an empty line will select standard mode.
?> x

Manual configuration mode selected.
The following cipher algorithms are available:
1. AES : 16 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes
2. Blowfish : 8 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes
3. CAMELLIA : 16 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes

Enter the number corresponding to your choice: 1

Selected algorithm "AES"
```

Рисунок 2.41 – Вибір алгоритму шифрування

```
Please select a key size in bits.  The cipher you have chosen
supports sizes from 128 to 256 bits in increments of 64 bits.
For example:
128, 192, 256
Selected key size: 256

Using key size of 256 bits
```

Рисунок 2.42 – Вибір ключа шифрування

При цьому EncFS запитує ввести пароль для шифрування. Після введення пароля, зашифрований вміст буде зберігатись в папці /root/encrypted_folder.

```
Configuration finished.  The filesystem to be created has
the following properties:
Filesystem cipher: "ssl/aes", version 3:0:2
Filename encoding: "nameio/block", version 4:0:2
Key Size: 256 bits
Block Size: 1024 bytes, including 16 byte MAC header
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.
File data IV is chained to filename IV.
File holes passed through to ciphertext.
```

Рисунок 2.43 – Завершення процесу шифрування та вивід інформації про параметри шифрування

Змонтуємо та розшифруємо вміст теки unencrypted_folder (Рис 2.44).. При цьому EncFS запитає ввести пароль, який був введений при створенні зашифрованого диску. Після введення правильного пароля, розшифрований том буде змонтований у теці /root/ecrypted_folder (Рис 2.45).

```
[root@redhat9srv ~]#
[root@redhat9srv ~]# encfs /root/unencrypted_folder /root/encrypted_folder
EncFS Password:
[root@redhat9srv ~]#
```

Рисунок 2.44 – Розшифрування та монтування вмісту теки unencrypted_folder

```
[root@redhat9srv run]# df -h
Filesystem             Size  Used Avail Use% Mounted on
devtmpfs               5.7G     0  5.7G   0% /dev
tmpfs                  5.8G     0  5.8G   0% /dev/shm
tmpfs                  2.3G   18M  2.3G   1% /run
/dev/mapper/rhel-root   36G  7.6G   28G  22% /
/dev/nvme0n1p2        1014M  455M  560M  45% /boot
/dev/nvme0n1p1         599M  7.0M  592M   2% /boot/efi
/dev/mapper/rhel-home   18G  4.7G   13G  28% /home
tmpfs                  1.2G  144K  1.2G   1% /run/user/1000
encfs                   36G  7.6G   28G  22% /root/encrypted_folder
[root@redhat9srv run]#
```

Рисунок 2.45 – Змонтована тека encrypted_folder

Тепер у теці /root/encrypted_folder будуть зберігатися розшифровані файли. Якщо ми змінимо або додамо файли до папки /root/encrypted_folder, то вони автоматично будуть з’являтись в теці /root/unencrypted_folder в зашифрованому вигляді.

Створимо текстовий файл encfstest.txt (Рис. 2.46) з довільним вмістом (Рис. 2.47).

```
[root@redhat9srv ~]#
[root@redhat9srv encrypted_folder]# touch /root/encrypted_folder/encfstest.txt
[root@redhat9srv encrypted_folder]#
[root@redhat9srv encrypted_folder]# ls -l
total 0
-rw-r--r--. 1 root root 0 Apr 28 15:40 encfstest.txt

[root@redhat9srv encrypted_folder]#
```

Рисунок 2.46 – Створення тестового файлу в теці encrypted_folder

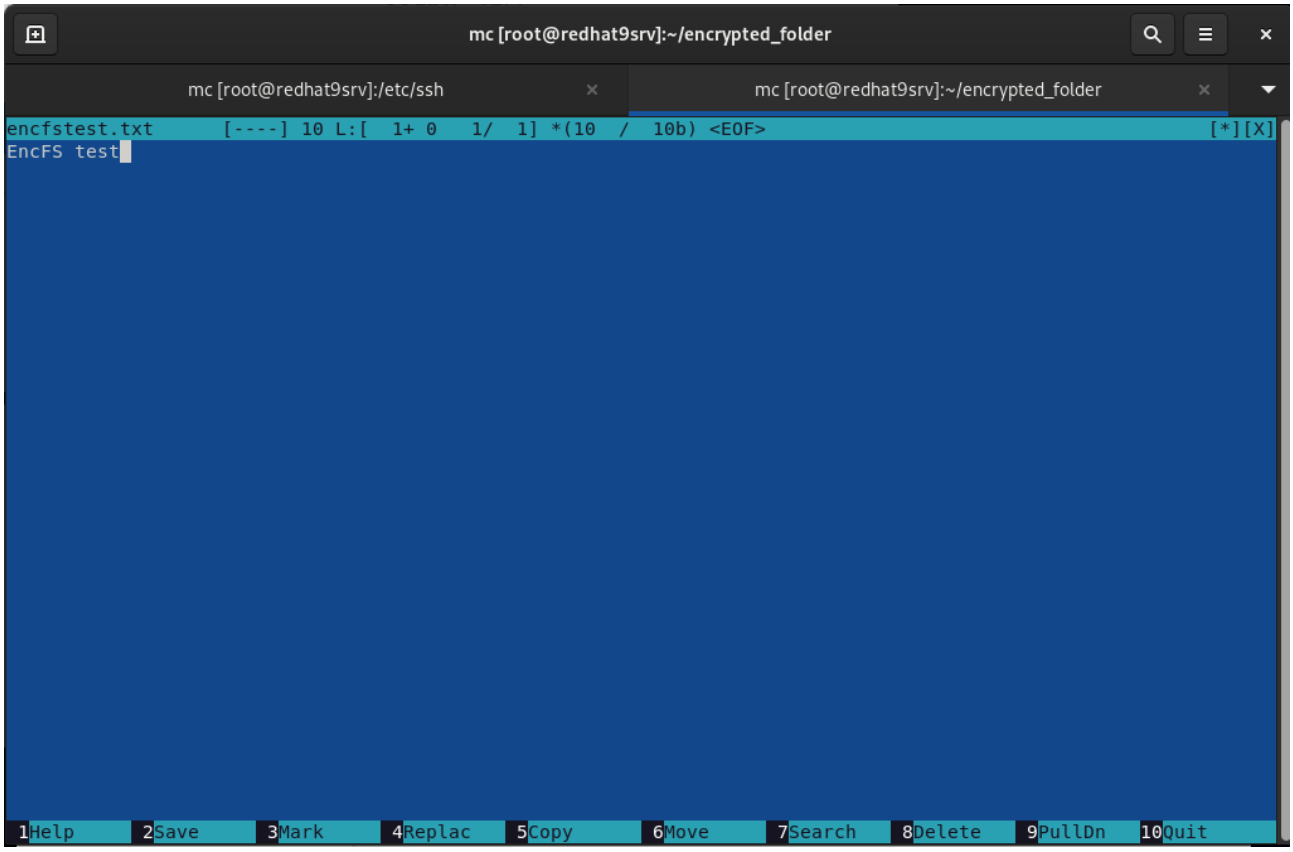


Рисунок 2.47 – Вміст файлу encfstest.txt

Тепер у папці /root/unencrypted_folder з’явився зашифрований файл (Рис 2.48 та Рис.2.49).

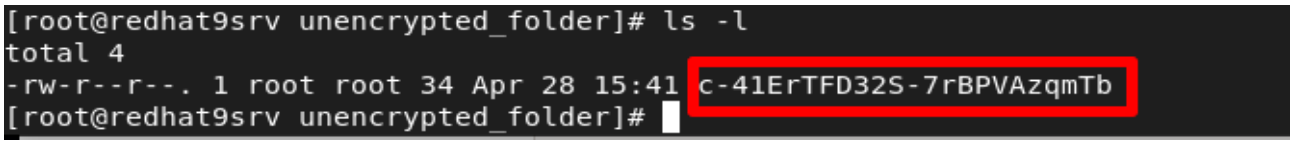


Рисунок 2.48 – Зашифрована назва файлу encfstest.txt


Рисунок 2.49 – Зашифрований вміст файлу encfstest.txt

У EncFS ключі шифрування зберігаються в спеціальному файлі, який називається ".encfs6.xml". Цей файл зберігається у каталозі, в якому створюються зашифровані файли та теки (Рис 2.50).

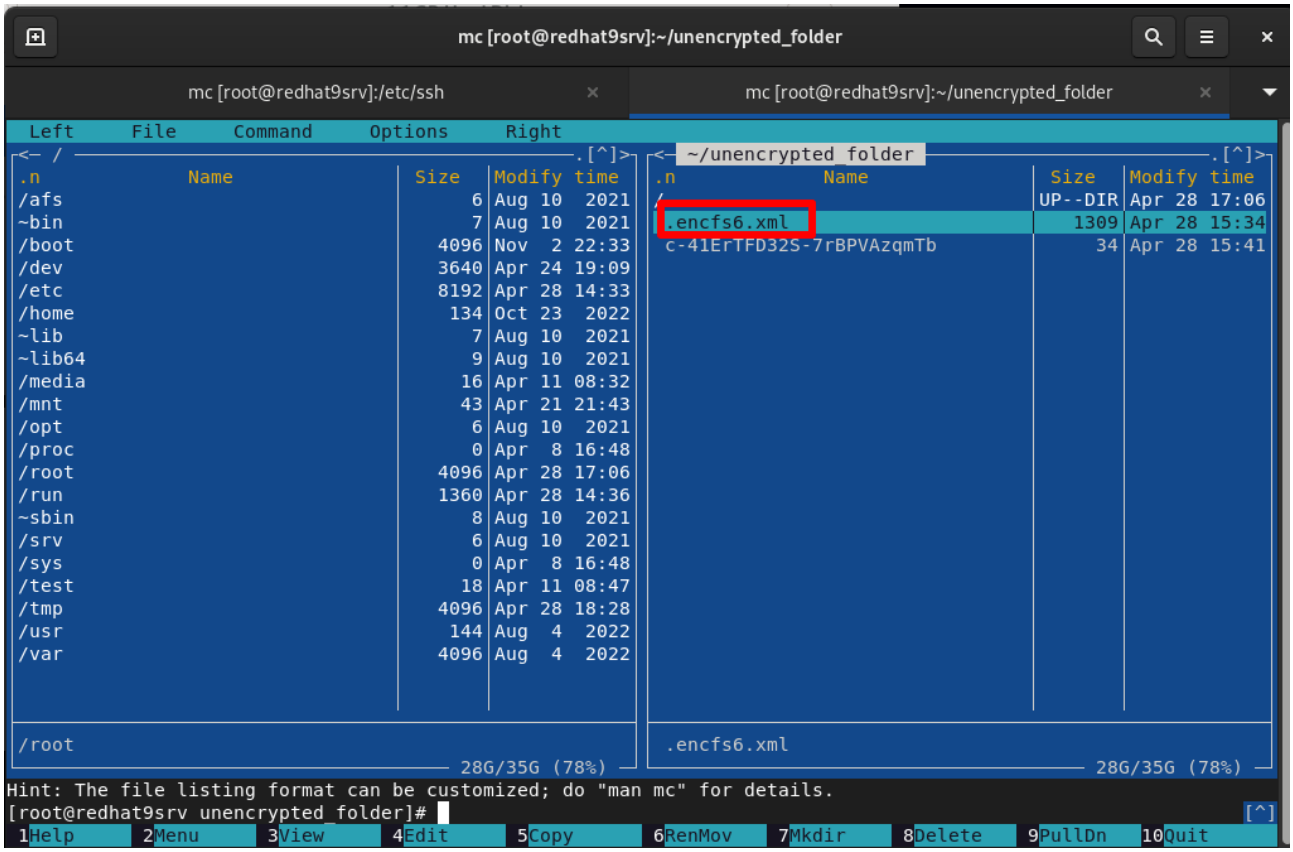


Рисунок 2.50 – Спеціальний ".encfs6.xml" файл для збереження ключа шифрування

Ключі шифрування знаходяться в зашифрованому вигляді в цьому файлі, і їх можна розшифрувати лише з використанням правильного пароля.

Якщо ви перенесете каталог з зашифрованими даними в інший каталог або на інший комп'ютер, потрібно забезпечити наявність файлу ".encfs6.xml" в тому ж каталозі, що і зашифровані дані. Інакше EncFS не зможе розшифрувати дані.

EncFS є потужним інструментом для шифрування конфіденційних даних на комп'ютері. Він забезпечує високий рівень захисту даних, простий у використанні та підтримується на різних операційних системах.

# 3 ПЕРЕВАГИ, НЕДОЛІКИ ТА КРОС-ПЛАТФОРМНІСТЬ МЕТОДІВ ШИФРУВАННЯ

## 3.1 Переваги та недоліки методів шифрування даних в Windows

### 3.1.1 BitLocker

Не дивлячись на те що BitLocker забезпечує захист даних на Windows-платформі він має також свої переваги та недоліки.

Основні переваги:

1) Безпека. BitLocker забезпечує високий рівень безпеки для даних, збережених на диску. Всі дані, які зберігаються на зашифрованому диску, неможливо прочитати без ключа.
2) Легкість використання. BitLocker дуже простий у використанні і може бути настроєний всього за декілька хвилин. Крім того, він може бути використаний з дисками різного типу, включаючи зовнішні диски та флешки.
3) Інтеграція з Windows. BitLocker інтегрований з ОС Windows, що дозволяє використовувати його безпосередньо з ОС та налаштовувати його за допомогою звичайних інструментів ОС.
4) Інтеграція з TPM. BitLocker за замовчуванням вимагає наявності TPM 2.0, що забезпечує безпеку ключа шифрування і запобігає певним видам атакам. Це означає, що ваші дані будуть захищені від будь-яких спроб злому.

Основні недоліки:

1) Обмеження версій Windows. BitLocker доступний не на всіх версій Windows. Наприклад він не доступний в Windows 10 Home. Це означає, що користувачі інших версій ОС Windows не можуть використовувати його.

2) Вимога запуску процесу відновлення BitLocker. Є багато конкретних подій, через які BitLocker перейде в режим відновлення під час спроби операційну систему з диску.
3) Вплив на продуктивність. Шифрування BitLocker може вплинути на продуктивність вашого комп'ютера, особливо на старіших комп'ютерах з менш потужними процесорами. Процес шифрування та розшифрування даних може займати додатковий час.
4) Вартість. Якщо вам потрібно використовувати BitLocker на комп'ютері з Windows, який не підтримує його, ви повинні придбати версію Windows, яка підтримує BitLocker. Це є додаткові витрати.
5) Сумісність. Якщо ви вирішили використовувати BitLocker, потрібно впевнитися, що він сумісний з іншими програмними продуктами, які ви використовуєте.

BitLocker - це дуже ефективний і простий у використанні інструмент для шифрування диску на Windows-платформі. Однак, перед тим, як вибрати його, важливо врахувати його обмеження та вимоги до обладнання.

### 3.1.2 Encrypting File System

EFS є вбудованим в Windows інструментом, який дозволяє шифрувати файли і папки на диску з метою захисту конфіденційної інформації. EFS також має свої переваги та недоліки:

Переваги:

1) Легкий у використанні. EFS простий в налаштуванні і використанні. Він дозволяє шифрувати файли і папки на диску за допомогою простого інтерфейсу.
2) Інтегрований з системою. EFS є частиною операційної системи Windows, тому його можна використовувати без необхідності встановлювати додаткові програми.
3) Швидкість. EFS не сильно впливає на продуктивність системи.

Недоліки:

1) Обмежена підтримка. EFS працює тільки на Windows і не підтримується на інших операційних системах.
2) Не захищає від всіх загроз. EFS не захищає файли від атак, які використовуються для отримання доступу до файлів під час виконання атаки на рівні ядра.
3) Відновлення ключів. якщо ключі EFS були втрачені або пошкоджені, можливо, не вдасться відновити файли і папки, які були зашифровані.

## 3.2 Переваги та недоліки методів шифрування даних в Linux

### 3.2.1 Linux Unified Key Setup

LUKS - це стандарт відкритих інтерфейсів для шифрування дискових просторів в Linux. LUKS2 є останньою версією LUKS, він має свої переваги та недоліки.

Переваги LUKS2:

1) Підтримка більш довгих ключів шифрування (до 512 біт), що забезпечує більшу безпеку даних.
2) Можливість шифрування даних на віртуальних дисках.
3) Підтримка додаткових шифрувальних алгоритмів, включаючи AES-XTS, Serpent-XTS, Twofish-XTS, Camellia- XTS та інші.
4) Наявність резервної копії заголовка диска, яка дозволяє відновлювати дані в разі пошкодження або втрати заголовка.
5) LUKS надає простий і зручний інтерфейс для створення, відкриття, закриття та зміни параметрів зашифрованого тому. Це дозволяє легко керувати доступом до даних і забезпечувати безпеку шифрування.

Недоліки LUKS2:

1) Більш висока вимога до обчислювальних ресурсів, порівняно з попередньою версією LUKS.

2) Управління ключами шифрування в LUKS може бути викликом, особливо якщо ви маєте багато зашифрованих томів або велику кількість користувачів.
3) Проблеми сумісності з попередніми версіями LUKS, що робить неможливим доступ до даних.
4) Обмежена підтримка на іншиї операційних системах, що може зменшити універсальність використання.

### 3.2.2 Encrypted File System

EncFS - це програмне забезпечення для створення зашифрованих томів файлів на операційній системі Linux, яке дозволяє користувачам створювати зашифровані контейнери, що містять файли та каталоги. Ось деякі переваги та недоліки EncFS:

Переваги:

1) Захист від несанкціонованого доступу. EncFS зашифровує файли та каталоги, що дозволяє користувачам зберігати важливу конфіденційну інформацію безпечно, навіть якщо хто-небудь має фізичний доступ до комп'ютера.
2) Файли можуть бути синхронізовані. EncFS дозволяє користувачам синхронізувати файли та каталоги між різними пристроями, що забезпечує зручну роботу з даними.
3) Безкоштовне програмне забезпечення. EncFS є відкритим програмним забезпеченням та доступний безкоштовно.
4) Простий у використанні. EncFS має простий інтерфейс, що дозволяє користувачам створювати та змінювати зашифровані файли та каталоги.

Недоліки:

1) Залежність від базової файлової системи. EncFS залежить від базової файлової системи, що може вплинути на продуктивність.

2) Можливість атак на безпеку. якщо ключ шифрування EncFS буде скомпрометований, зловмисник може отримати доступ до зашифрованих даних.
3) Проблеми з використанням на віддалених серверах: EncFS може працювати повільно на віддалених серверах з великою кількістю файлів;
4) Обмеження на розмір файлів: EncFS може мати обмеження на розмір файлів, що може бути зашифровано.
5) Використання резервного копіювання в ручному режимі. EncFS не має вбудованого механізму автоматичного резервного копіювання, тому користувачам доведеться вручну створювати резервні копії своїх зашифрованих даних.
6) Вимоги до знань користувача. EncFS вимагає певних знань зі створення та використання зашифрованих томів файлів, тому може бути важким у використанні для користувачів з обмеженим досвідом роботи з шифруванням.

### 3.3 Крос-платформність методів шифрування

Шифрування даних на Windows та Linux може бути сумісним, якщо використовуються спільні стандарти та формати шифрування. Важливо враховувати, що хоча деякі методи шифрування можуть бути сумісними на рівні алгоритмів шифрування, формати контейнерів або томів можуть відрізнятися. Тому перед використанням шифрування даних на різних платформах слід ознайомитися зі специфічними реаліями інструментів шифрування, які ви плануєте використовувати.

Є деякі інструменти, які дозволяють працювати з LUKS на Windows, такі як FreeOTFE та LibreCrypt. Однак, їх підтримка є обмеженою та не завжди гарантує повну сумісність з усіма версіями Windows.

Якщо потрібно використовувати зашифрований диск на Linux та на Windows, то краще розглянути альтернативні формати шифрування, які підтримуються на обох операційних системах, такі як BitLocker. Це дозволить

зберігати та обмінюватися даними між Windows та Linux без проблем зі сумісністю.

Якщо ви вже використовуєте LUKS для захисту своїх даних на Linux-системі та потрібно отримати доступ до цих даних на Windows, то зручніше буде скопіювати дані з зашифрованого диску LUKS на новий диск, який буде зашифрований форматом шифрування, підтримуваним на Windows.

Для цього вам потрібно буде розшифрувати диск LUKS на Linux-системі за допомогою пароля або ключа шифрування, а потім створити новий зашифрований диск на Windows, скопіювати дані на нього та зашифрувати його за допомогою підтримуваного формату шифрування, такого як BitLocker.

При цьому, важливо впевнитися, що новий зашифрований диск на Windows буде захищений надійним паролем або ключем шифрування та буде зберігатися в безпечному місці. Крім того, варто пам'ятати про те, що копіювання даних з зашифрованого диску може бути ресурсозатратним процесом, залежно від обсягу даних та швидкості диску.

Диск, який зашифровано BitLocker, може бути використаний в Linux, це може потребувати не значних додаткових зусиль та налаштувань.

Існують різні програми для розшифрування BitLocker на Linux. Одна з них - dislocker, яка є відкритим програмним забезпеченням. Dislocker дозволяє зчитувати дані з зашифрованого диску BitLocker на Linux, а також записувати дані на диск, якщо він розблокований [13].

Як правило в сучасних операційних системах Linux (RedHat, Ubuntu, Oracle та інші) dislocker є встановлений за замовчуванням після інсталяції системи. Якщо dislocker відсутній його можна легко встановити з портів чи вихідних файлів. Після цього можна виконати команду для розблокування зашифрованого диску і змонтувати його в системі Linux. Однак, важливо знати, що розшифрування BitLocker на Linux може бути складнішим процесом, ніж на Windows, і може вимагати додаткових знань та навичок [14].

Проведемо розшифрування тестового USB флеш диску з міткою диску “Encrypted_F” в RedHat Linux. Диск був попередньо зашифрованого за допомогою Bitlocker в Windows 10 (Рис. 3.1).

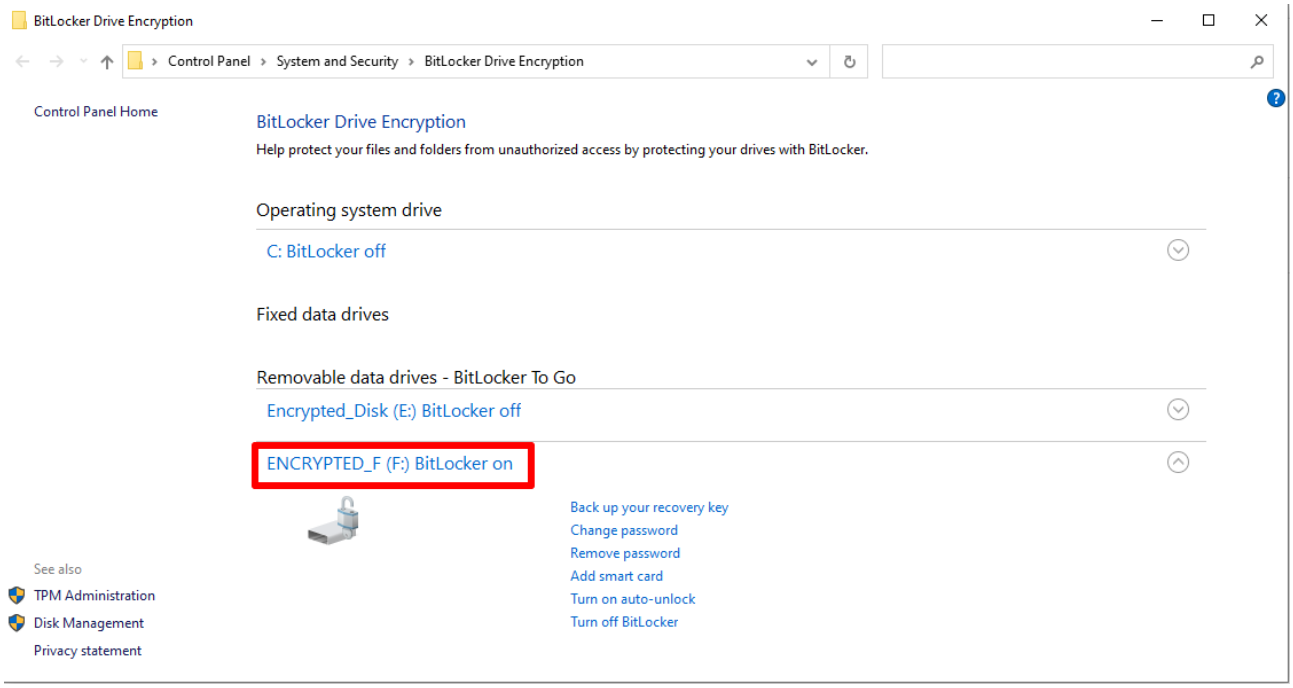


Рисунок 3.1 – USB флеш зашифрована Bitlocker

Диск містить тестовий файл TestFileBitlocker.txt з довільним вмістом (Рис. 3.2).

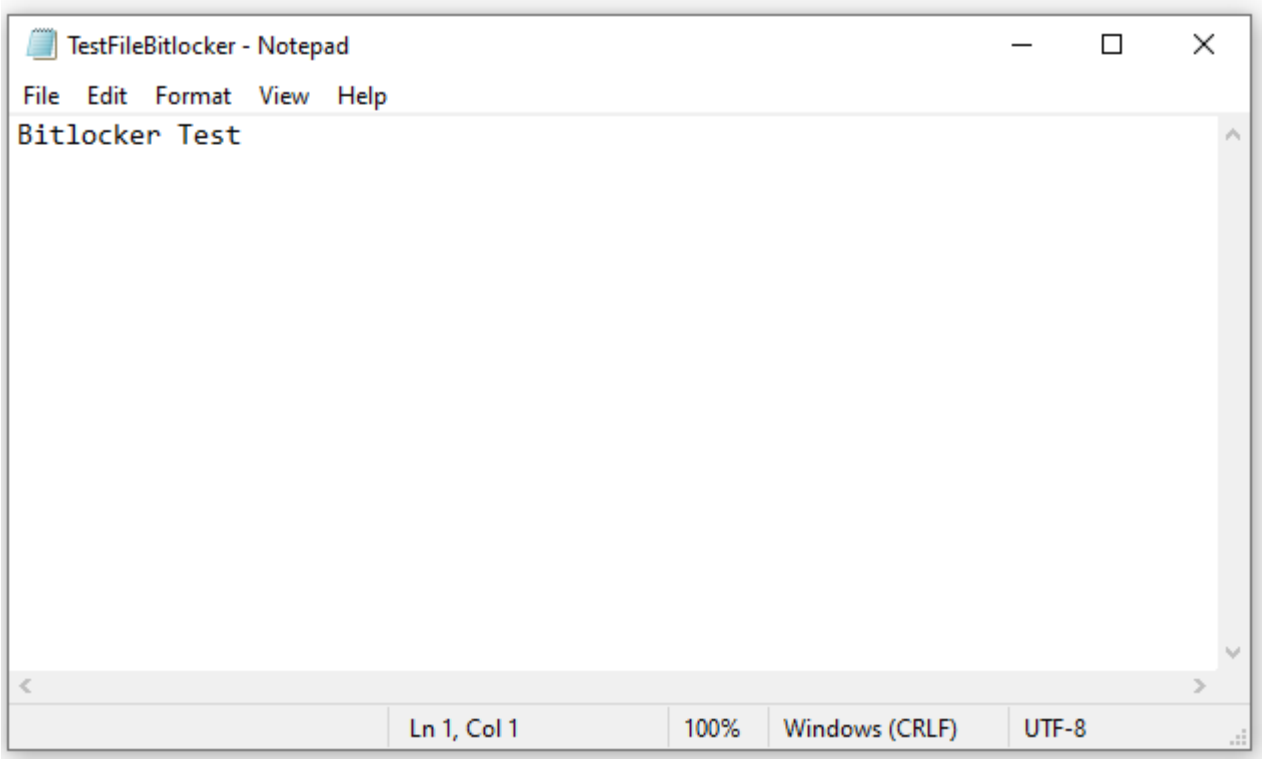


Рисунок 3.2 – Вміст файлу TestFileBitlocker.txt

Після виконання процедури розшифрування (Рис. 3.3) можна переконатись що вміст диску (Рис. 3.4) та файлу (Рис. 3.5) відображається коректно.

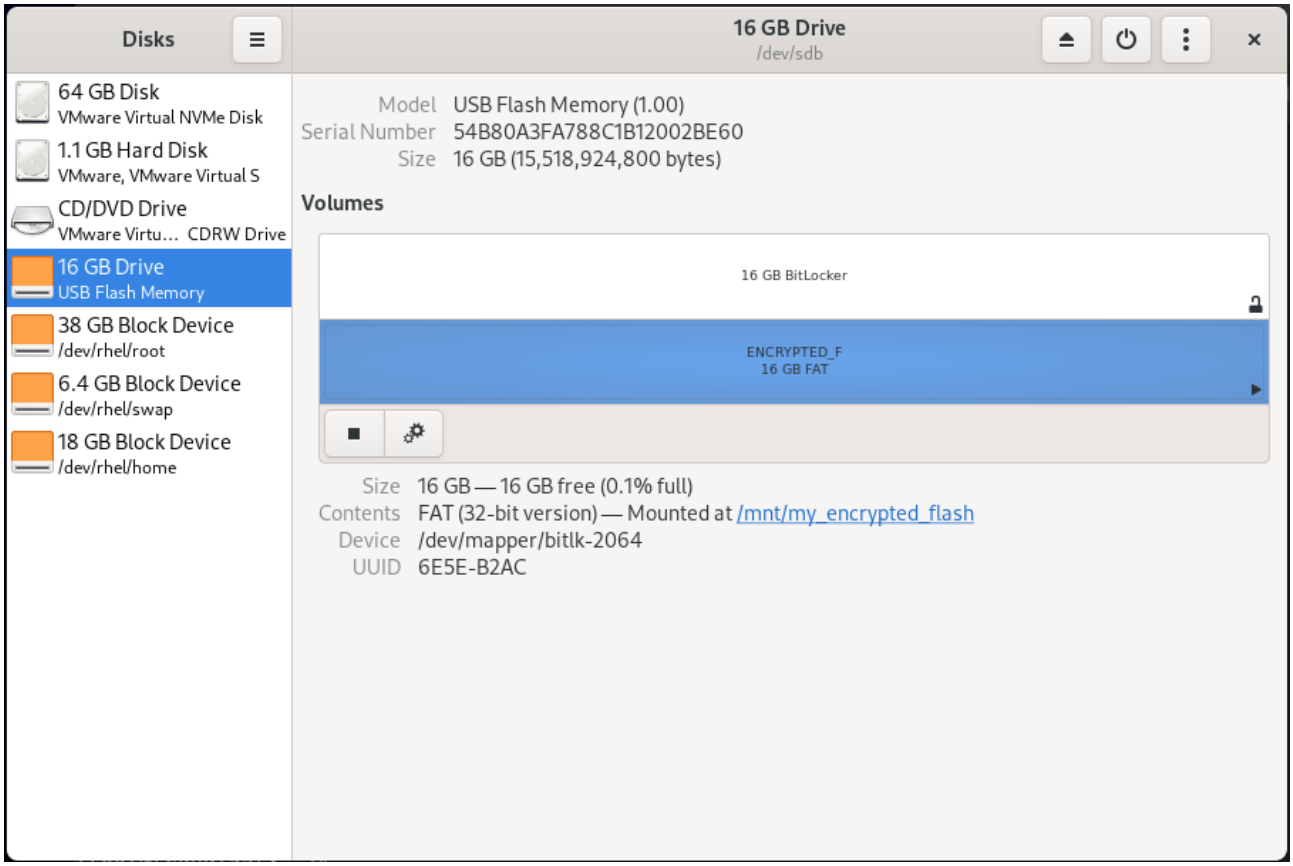


Рисунок 3.3 – Розшифрований на змонтований USB диск

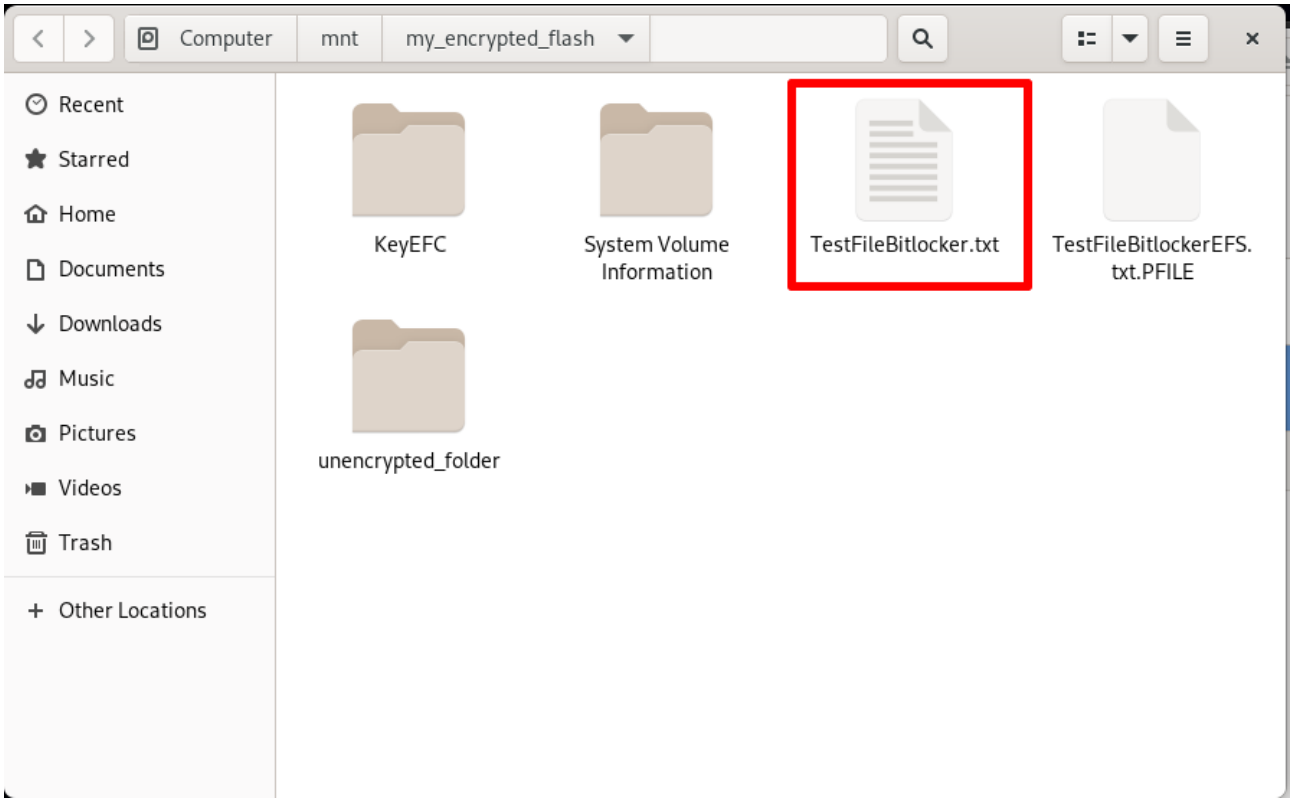


Рисунок 3.4 – Вміст  USB диску


Рисунок 3.5 – Вміст файлу TestFileBitlocker.txt відкритий в Redhat Linux

EncFS в Linux створює віртуальний зашифрований том, який теоретично може бути використаний в будь-якій операційній системі, включаючи Windows. Однак, EncFS спеціально створений для Linux-систем і не підтримується офіційно на Windows.

Тому, щоб відкрити зашифрований файл, створений з EncFS, на Windows, вам необхідно буде встановити додаткове програмне забезпечення, яке підтримує EncFS. Є кілька програм для Windows, наприклад EncFSMP, які підтримують EncFS і дозволяють розшифрувати файли, створені за допомогою EncFS.

У будь-якому випадку, коректність роботи EncFS на Windows не гарантується, тому перед використанням EncFS на Windows необхідно здійснити відповідні дослідження та переконатися, що програмне забезпечення працює стабільно і безпечно в операційній системі.

EncFSMP - це безкоштовна крос-платформна програма, яка дозволяє шифрувати та розшифровувати файли та теки за допомогою EncFS. EncFSMP підтримує Windows та Linux, що дозволяє використовувати EncFS на різних операційних системах. Дана програма має графічний інтерфейс користувача, що

дозволяє створювати та керувати зашифрованими томами файлів. Вона пропонує можливості налаштування параметрів шифрування, таких як алгоритми шифрування та хешування, а також паролі та ключі для захисту файлів. EncFSMP дозволяє підключати EncFS томи файлів як віртуальні диски на комп'ютері та працювати з ними як звичайними дисками.

Скопіюємо теку unencrypted_folder створену в пункті 2.2.2 на USB флеш. Дата тека містить зашифрований файл encfstest.txt та файл .encfs6.xml з параметрами шифрування.

Для виконання процесу розшифрування та керування ним буде використано графічну програму EncFSMP в Windows 10.

Послідовність розшифрування це досить простий процес за умови коректного налаштування програми розшифрування (Рис. 3.6 - Рис. 3.10).

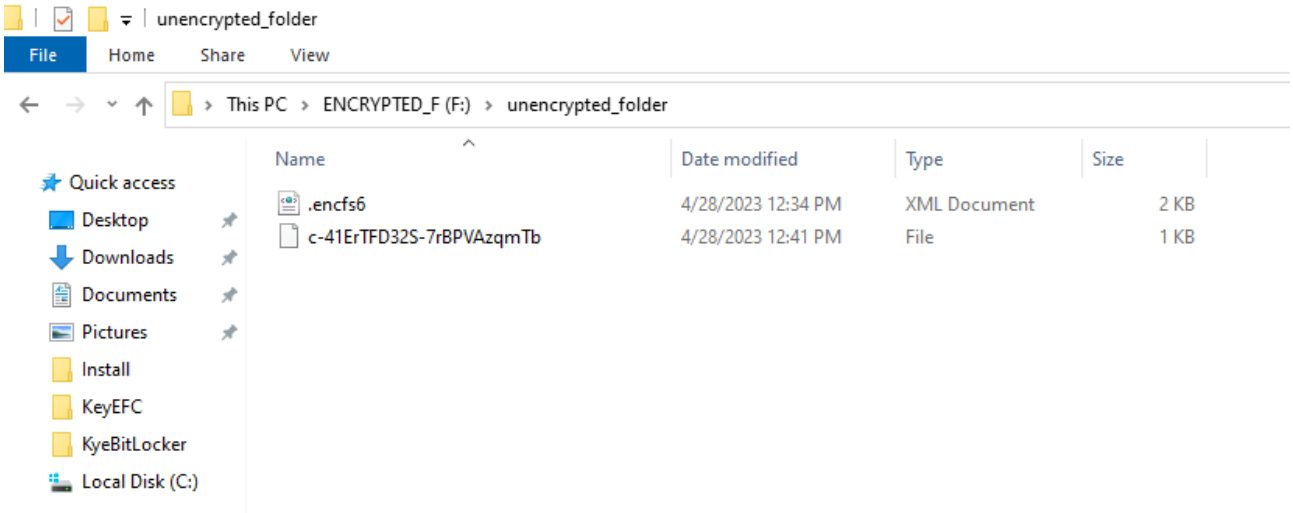


Рисунок 3.6 – Зашифрований файл encfstest.txt та файл .encfs6.xml з параметрами шифрування

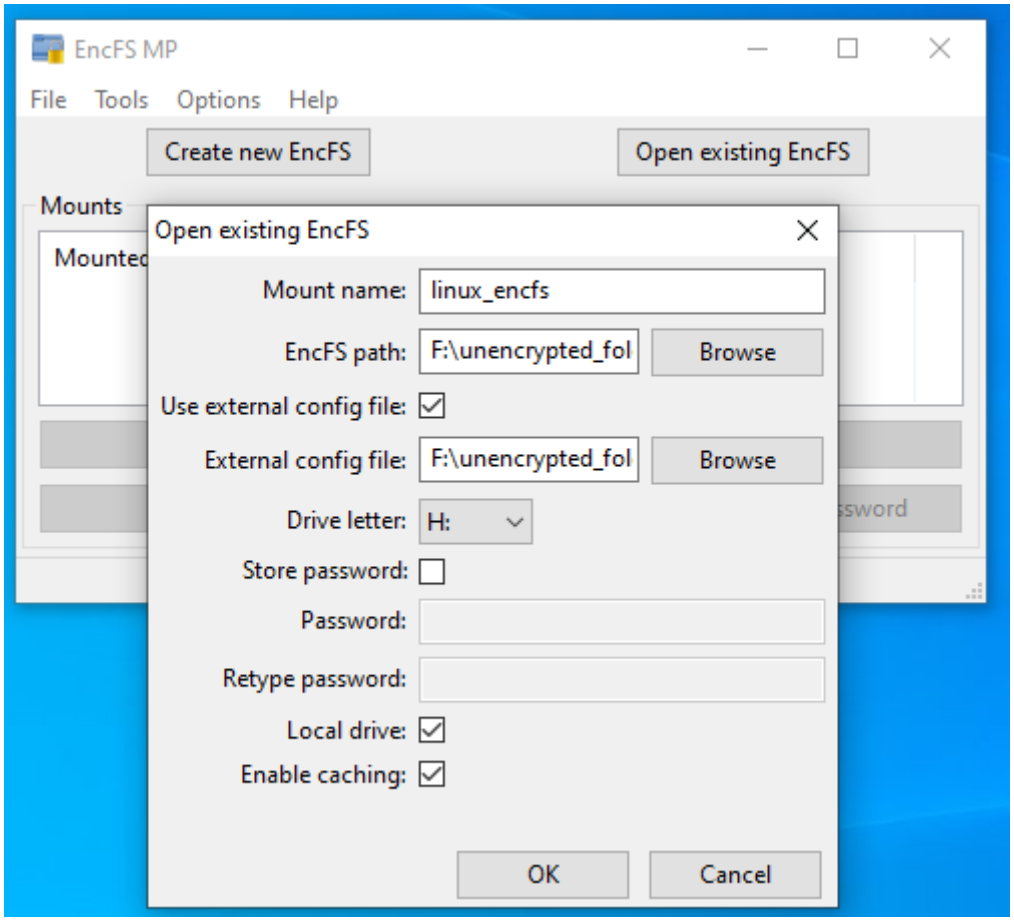


Рисунок 3.7 – Налаштування параметрів для розшифрування вмісту unencrypted_folder

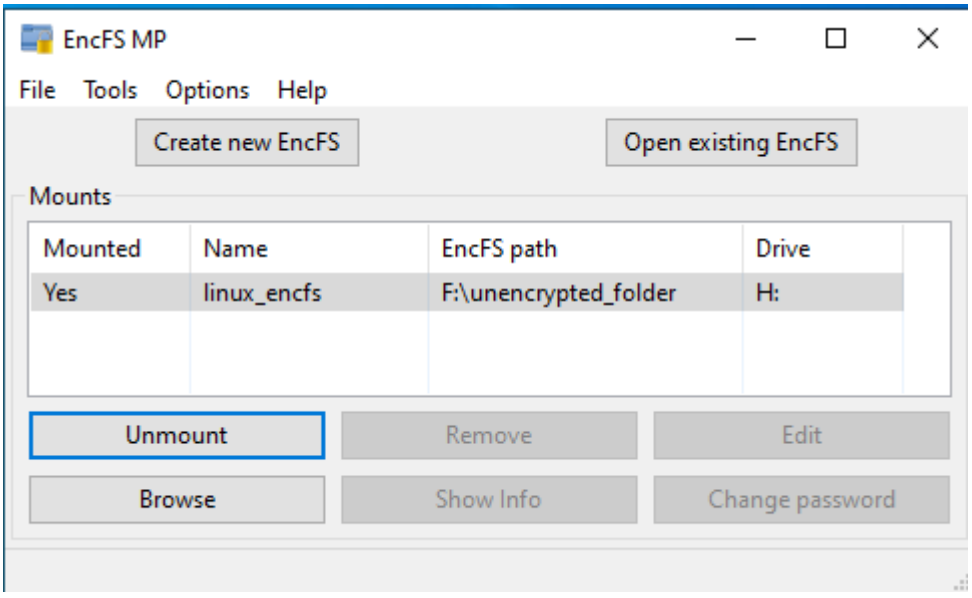


Рисунок 3.8 – Монтування розшифрованого вмісту unencrypted_folder як диск H:

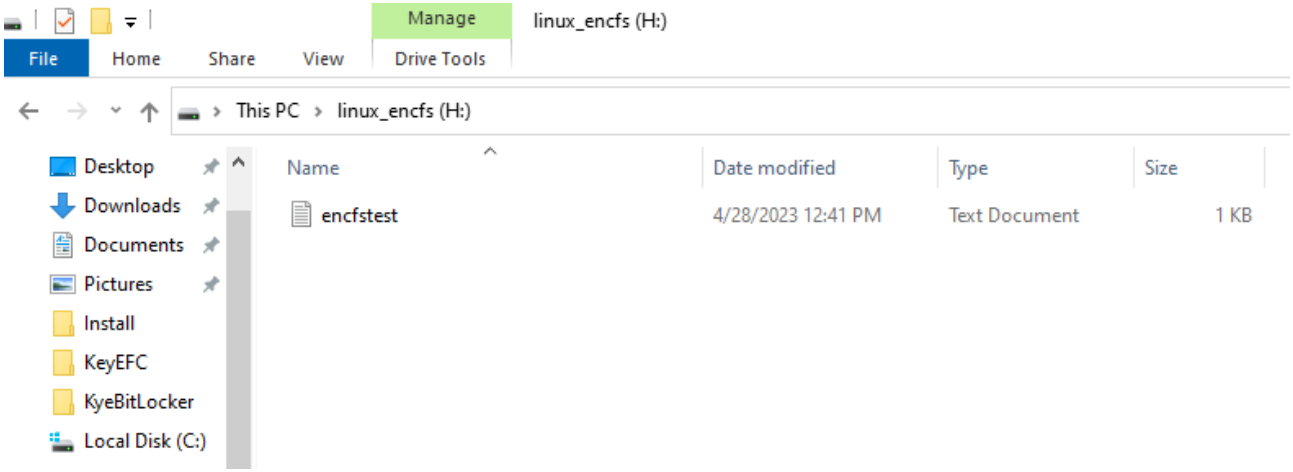


Рисунок 3.9 – Вміст диск H:

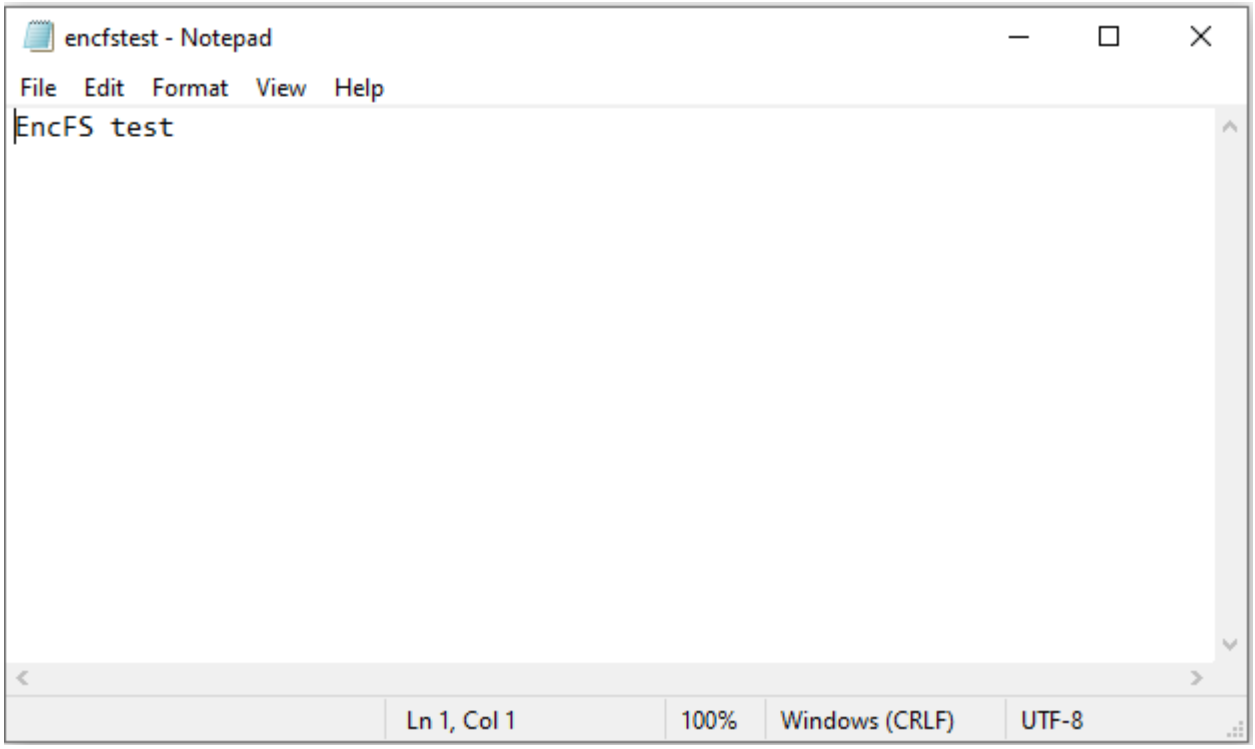


Рисунок 3.10 – Розшифрований вміст файлу encfstest.txt

EncFSMP є потужним та зручним інструментом для захисту конфіденційної інформації за допомогою EncFS на різних операційних системах. Однак, перед використанням EncFSMP, необхідно вивчити документацію та переконатися, що розумієте усі параметри шифрування та налаштування EncFS в операційній системі Linux та Windows , щоб забезпечити максимальний рівень безпеки для своїх даних.

Файли, зашифровані за допомогою Encrypting File System (EFS) в Windows, не можуть бути просто відкриті в Linux за допомогою додаткового програмного

забезпечення, оскільки EFS використовує специфічні ключі шифрування, які зберігаються в захищеному просторі Windows. Копії цих ключів складно опрацьовувати в Linux.  Результат розшифрування є непередбачуваним.

Методи шифрування даних на Windows, такі як BitLocker і EFS, та на Linux, такі як LUKS і EncFS, не є прямо сумісними один з одним. Для обміну шифрованими даними між цими платформами потрібно використовувати сторонніх програм або інструментів.

# 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

## 4.1 Долікарська допомога при ураженні електричним струмом

Українське законодавство містить низку положень, що регулюють питання долікарської допомоги при ураженні електричним струмом.

Зокрема, Закон України "Про охорону праці" від 14.10.1992 № 2694-XII зобов'язує роботодавців забезпечувати безпеку та охорону праці працівників, в тому числі і при роботі з електроустановками. Роботодавець має забезпечити належний стан та регулярну перевірку електроустановок, використовувати електрообладнання, яке відповідає вимогам безпеки та має сертифікат відповідності [16].

Для забезпечення безпеки під час роботи з ПК, на робочому місці повинні бути встановлені заходи технічного захисту від електричного струму. Закон України "Про охорону праці" встановлює вимоги щодо надання працівникам інструктажу з охорони праці та безпеки під час роботи з електроустановками.

У випадку, коли в результаті ураження електричним струмом сталася смерть або тілесні ушкодження потерпілого, винна особа може бути притягнута до кримінальної відповідальності відповідно до Кримінального кодексу України.

Наказ Міністерства охорони здоров'я України від 09.03.2022 р. № 441 " Про затвердження порядків надання домедичної допомоги особам при невідкладних станах" встановлює порядки надання долікарської допомоги особам при невідкладних станах, в тому числі у разі ураження електричним струмом [17].

Отже, у випадку ураження електричним струмом, долікарська допомога має невідкладний характер та повинна надаватися на місці події.

Ось послідовність дій, які потрібно виконати:

1) Припинити контакт з джерелом струму. Для цього можна вимкнути лінію живлення, виключити пристрій, що спричиняє ураження або від'єднати жертву від джерела струму за допомогою ізольованого предмета (наприклад, дерев'яної палиці, резинової шини).

2) Перед наданням допомоги переконатися у відсутності небезпеки для себе, оточуючих, постраждалого та тільки за її відсутності перейти до наступного кроку.
3) Якщо постраждалий у свідомості, заспокоїти та пояснити свої наступні дії.
4) Здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику.
5) Забезпечити постійний нагляд за постраждалим до приїзду бригади екстреної (швидкої) медичної допомоги.
6) При погіршенні стану постраждалого до приїзду бригади екстреної (швидкої) медичної допомоги повторно здійснити виклик екстреної медичної допомоги.
7) За можливості зібрати у постраждалого чи оточуючих максимально можливу інформацію стосовно обставин отримання травми. Всю отриману інформацію передати працівникам бригади екстреної (швидкої) медичної допомоги або диспетчеру служби екстреної медичної допомоги.
8) Якщо до приїзду бригади екстреної (швидкої) медичної допомоги постраждалий втратив свідомість, слід перейти до Порядку надання домедичної допомоги дорослим при раптовій зупинці кровообігу або Порядку надання домедичної допомоги дітям при раптовій зупинці кровообігу, затверджених наказом Міністерства охорони здоров’я України від 09 березня 2022 року № 441.

Потерпілий може мати різні види ушкоджень: від легких опіків до серйозних травм та порушень функцій внутрішніх органів. Необхідність подальшого лікування та реабілітації залежить від ступеня тяжкості ушкодження.

Отже, долікарська допомога при ураженні електричним струмом має важливе значення для збереження життя та здоров'я потерпілого. Важливо дотримуватися вимог законодавства та правил безпеки праці при роботі з

електричними установками, щоб запобігти подібним випадкам та забезпечити безпеку праці на виробництві.

## 4.1 Вплив кольору на покращення умов праці та підвищення продуктивності праці

Кольори можуть впливати на наш настрій, емоційний стан та рівень енергії, що може впливати на продуктивність праці. Дослідження показують, що різні кольори можуть мати різний ефект на працівників залежно від їхньої праці та особистих вподобань.

Синій колір може знижувати агресію та стрес, знижувати пульс, а також підвищувати продуктивність. Він добре підходить для робочих місць, де потрібна концентрація, наприклад, в офісах, де працюють програмісти, адміністратори та інші працівники, що вимагають багато уваги та відповідальності.

Зелений колір є одним з найбільш заспокійливих кольорів, який може знижувати рівень стресу та покращувати настрій. Він добре підходить для робочих місць, де потрібна творчість, наприклад, у мистецьких або дизайнерських студіях.

Червоний колір може збільшувати енергію та стимулювати активність, але водночас може збільшувати рівень стресу та агресії. Він може бути використаний на робочих місцях, де потрібна енергія та активність, наприклад, у спортивних клубах або відділах продажу.

Жовтий колір може підвищувати настрій та енергію, але водночас може викликати відволікання та розсіювання уваги. Він може бути використаний на робочих місцях, де потрібна енергія та веселість, наприклад, у рекламних агентствах або креативних студіях.

Оранжевий колір може підвищувати енергію та стимулювати творчість, але водночас може викликати стрес та роздратування. Він може бути використаний на робочих місцях, де потрібна комунікація та спілкування, наприклад, у відділах клієнтського сервісу або у рекламних агентствах.

Фіолетовий колір може підвищувати концентрацію та творчість, але водночас може викликати меланхолію та депресію. Він може бути використаний на робочих місцях, де потрібна креативність та концентрація, наприклад, у мистецьких або дизайнерських студіях.

Отже кольори можуть мати різний вплив на різні види діяльності та завдання. Наприклад, для робіт, що вимагають великої уваги та концентрації, можуть бути корисними більш спокійні та нейтральні кольори, такі як блідо-блакитний, блідо-зелений чи блідо-сірий. У той же час, для робіт, пов'язаних з креативністю та інноваціями, можна використовувати яскравіші та барвисті кольори, такі як помаранчевий, червоний чи жовтий, що можуть стимулювати творчий потенціал працівників.

Важливо також звернути увагу на сполучення кольорів на робочому місці. Комбінації кольорів можуть викликати різні емоції та впливати на настрій та продуктивність. Наприклад, сполучення зеленого та синього може знижувати стрес та підвищувати продуктивність, тоді як сполучення червоного та жовтого може викликати напругу та роздратування.

Також важливо зазначити, що вибір кольору для робочого місця повинен відповідати не тільки функціональності та емоційному стану працівників, але й бути відповідним з образом компанії та її бренду. Наприклад, якщо компанія має брендові кольори, то вони можуть бути використані для створення узгодженої атмосфери на робочому місці.

Крім того, важливо не перебільшувати вплив кольору на продуктивність працівників та не забувати про інші аспекти, що впливають на умови праці, такі як комфортне освітлення, правильна організація робочого місця та розміщення обладнання.

Отже, вибір кольору для робочого місця може мати значний вплив на умови праці та продуктивність працівників. Варто враховувати характеристики своєї роботи та особисті вподобання при виборі кольорів для робочого місця.

# ВИСНОВКИ

Під час дослідження ефективності та надійності методів шифрування в операційних системах Windows та Linux було встановлено, що обидві операційні системи мають добре розроблені та ефективні методи шифрування даних. У Windows вбудованим методом є BitLocker, який забезпечує надійний рівень захисту даних на локальному диску, а також може бути використаний для шифрування зовнішніх носіїв даних. У Linux найбільш поширеними методами є LUKS, який також забезпечують надійний рівень захисту даних.

Для вибору методу шифрування в конкретних сценаріях використання операційних систем Windows та Linux, потрібно враховувати такі рекомендації:

1) Якщо потрібно забезпечити шифрування всього диску, рекомендується використовувати BitLocker в Windows або LUKS в Linux.
2) Якщо потрібно забезпечити шифрування зовнішнього носія даних, можна використовувати BitLocker To Go в Windows або LUKS з підтримкою USB-накопичувачів в Linux.
3) Якщо потрібно забезпечити захист окремих файлів або папок, можна використовувати вбудовані інструменти шифрування файлів в операційній системі, такі як EFS в Windows або EncFS в Linux.

Під час вибору методу шифрування слід враховувати вимоги до продуктивності та доступності даних, такі як швидкість шифрування та дешифрування, можливість резервного копіювання та відновлення даних.

У загальному, для забезпечення надійного та ефективного шифрування даних в операційних системах Windows та Linux, слід використовувати методи шифрування, які відповідають конкретним вимогам та сценаріям використання, та забезпечувати відповідну захищеність ключів шифрування та паролів. Також важливо регулярно перевіряти ефективність застосованих методів шифрування та оновлювати їх за необхідності.

Для забезпечення найвищого рівня захисту даних, рекомендується використовувати комбінацію різних методів шифрування, наприклад, шифрування всього диску та шифрування окремих файлів або папок. Також

важливо забезпечити фізичну безпеку носіїв даних, щоб запобігти їх втраті або крадіжці.

Застосування методів шифрування даних в операційних системах Windows та Linux може бути корисним для різних сфер діяльності, таких як корпоративні мережі, фінансові установи, медичні заклади, державні організації, а також інші сегменти, де безпека даних має високий пріоритет. Наприклад, використання методів шифрування може допомогти захистити конфіденційну інформацію підприємства від несанкціонованого доступу, втрати даних внаслідок крадіжки або витоку.

Однак, впровадження шифрування даних також може мати свої виклики. Наприклад, забезпечення належного управління ключами шифрування, забезпечення резервного копіювання та відновлення даних, а також забезпечення сумісності з іншими системами та рішеннями можуть бути складними завданнями. Додатково, використання шифрування може впливати на продуктивність системи, збільшуючи навантаження на процесор та збільшуючи час доступу до даних.

Важливо також зазначити, що відповідність вимогам щодо безпеки та захисту даних є постійним процесом, оскільки загрози та атаки на безпеку постійно змінюються та еволюціонують. Тому вибір оптимального методу шифрування має бути відповідно оцінений та адаптований до змінних потреб організації, ризиків та загроз, а також регуляторних вимог.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Горбенко І. Д. Гриненко Т. О. Захист інформації в інформаційно-телекомунікаційних системах: Навч. посібник. Ч.1. Криптографічний захист інформації - Харків: ХНУРЕ, 2004 - 368 с
2. A Graduate Course in Applied Cryptography [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://toc.cryptobook.us/book.pdf
3. Symmetric Cryptography vs Asymmetric Cryptography [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.baeldung.com/cs/symmetric-vs-asymmetric-cryptography
4. Differences Between Stream Cipher and Block [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.baeldung.com/cs/stream-cipher-vs-block-cipher
5. BitLocker [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://learn.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-overview
6. Manage-bde [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/manage-bde
7. How It Works Encrypting File System [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – https://learn.microsoft.com/en-us/previous-versions/technet-magazine/cc160993(v=msdn.10)
8. Cipher [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cipher
9. LUKS2 On-Disk Format Specification [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://gitlab.com/cryptsetup/LUKS2-docs/blob/main/luks2_doc_wip.pdf

10. Encrypt Drives using LUKS on Oracle Linux [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.oracle.com/en/learn/ol-luks/index.html#introduction
11. Encrypting block devices using LUKS [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/encrypting-block-devices-using-luks_security-hardening#luks-disk-encryption_encrypting-block-devices-using-luks
12. EncFS [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://github.com/vgough/encfs/blob/master/encfs/encfs.pod
13. Dislocke [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://github.com/Aorimn/dislocker
14. Dislocker-fuse [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://manpages.ubuntu.com/manpages/jammy/man1/dislocker.1.html
15. EncFSMP FAQ [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://encfsmp.sourceforge.io/faq.html
16. Закон України "Про охорону праці" від 14.10.1992 № 2694-XII, Version 1 [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text
17. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0356-22#n769